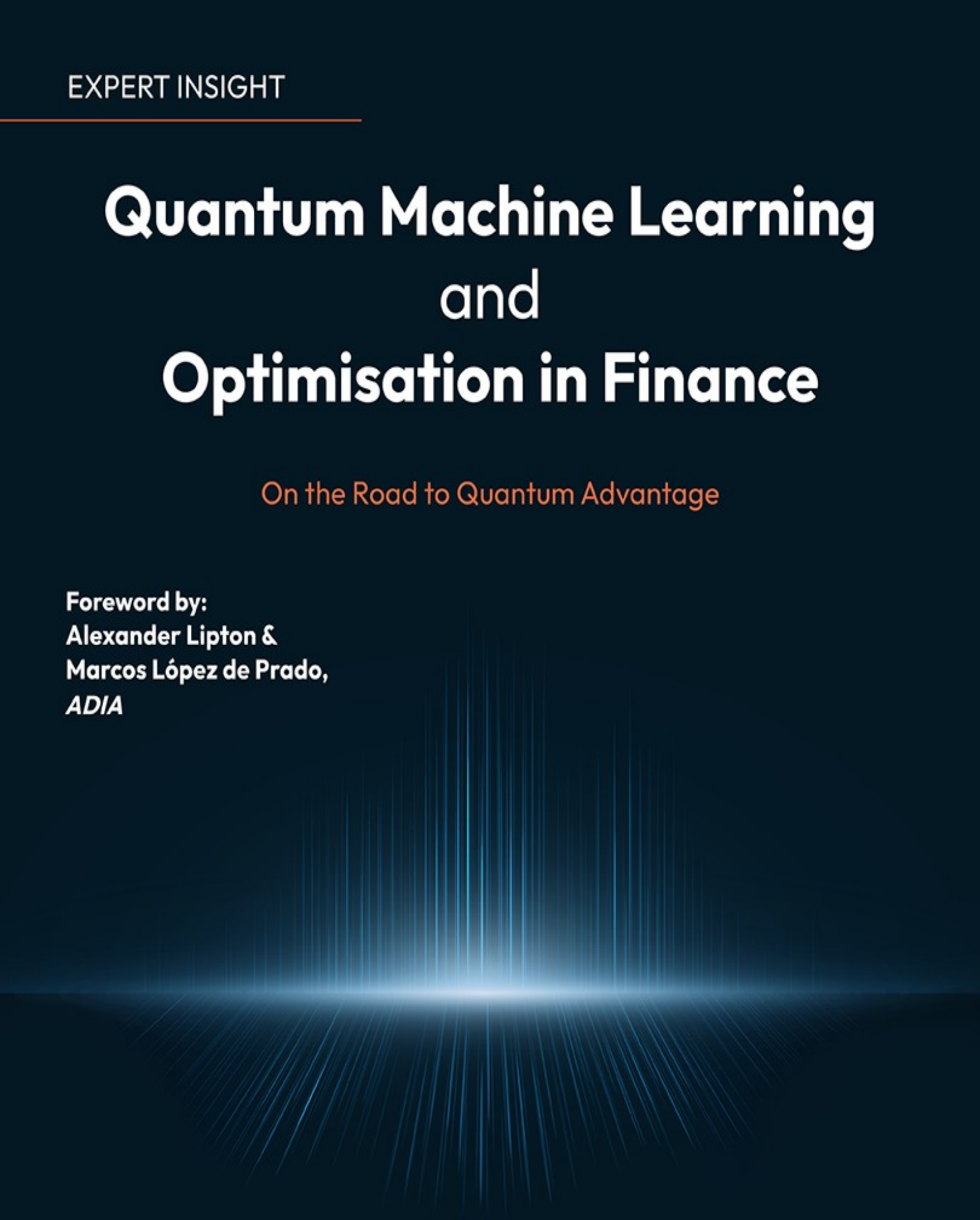
EXPERT INSIGHT

Quantum Machine Learning and Optimisation in Finance

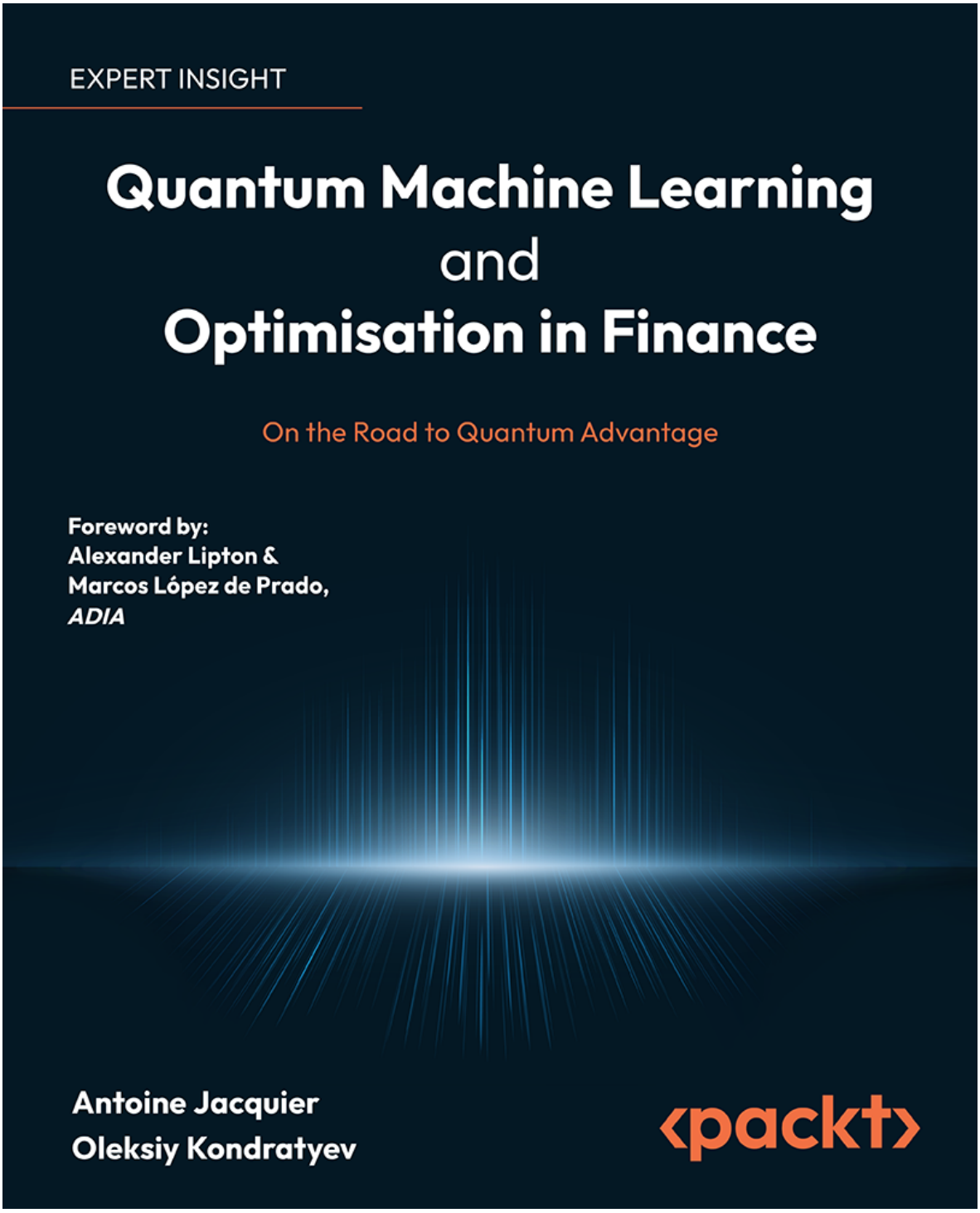
On the Road to Quantum Advantage

Foreword by:
Alexander Lipton &
Marcos López de Prado,
ADIA



Antoine Jacquier
Oleksiy Kondratyev

‹packt›


EXPERT INSIGHT
Quantum Machine Learning
and
Optimisation in Finance
On the Road to Quantum Advantage
Foreword by:
Alexander Lipton &
Marcos López de Prado,
ADIA
Antoine Jacquier
Oleksiy Kondratyev
packt

Quantum Machine Learning and Optimisation in Finance

On the Road to Quantum Advantage

Antoine Jacquier

Oleksiy Kondratyev

BIRMINGHAM—MUMBAI

Quantum Machine Learning and Optimisation in Finance

Copyright © 2022 Packt Publishing All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews. Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the authors, nor Packt Publishing or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book. Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

Senior Publishing Product Manager: Tushar Gupta

Acquisition Editor – Peer Reviews: Saby D'silva

Project Editor: Amit Ramadas

Development Editor: Bhavesh Amin

Copy Editor: Safis Editing

Technical Editor: Aniket Shetty

Proofreader: Safis Editing

Presentation Designer: Sandip Tadge

First published: October 2022

Production reference: 1211022

Published by Packt Publishing Ltd.

Livery Place

35 Livery Street

Birmingham

B3 2PB, UK.

ISBN 978-1-80181-357-0

www.packt.com

To Loulou and the three musketeers

- A.J.

To my family and in memory of Viktor Yatsunyk

- OK

Foreword

As far as computational prospects (including hardware and software) and related developments are concerned, we live in the most exciting times. Every day brings new achievements, new promises, and, on occasion, new disappointments. Presently, some of the ideas discussed for decades are reaching the usable stage, the most exciting developments being thermonuclear fusion, all-purpose artificial intelligence, distributed ledger technology, and quantum computing. However, not surprisingly, breakthroughs in these fields are very hard to achieve, so, despite strenuous government and private efforts and lavish funding, their stated goals have not been reached yet.

Jacquier and Kondratyev, two of the strongest quants of their generation, have written a remarkable book about quantum computers and their applications in finance, emphasizing practical aspects of quantum machine learning and optimisation. To put the subject of this book in a proper context, let us briefly touch upon the history of computing devices. Without exaggeration, one can claim that the history of human progress is closely related to the history of computing devices. Original simple but beneficial instruments, such as abacuses, were invented at the dawn of civilization. Eventually, computing devices developed into potent tools, such as supercomputers, that define our day-to-day existence and future as species.

To start with, it is worth mentioning some milestones. Abacuses were used in Babylonia as early as c. 2700–2300 BC; eventually, they spread worldwide and became known as Roman abacuses.

Computational devices were used together with memory devices, such as tally sticks and clay tablets. In classical antiquity, analog computers, such as the fabled Antikythera, probably intended for astronomical calculations, were actively used. Subsequently, medieval Muslim astronomers and engineers brought such devices to the next level. As a result, they developed remarkable objects, such as sundials, planispheres, and astrolabes.

Eventually, the center of progress moved to Europe and later to North America, where various calculating tools, such as the slide rule and mechanical calculator, were introduced in early modern times. However, the real breakthrough was achieved in 1804 when Jacquard created a loom programmable via punched cards. Babbage, "the father of the computer", invented the first mechanical computer programmable with punch cards in the United Kingdom. Unfortunately, his Difference Engine and Analytical Engine were never completed because their execution involved severe practical issues, including insurmountable engineering obstacles, lack of funding, and general ridicule (Babbage's story should serve as a fair warning to all the intrepid inventors who come with ideas centuries ahead of their time). Several decades later, the American inventor Hollerith was much more successful. He used punched cards to store data readable by a tabulator machine; IBM can trace its origins to these humble cards.

In the late 1800s to early 1900s, analog computers, such as tide-predicting machines, using physical phenomena to model the problem they were built to solve, became all the rage. Specifically, an analog computer's developers must find physical processes governed by the same or similar equations as the problem of

actual interest. Since, by design, such computers use noisy continuous values subject to various errors, they produce approximate solutions. However, such solutions are highly beneficial in many applications, including warfare, navigation, and economics, to mention but a few. One of the more remarkable examples of analog computers is the Phillips Hydraulic Computer, using simple tools to control the water's flow to model the national economy of the United Kingdom with a 2% accuracy. Others are the battleship's fire-control systems and bomb sights. Shortly before and during World War II, digital computers evolved into formidable competitors of analog computers and, by the 1950s, replaced them altogether, except for some highly specialized applications. Stemming from theoretical insights due to Turing, who invented the celebrated eponymous Turing Machine, and several other pioneers and practical engineering inventions, such as vacuum tubes and transistors, digital computers conquered the world. Initially, digital computers, such as the Ananasov-Berry computer, the Colossus, and the ENIAC, to mention but a few, were enormous machines built to serve military needs, including nuclear weapons design and cryptography. Eventually, computers found a wide plethora of commercial applications.

Modern digital computers store the necessary data in a magnetic memory in the form of 0-1 bits; they operate on this data via logical gates. The fact that the data is digitised has profound implications. On the one hand, while undeniably humongous, the amount of data a computer can store and the speed at which it can process this data are limited. On the other hand, inevitable errors inherent in any physical device are relatively easy to control.

Quantum computers were independently conceived in the early 1980s by Benioff, who proposed a quantum version of the Turing Machine, Feynman, and Manin. Feynman, with his usual eloquence, stated: "Nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical, and by golly, it's a wonderful problem because it doesn't look so easy". Quantum computers use quantum mechanical phenomena to solve computational problems that conventional computers cannot solve efficiently (or at all). To build a working quantum computer, one must overcome tremendous difficulties and achieve conditions such that quantum effects become dominant or, at least, noticeable. For example, such a computer must operate at low temperatures close to absolute zero. The primary distinction between quantum computers and their classical brethren is how they store and operate on the data. Instead of the 0-1 bits used by digital computers, quantum computers use the so-called qubits, capable of storing continuous information represented by a point on the so-called Bloch sphere; however, once measured, the data collapse to the classical 0-1 state. In theory, qubits should be entangled so that their quantum states depend on the state of all other qubits. The data, stored by a quantum computer, is processed via quantum gates. Since the data is continuous rather than discrete, the storage problem is not an issue per se; however, the continuity of the data makes quantum computers prone to inevitable errors. As a result, fault-tolerant quantum computing with suitably many qubits remains a very distant possibility. Furthermore, building an actual qubit is very difficult. Different possibilities have been tried, such as Josephson's junctions, trapped ions, and many others.

Quantum computing has numerous exciting prospective goals, although it is unclear if all (or any) of them can be reached. The best known is cryptography, more concretely, factorising huge integers into prime factors using Shor's algorithm, thus breaking conventional asymmetric encryption. It is worth mentioning that Shor's algorithm is probabilistic; hence, it is particularly well adapted to quantum computers. The promise of achieving the factorisation breakthrough galvanised governments and private companies into pouring billions of dollars into quantum computing and related fields. For instance, the US government launched the National Quantum Initiative to explore and promote Quantum Information Science. In addition, the National Institute of Standards and Technology evaluates and standardizes quantum-resistant public-key cryptographic algorithms. However, the biggest number reliably factored by Shor's algorithm is 35, factored in 2021 using a computer with very few qubits. Another exciting application is to search problems, which can be efficiently handled with Grover's algorithm, although practical implementations are lacking. Several research groups, such as the Google-NASA collaboration, actively study machine learning applications. Nevertheless, large-scale applications are still far from being conquered.

This situation brings us back to the Jacquier and Kondratyev book. Given the relatively slow progress achieved in quantum computation, it is natural to go back to the early days of modern computing hardware and see if one can use a quantum-like computer as a modern analog computer; this is precisely what Jacquier and Kondratyev do. Specifically, rather than waiting for quantum computers to reach their top form, they advocate using quantum annealing computers and similar analog machines to

perform the required calculations. Such computers start with a simple Hamiltonian with a known ground state and slowly (adiabatically) evolve the original Hamiltonian into the Hamiltonian of interest while always preserving the ground state. Thus, when the process is completed, the measurement allows one to find the ground state for the actual problem rather than a simplified one. Jacquier and Kondratyev use analog quantum computers to solve several exciting mathematical finance problems with verve and panache. Specifically, they discuss quantum boosting and demonstrate how to apply it to predict credit card defaults and solve classification problems. Subsequently, Jacquier and Kondratyev turn their attention to quantum Boltzmann machines and explain how to use them for distribution sampling.

Analog quantum computers can be used to solve many financial engineering problems. For instance, a quantum annealer is a perfect tool for solving the multi-period integer portfolio optimisation problem, which is NP-Complete. Monte Carlo simulations, which are fundamental for derivatives pricing and other related tasks, are particularly natural to perform using quantum computers since they can be viewed as true random number generators.

When it comes to the digital gate model quantum computing, Jacquier and Kondratyev discuss quantum neural networks and their applications to machine learning. Later, the authors cover Quantum Circuit Born Machine, Variational Quantum Eigensolver, and the Quantum Approximate Optimisation Algorithm. Finally, Jacquier and Kondratyev discuss new quantum algorithms, such as quantum kernels, Bayesian

Quantum Circuit, Quantum Fourier Transform, and Quantum Monte Carlo Simulation.

Jacquier and Kondratyev are excited about quantum algorithms and their potential applications; however, they are not starry-eyed and approach quantum algorithms cautiously. Specifically, they put much effort into showing how classical algorithms can solve problems they are interested in and when quantum algorithms outperform their classical brethren.

While building quantum computers originally envisioned by Feynman and Manin might still be decades away, more practical analog quantum computers already exist. Experience suggests that finance is one of the fields where breakthroughs in computing tend to be used in real-time. Quants who want to use quantum algorithms in their day-to-day work could scarcely do better than starting their journey by studying this book.

Bon voyage!

Alexander Lipton and Marcos López de Prado,
Abu Dhabi Investment Authority

Endorsements

Quantum Computing is often described as a transformative and disruptive technology. The potential is clear; however, currently, hardware does not yet offer the necessary performance to harness this potential. As the technology scales, four main problem types will become addressable. Simulation – in areas such as fluid dynamics, chemical catalysis, and structural biology; Machine Learning – in areas such as the use of AI in autonomous vehicles, personalised medicine, and fraud detection; Optimisation – in areas such as resourcing and logistics; and Cryptography – in areas of security and systems resilience.

In this book, Antoine and Oleksiy marry their expertise in mathematics and algorithm development with a deep understanding of and experience in global banking to address the problem types and current approaches in quantum-enabled machine learning and optimisation. Step by step, they take the reader through familiarisation with the methodology and mathematics of quantum mechanics and implementations of machine learning and optimisation in quantum annealers and gate model architectures. They describe both long-term, large-scale approaches and more practical, near-term NISQ-era variational and approximate techniques. Boston Consulting forecasts global financial services will see a $70-135bn impact from quantum computing in the coming two decades. $12.6tr of managed portfolio funds are located in the UK, with the UK finance

sector contributing £132bn to the UK economy, employing over 1.1m people in 2019.

Recent surveys of UK business leaders suggest that 94% believe quantum computing will impact their organisation or sector significantly, and 72% intend to start strategic planning or create a pilot team by 2024. Only 6% have development teams in place today. In this book, Antoine and Oleksiy have created a toolkit to help focus development teams within the finance sector and beyond on the core areas of understanding and impact of quantum computing.

Demonstrating value through the application of quantum computing from the realms of physics and mathematics is a challenging one; nevertheless, the finance sector is an area ripe with computational challenges – particularly in machine learning and optimisation – that lend themselves both to business advantage and opportunities for quantum advantage.

—Dr. Michael N Cuthbert, Director, National Quantum Computing Centre

This book reviews different types of quantum computers as well as some important principles and algorithms. Quantum Machine Learning and Optimization in Finance is a great book to learn more about these two important applications of quantum computing.

—Dr. Ray O. Johnson, CEO, Technology Innovation Institute

The book is a wonderful guide for both the quantum computing scientist interested in understanding the applications of quantum computing in quantitative finance and the finance professional looking to explore the new computational tools offered by quantum computing.

Without giving up mathematical rigour, the authors manage to explain clearly and make order in the new and still growing discipline of quantum computing applied to quantitative finance.

This is no small feat, and the authors take us on a journey that starts from the foundations of quantum mechanics and leads to quantum annealing, gate-based quantum computing, and their algorithms. The journey never departs from the practical relevance of the subject to quantitative finance: several algorithms and numerical finance use cases and applications are developed explicitly and extensively.

Despite the complexity of the subject, the book is highly readable and successfully manages to illustrate and combine rigorous quantum computing methods with their numerous practical exemplifications in quantitative finance.

—Marco Paini, Director, Technology Partnerships Europe at Rigetti Computing

Jacquier and Kondratyev succeed in leading the reader on a journey through both foundational concepts and selected research material in a very pedagogical and pleasant way.

By studying the book, readers will be well equipped to understand, formulate, and attack the main challenges of current quantum computing experiments applied to financial problems.

—Davide Venturelli, Ph.D., Associate Director of Quantum Computing, USRA

Contributors

About the authors

Antoine Jacquier graduated from ESSEC Business School before obtaining a PhD in Mathematics from Imperial College London.

His research focuses on stochastic analysis, asymptotic methods in probability, volatility modelling, and algorithms in quantum computing. He has published about 50 papers and has co-written several books. He is also the Director of the MSc in Mathematics and Finance at Imperial College and regularly works as a quantitative consultant for the Finance industry.

He has a keen interest in running and whisky.

I would like to thank all the people who challenged and motivated me in Applied Mathematics, asking the right questions, pointing to interesting problems, and proposing diverse solutions from many angles. Chief among them are Jim Gatheral, Peter Friz, Mathieu Rosenbaum, Josef Teichmann, Mark Davis, Claude Martini, and Aleksandar Mijatović, whose constant and generous ideas led to numerous advances in my career.

My journey through quantum computing, while not on the obvious path dictated by my mathematical background, started out of this scientific curiosity acquired over the years. I am indebted to Kostas Kardaras, Mugad Oumgari, Alexandros Pavlis, and Amine Assouel for crawling

patiently with me through the dark meanders of quantum computing and quantum mechanics.

Oleksiy Kondratyev obtained his PhD in Mathematical Physics from the Institute for Mathematics, National Academy of Sciences of Ukraine, where his research was focused on studying phase transitions in quantum lattice systems.

Oleksiy has over 20 years of quantitative finance experience, primarily in banking. He was recognised as Quant of the Year 2019 by Risk magazine and joined Abu Dhabi Investment Authority as a Quantitative Research & Development Lead in the summer of 2021.

Outside the world of finance and quantum computing, Oleksiy's passion is for sailing, in particular offshore racing. Oleksiy holds the RYA Yachtmaster Ocean certificate of competence and is a member of the Royal Ocean Racing Club.

I would like to thank Davide Venturelli for introducing me to the wonderful world of quantum computing. As a physicist, I am fascinated by the possibility of performing computation using quantum mechanical systems and as a quant, I appreciate the impact quantum computing will have on quantitative finance.

I am deeply grateful to Bill Winters for his support and interest in quantum computing. It takes both vision and courage to back the development of new emerging technology that does not promise immediate payoff but has the potential to change the finance industry as we know it.

Writing a book is a long journey and a result of many years of intensive research. This book would not have been possible without the discussions, collaborations, exchange of ideas, and support of Majed Al Romaithi, Bhavesh Amin, David Bell, Michael Brett, Kasper Christoffersen, Brian Coyle, Michael Cuthbert, Tushar Gupta, Max Henderson, Mark Hodson, Blanka Horvath, Wendy Huang, Ray Johnson, Elham Kashefi, Geoff Kot, Alexander Lipton, Charissa Liu, Marcos López de Prado, Aaron Lott, Alex Manson, Roger McKinley, Ashley Montanaro, Krzysztof Osiewalski, Marco Paini, Manos Papathanasiou, Amit Ramadas, Chad Rigetti, Christian Schwarz, Elena Strbac, Robert Sutor, Agnieszka Verlet, José Viñals, Colin Williams, Stefan Wörner, and Safis Editing.

About the reviewer

Gerhard Hellstern (Prof., Dr. rer. nat, graduate physicist, *1971) has been a professor at the Faculty of Economics at the Baden-Württemberg Cooperative State University in Ravensburg since 2018. From 1990 - 1995, he studied physics at the University of Tübingen and the State University of New York at Stony Brook; in 1998, he graduated as Dr. rer. nat. From 1998 to 2018, he was employed by several commercial banks and then for 17 years at Deutsche Bundesbank. He was in charge of the banking audits division for many years. Gerhard Hellstern has been involved in the application of data science methods (data analytics as well as machine and deep learning) in finance for many years. These methods also include Quantum Computing and Quantum Machine Learning. He is a Qiskit advocate at IBM and a member of the research network Quantum Computing of the Fraunhofer Gesellschaft. His current research focuses on applications of Quantum Computing / Quantum Machine Learning in the financial sector and beyond, and he has published several papers in this domain.

Join our book's Discord space

Join our Discord community to meet like-minded people and learn alongside more than 2000 members at:
https://packt.link/quantum



Table of Contents

1. The Principles of Quantum Mechanics
 1.1 Linear Algebra for Quantum Mechanics
  1.1.1 Basic definitions and notations
  1.1.2 Inner products
  1.1.3 From linear operators to matrices
  1.1.4 Condition number
  1.1.5 Matrix decompositions and spectral theorem
  1.1.6 Hermitian matrices
  1.1.7 Rotation matrices
  1.1.8 Polar coordinates
  1.1.9 Dirac notations
  1.1.10 Quantum operators
  1.1.11 Tensor product
 1.2 Postulates of Quantum Mechanics
  1.2.1 First postulate – Statics
  1.2.2 Second postulate – Dynamics
  1.2.3 Third postulate – Measurement
  1.2.4 Fourth postulate – Observable
  1.2.5 Fifth postulate – Composite System
 1.3 Pure and Mixed States
  1.3.1 Density matrix
  1.3.2 Pure state
  1.3.3 Mixed state

I Analog Quantum Computing – Quantum Annealing

2. Adiabatic Quantum Computing
 2.1 Complexity of Computational Problems
 2.2 Principles of Adiabatic Quantum Computing

2.2.1 The Quantum Adiabatic Theorem
2.2.2 Optimisation and metaheuristics
2.3 Implementations of AQC
2.3.1 The short history of quantum annealing
2.3.2 Inter-generational comparison of D-Wave quantum annealers
2.3.3 Physical realisations of quantum annealers
2.3.4 Chimera graph and embedding of the logical qubits
2.4 Universality of AQC
3. Quadratic Unconstrained Binary Optimisation
3.1 Principles of Quadratic Unconstrained Binary Optimisation
3.1.1 QUBO to Ising transformation
3.1.2 QUBO problem examples
3.2 Forward and Reverse Quantum Annealing
3.2.1 Forward quantum annealing
3.2.2 Reverse quantum annealing
3.3 Discrete Portfolio Optimisation
3.3.1 QUBO encoding
3.3.2 The coarse-grained encoding scheme
3.3.3 Construction of the instance set for numerical experiments
3.3.4 Classical benchmark – Genetic Algorithm
3.3.5 Establishing quantum speedup
4. Quantum Boosting
4.1 Quantum Annealing for Machine Learning
4.1.1 General principles of the QBoost algorithm
4.1.2 QUBO to Ising
4.2 QBoost Applications in Finance

4.2.1 Credit card defaults
4.2.2 QUBO classification results
4.3 Classical Benchmarks
4.3.1 Artificial neural network
4.3.2 Training artificial neural networks
4.3.3 Decision trees and gradient boosting
4.3.4 Benchmarking against standard classical classifiers
5. Quantum Boltzmann Machine
5.1 From Graph Theory to Boltzmann Machines
5.2 Restricted Boltzmann Machine
5.2.1 The RBM as an energy-based model
5.2.2 RBM network architecture
5.2.3 Sample encoding
5.2.4 Boltzmann distribution
5.2.5 Extensions of the Bernoulli RBM
5.3 Training and Running RBM
5.3.1 Training RBM with Boltzmann sampling
5.3.2 The Contrastive Divergence algorithm
5.3.3 Generation of synthetic samples
5.4 Quantum Annealing and Boltzmann Sampling
5.4.1 Boltzmann sampling
5.4.2 Mapping
5.4.3 Hardware embedding and parameters optimisation
5.4.4 Generative models
5.5 Deep Boltzmann Machine
5.5.1 Training DBMs with quantum annealing
5.5.2 A DBM pipeline example
II Gate Model Quantum Computing
6. Qubits and Quantum Logic Gates

6.1 Binary Digit (Bit) and Logic Gates
6.1.1 Logic gates
6.1.2 NAND as a universal logic gate
6.1.3 Building an addition operator from the NAND gates
6.2 Physical Realisations of Classical Bits and Logic Gates
6.2.1 Implementation of the NAND gate
6.2.2 Implementation of the RAM memory cell
6.3 Quantum Binary Digit (Qubit) and Quantum Logic Gates
6.3.1 Computation according to the laws of quantum mechanics
6.3.2 Qubit
6.3.3 One-qubit quantum logic gates
6.3.4 Two-qubit quantum logic gates
6.3.5 The Toffoli gate
6.4 Reversible Computing
6.5 Entanglement
6.5.1 Quantum entanglement and why it matters
6.5.2 Entangling qubit states with two-qubit gates
6.6 Quantum Gate Decompositions
6.7 Physical Realisations of Qubits and Quantum Gates
6.7.1 The DiVincenzo criteria
6.7.2 Superconducting qubits
6.7.3 Photonic qubits
6.7.4 Trapped ion qubits
6.8 Quantum Hardware and Simulators
7. Parameterised Quantum Circuits and Data Encoding
7.1 Parameterised Quantum Circuits

7.2 Angle Encoding
7.2.1 The basic encoding scheme
7.2.2 Encoding two features per quantum register
7.2.3 Mapping a classical data sample into a quantum state
7.3 Amplitude Encoding
7.4 Binary Inputs into Basis States
7.5 Superposition Encoding
7.6 Hamiltonian Simulation
8. Quantum Neural Network
8.1 Quantum Neural Networks
8.2 Training QNN with Gradient Descent
8.2.1 The finite difference scheme
8.2.2 The analytic gradient approach
8.2.3 The parameter shift rule for analytic gradient calculation
8.3 Training QNN with Particle Swarm Optimisation
8.3.1 The Particle Swarm Optimisation algorithm
8.3.2 PSO algorithm for training quantum neural networks
8.4 QNN Embedding on NISQ QPU
8.4.1 NISQ QPU connectivity
8.4.2 QNN embedding scheme
8.5 QNN Trained as a Classifier
8.5.1 The ACA dataset and QNN ansatz
8.5.2 Training an ACA classifier with the PSO algorithm
8.6 Classical Benchmarks
8.6.1 Logistic Regression and Random Forest
8.6.2 Benchmarking against standard classical

classifiers
8.7 Improving Performance with Ensemble Learning
8.7.1 Majority voting
8.7.2 Quantum boosting
9. Quantum Circuit Born Machine
9.1 Constructing QCBM
9.1.1 QCBM architecture
9.1.2 QCBM embedding
9.2 Differentiable Learning of QCBM
9.2.1 Sample encoding
9.2.2 Choosing the right cost function
9.3 Non-Differentiable Learning of QCBM
9.3.1 The principles of Genetic Algorithm
9.3.2 Training QCBM with a Genetic Algorithm
9.4 Classical Benchmark
9.5 QCBM as a Market Generator
9.5.1 Non-parametric modelling of market risk factors
9.5.2 Sampling from the learned probability distributions
9.5.3 Training algorithm convergence and hyperparameter optimisation
10. Variational Quantum Eigensolver
10.1 The Variational Approach
10.2 Calculating Expectations on a Quantum Computer
10.2.1 The one-qubit case
10.2.2 The two-qubit case
10.2.3 The multi-qubit case
10.3 Constructing the PQC
10.3.1 One-qubit ansatz
10.3.2 Multi-qubit ansatz

10.4 Running the PQC
10.4.1 Experimenting with the two-qubit ansatz
10.4.2 Analysis of the obtained results
10.5 Discrete Portfolio Optimisation with VQE
11. Quantum Approximate Optimisation Algorithm
11.1 Time Evolution
11.2 The Suzuki-Trotter Expansion
11.3 The Algorithm Specification
11.4 The Max-Cut Problem
11.4.1 QAOA gates
11.4.2 QAOA circuit
12. The Power of Parameterised Quantum Circuits
12.1 Strong Regularisation
12.1.1 Lipschitz constant
12.1.2 Regularisation example
12.2 Expressive Power
12.2.1 Multilayer PQC
12.2.2 Tensor network PQC
12.2.3 Measures of expressive power
12.2.4 Expressive power of PQC
13. Looking Ahead
13.1 Quantum Kernels
13.1.1 Classical kernel method
13.1.2 Quantum kernel method
13.1.3 Quantum circuits for the feature maps
13.2 Quantum Generative Adversarial Networks
13.3 Bayesian Quantum Circuit
13.4 Quantum Semidefinite Programming
13.4.1 Classical semidefinite programming
13.4.2 Maximum risk analysis

13.4.3 Robust portfolio construction
13.4.4 Quantum semidefinite programming
13.5 Beyond NISQ
13.5.1 Quantum Fourier Transform
13.5.2 Quantum Phase Estimation
13.5.3 Monte Carlo speedup
13.5.4 Quantum Linear Solver
Biblography
Other_Books_You_May_Enjoy
Index

Standard notations

$\mathbb{C}$	Complex numbers
$\mathcal{H}$	Hamiltonian
$\mathcal{I}$	Identity operator
log	Natural logarithm
$\log_2$	Logarithm base 2
$\mathbb{N}$	Natural numbers
$\mathbb{P}$	Probability
$\mathcal{P}$	Projection operator (measurement operator)
q	Vector of binary variables, $q := (q_1,\ldots,q_N)$, $(q_i \in \{0,1\})_{i=1,\ldots,N}$
$\mathbb{R}$	Real numbers
s	Vector of spin variables, $s := (s_1,\ldots,s_N)$, $(s_i \in \{-1,+1\})_{i=1,\ldots,N}$

$\sigma_\alpha^{\mathrm{i}}$	Pauli operator ($\alpha \in \{x,y,z\}$) acting on qubit i
u	Variable (scalar)
u	Column vector
$\mathrm{u}^\top$	Transpose of a column vector
u^*	Complex conjugate of a column vector
$\|\mathrm{u}\rangle$	Ket: column vector in Dirac notation
$\langle\mathrm{u}\|$	Bra: complex conjugate transpose of column vector in Dirac notation
$\langle\mathrm{u}\|\mathrm{v}\rangle$	Inner product
$\|\mathrm{u}\rangle\langle\mathrm{v}\|$	Outer product
$\|\mathrm{u}\rangle\otimes\|\mathrm{v}\rangle$	Tensor product
U	Matrix
U	Quantum logic gate
$\mathcal{U}$	Operator
$\mathcal{U}^\dagger$	Adjoint operator
$\langle\psi\|\mathcal{U}\|\psi\rangle$	Expectation value of operator $\mathcal{U}$ in state $\|\psi\rangle$

Standard abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
AQC	Adiabatic Quantum Computing
BQC	Bayesian Quantum Circuit
CD	Contrastive Divergence

CNN	Convolutional Neural Network
DBM	Deep Boltzmann Machine
DNN	Deep Neural Network
FN	False Negative
FP	False Positive
GA	Genetic Algorithm
GAN	Generative Adversarial Network
LSTM	Long Short Term Memory
ML	Machine Learning
MLP	Multi-Layer Perceptron
MPQC	Multilayer Parameterised Quantum Circuit
MPS	Matrix Product State
MPT	Modern Portfolio Theory
NISQ	Noisy Intermediate Scale Quantum
NP	Non-deterministic Polynomial
PDE	Partial Differential Equation
PDF	Probability Density Function
PQC	Parameterised Quantum Circuit
PSO	Particle Swarm Optimisation
QA	Quantum Annealing
QAO	Quantum Anharmonic Oscillator
QAOA	Quantum Approximate Optimisation Algorithm

QAML	Quantum Annealing for Machine Learning
QBM	Quantum Boltzmann Machine
QCBM	Quantum Circuit Born Machine
QCNN	Quantum Convolutional Neural Network
QFT	Quantum Fourier Transform
QGAN	Quantum Generative Adversarial Network
QHO	Quantum Harmonic Oscillator
QLS	Quantum Linear Solver
QLSTM	Quantum Long Short Term Memory
QMC	Quantum Monte Carlo
QML	Quantum Machine Learning
QNN	Quantum Neural Network
QPE	Quantum Phase Estimation
QPU	Quantum Processing Unit
QRAM	Quantum Random Access Memory
QSDK	Quantum Software Development Kit
QSDP	Quantum Semidefinite Programming
QUBO	Quadratic Unconstrained Binary Optimisation
QVA	Quantum Variational Autoencoder
RAM	Random Access Memory
RBM	Restricted Boltzmann Machine
ReLU	Rectified Linear Unit
SVD	Singular Value Decomposition

SVM	Support Vector Machine
TN	True Negative
TP	True Positive
TPQC	Tensor Network Parameterised Quantum Circuit
TTS	Time-to-Solution
VQE	Variational Quantum Eigensolver

Preface

Quantum Machine Learning – Most **overhyped** and **underestimated** field at the same time

---

Iordanis Kerenidis

Introduction

Why Quantum Computing? Quantum computing and AI will revolutionise and disrupt our society in the same way as classical digital computing did in the second half of the 20th century and the internet did in the first two decades of the 21st century.

Quantum computing (or, more generally, Quantum Information Theory) has been the subject of extensive research since the 1960s, but it was only in the last decade that progress on the hardware side has made it possible to test quantum computing algorithms; and it was only in the last several years that quantum computing's supremacy was finally claimed as an experimental fact (i.e., a landmark experiment conducted on Google's 53-qubit Sycamore quantum chip [16]).

The story of quantum computing is, in this respect, similar to the story of AI: AI was born in the 1950s but then experienced two "winters", when interest in AI and Machine Learning declined considerably (following the Lighthill report in the UK and the Speech Understanding Research debacle in the US in the 1970s, and the LISP collapse in the 1990s), before becoming widely used

and adopted to the point that we can no longer imagine our life without it.

Even though we cannot rule out a "quantum computing winter" before quantum computing technology becomes embedded in everyday life to the same extent as the internet, smartphones, and AI, the whole range of quantum computing breakthroughs we have witnessed in the last few years makes it somewhat unlikely.

With recent advances in the field, we have finally reached the era of Noisy Intermediate-Scale Quantum (NISQ) computing [237]. NISQ-era computers are powerful enough to test quantum computing algorithms and solve non-trivial real-world problems – and establish quantum speedup and quantum advantage over comparable classical hardware.

However, it is likely that the first real-world production-level business applications will be a hybrid quantum-classical protocol, where most of the computation and data processing is done classically, but the hardest problems are outsourced to the quantum chip. In finance, discrete portfolio optimisation problems, which are NP-hard, are such examples and clear objectives to tackle.

Why Quantum Machine Learning? It is a combination of quantum computing and AI that will likely generate the most exciting opportunities, including a whole range of possible applications in finance, but also in medicine, chemistry, physics, etc. We have already witnessed the first promising results achieved with Parameterised Quantum Circuits trained as either generative models (such as Quantum Circuit Born Machine, which can be used as a synthetic data generator) or discriminative

models (such as a Quantum Neural Network that can be trained as a classifier). The possible use cases include market generators, data anonymisers, credit scoring, and the generation of trading signals.

All the models and techniques mentioned so far rely on the existence of universal, gate model quantum computers. However, there is another type of quantum hardware – quantum annealers – which realise the principle of adiabatic quantum computing. Quantum annealers are analog quantum computers that are very well suited for solving complex optimisation problems that are NP-hard for classical computers. Optimisation problems form a large class of hard-to-solve financial problems, not to mention the fact that many supervised and reinforcement learning tools used in finance are trained via solving optimisation problems (minimisation of a cost function, maximisation of reward).

An example of discriminative machine learning problems solved using quantum annealers includes building a strong classifier from several weak ones – the Quantum Boosting algorithm. The strong classifier is highly resilient against overtraining and against errors in the correlations of the physical observables in the training data. The quantum annealing-trained classifiers perform comparably to the state-of-the-art classical machine learning methods. However, in contrast to these methods, the annealing-based classifiers are simple functions of directly interpretable experimental parameters with clear physical meaning and demonstrate some advantage over traditional machine learning methods for small training datasets.

Another application of quantum annealing is in generative learning. In Deep Learning, a well-known approach for training a

deep neural network starts with training a generative Deep Boltzmann Machine, typically using the Contrastive Divergence (CD) algorithm, then fine-tuning the weights using backpropagation or other discriminative techniques. However, generative training is often time consuming due to the slow mixing of Boltzmann (Gibbs) sampling. The quantum sampling-based training approach can achieve comparable or better accuracy with significantly fewer iterations of generative training than conventional CD-based training.

The main focus of this book is therefore on tackling practical real-world applications of Quantum Machine Learning (QML) algorithms executable on NISQ hardware rather than adopting the more traditional quantum computing textbook approach, diligently describing standard quantum computing algorithms (Shor's, Grover's, …), the quantum hardware demands of which are well beyond the capabilities of NISQ computers. The focus is also on the hybrid quantum-classical computational protocols that reflect the most productive way of harnessing the power of quantum computing – it is in tandem with classical computing that quantum computing solutions can provide maximum benefits to the users.

In this book, we cover all major QML algorithms that have been the subject of intensive research by the industry and that have shown early signs of potential quantum advantage. We also provide a balanced view of both analog and digital quantum computers and do not try to make a call on which quantum computing technology (superconducting qubits, trapped ions, neutral atoms, etc.) will be the eventual winner. The material is presented in a hardware-agnostic way with a strong emphasis on

the fundamental characteristics of the algorithms rather than their hardware realisations, although we do not ignore the question of algorithms' embedding and the practical limitations of the existing quantum computing hardware.

Why Finance? It is reasonable to expect that the incredibly fast rate of quantum hardware improvements we have witnessed over the last several years will lead to the widespread adoption of quantum computing techniques in finance. The finance industry is already investigating the potential of QML to solve classically hard practical problems and assist in achieving digital transformation. We might have moved past the point of quantum computing supremacy, but our quest to establish quantum computing advantage has just begun.

Quantitative finance is a discipline rich in interesting but computationally hard problems. Many such problems are interdisciplinary in nature and often require the transformation and adoption of mathematical and computational techniques developed in other fields. Here, we can mention, for example, the application of the theory of stochastic differential equations to option pricing [226], methods of optimal control theory to management science and economics [260], machine learning techniques to portfolio construction, and optimisation [193].

This is why we turn to finance when we are looking for a wide range of real-world use cases to test (and improve!) quantum computing algorithms. The book provides many examples of the quantum computing techniques and algorithms applied to solving practical financial problems such as portfolio optimisation, credit card default prediction, credit approvals, and

generation of synthetic market data. At the same time, the methods and techniques are formulated and presented in the most general form – we hope our readers will discover many new exciting quantum computing use cases in finance and beyond.

Who this book is for

The book is primarily aimed at three main groups: academic researchers and STEM students; finance professionals working in the field of quantitative finance and related areas; computer scientists and ML/AI experts. At the same time, the book is organised in such a way as to be accessible and useful to a much wider audience.

The book does not require any prior knowledge of quantum mechanics and the complexity of the mathematical apparatus should not feel intimidating: although we do not sacrifice mathematical rigour, the emphasis is very much on the understanding of the fundamental properties of the models and algorithms.

What this book covers

The book is split into two parts reflecting the natural progression from analog to digital quantum computing, with an increasing depth in the analysis and understanding of algorithms. However, we start with a chapter that covers the basic principles of quantum mechanics and provides the motivation for the computational methods based on those principles.

Chapter 1, The Principles of Quantum Mechanics, covers the basic mathematical principles of quantum mechanics. It provides the necessary definitions and discusses the postulates of quantum mechanics and their relevance to quantum computing.

Part I: Analog Quantum Computing – Quantum Annealing

For a number of years, quantum annealers were the only large-scale quantum computing devices available for experiments in solving non-trivial NP-hard combinatorial optimisation problems. Although quantum annealing specifically targets solving classically hard optimisation problems, it can also be used for many different hybrid quantum-classical problems, such as samplers and classifiers. The book provides detailed coverage of these applications and illustrates them on specific financial use cases.

Chapter 2, Adiabatic Quantum Computing, introduces the concept of analog quantum computing. The chapter starts with the principles of adiabatic quantum computing and proceeds with the quantum adiabatic theorem. The physical realisation of adiabatic quantum computing is quantum annealing, which is explained alongside its classical counterpart – simulated annealing. The chapter also discusses the implementation, limitations, and universality of adiabatic quantum computing.

Chapter 3, Quadratic Unconstrained Binary Optimisation, describes the single most important application of quantum annealing: solving classically hard optimisation problems. A wide range of combinatorial optimisation problems can be formulated as Quadratic Unconstrained Binary Optimisation (QUBO) problems (or, equivalently, as Ising problems) solvable on a quantum

annealer. The chapter provides in-depth coverage of the forward and reverse quantum annealing techniques and demonstrates the power of quantum annealing on a discrete portfolio optimisation use case.

Chapter 4, Quantum Boosting, extends the range of QUBO applications beyond combinatorial optimisation and outlines the Quantum Boosting algorithm designed to combine a large number of weak classical classifiers into a strong classifier. The algorithm is formulated as a QUBO problem executable on a quantum annealer and applied to the use case of building a strong predictor of credit card defaults from a large number of weak predictors.

Chapter 5, Quantum Boltzmann Machine, explores further machine learning applications of quantum annealing. The Quantum Boltzmann Machine can be used as a generative model for sampling from a learned probability distribution as well as an efficient method of pre-training deep feedforward neural networks.

Part II: Gate Model Quantum Computing

Gate model quantum computing hardware has demonstrated enormous progress in recent years and is quickly approaching the quantum advantage threshold. The search for quantum advantage – the real-world productive application of a quantum computing solution that outperforms any viable classical alternative – is one of the strongest motivations for quantum computing research in finance and elsewhere. The book explores the main quantum computing algorithms implementable on existing NISQ devices

and highlights a range of possible financial applications that may benefit from this new quantum computing paradigm.

Chapter 6, Qubits and Quantum Logic Gates, introduces the paradigm of gate model quantum computing. We start with the basic concepts of classical digital computing and expand the computational logic to accommodate the new principles of superposition and entanglement. The chapter draws parallels between and contrasts classical and quantum logic gates and shows how to assemble quantum circuits from individual quantum logic gates.

Chapter 7, Parameterised Quantum Circuits and Data Encoding, proceeds with the construction of quantum algorithms covering both the theoretical and the practical aspects of building Parameterised Quantum Circuits (PQCs), and demonstrates how classical samples can be encoded into quantum states processed by the PQCs. The chapter provides a detailed description of specific data encoding techniques.

Chapter 8, Quantum Neural Network, considers parameterised quantum circuits trained as classifiers. Throughout this chapter, we show how differentiable and non-differentiable learning algorithms can be used to efficiently train quantum neural networks. The chapter also discusses the limitations of existing QPUs and how to design quantum circuits that extract maximum benefit from the available quantum computing hardware. We investigate QNN performance on a credit approval use case and benchmark it against several standard classical classifiers.

Chapter 9, Quantum Circuit Born Machine, introduces a quantum counterpart to classical generative models such as Boltzmann

Machines – the Quantum Circuit Born Machine (QCBM). The chapter starts with the definition of QCBM and how it can be efficiently configured and run on available QPUs, continues with the differentiable and non-differentiable learning and training procedures, and concludes with the market generator use case benchmarked against classical Restricted Boltzmann Machine.

Chapter 10, Variational Quantum Eigensolver, introduces the variational principle and formulates the Variational Quantum Eigensolver (VQE) approach to optimisation problems. The chapter discusses a hybrid quantum-classical approach to training the VQE and looks at the practical aspects of running it on NISQ devices.

Chapter 11, Quantum Approximate Optimisation Algorithm, describes the gate model quantum computing approach (inspired by quantum annealing) to solving QUBO-type problems, such as NP-hard Max-Cut optimisation problems.

Chapter 12, The Power of Parameterised Quantum Circuits, investigates the main sources of quantum advantage we expect to demonstrate on practical applications of parameterised quantum circuits. The chapter focuses on two elements: strong regularisation provided by quantum neural networks and the expressive power of quantum generative models.

Chapter 13, Looking Ahead, discusses new promising quantum algorithms and techniques such as the quantum kernel method, quantum GAN, Bayesian quantum circuit, and quantum semi-definite programming.

To get the most out of this book

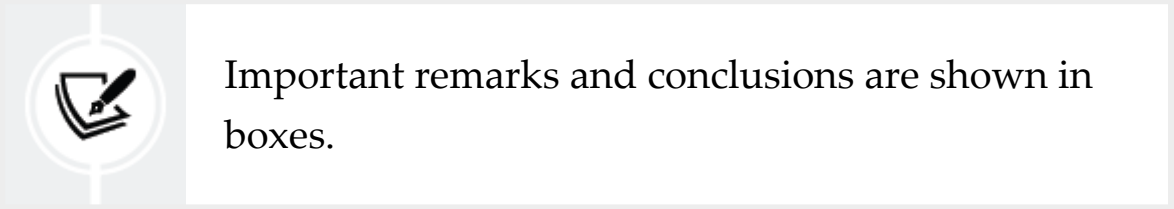
This book is intended as an in-depth introduction to the power of quantum computing techniques for quantitative finance problems. While it is designed as self-contained, this book assumes that the reader has some familiarity with basic mathematical concepts in algebra, analysis, and computing. Knowledge of quantum mechanics is not required, and the main tools thereof shall be explained and made accessible to non-physicists.

Conventions used

There are a number of text conventions used throughout this book.

`CodeInText`: indicates code words in the text, software packages, folder names, path names, etc.

When we wish to draw your attention to a particular definition or notation, the relevant lines or items are set in either *italic* or **bold**.



Get in touch

Feedback from our readers is always welcome.

General feedback: Email feedback@packtpub.com, and mention the book's title in the subject of your message. If you have

questions about any aspect of this book, please email us at questions@packtpub.com.

Errata: Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you have found a mistake in this book, we would be grateful if you could report this to us. Please visit http://www.packtpub.com/submit-errata, select your book, click on the Errata Submission Form link, and enter the details.

Piracy: If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at copyright@packtpub.com with a link to the material.

If you are interested in becoming an author: If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit http://authors.packtpub.com.

Share your thoughts

Once you've read *Quantum Machine Learning and Optimisation in Finance*, we'd love to hear your thoughts! Please click here to go straight to the Amazon review page for this book and share your feedback.

Your review is important to us and the tech community and helps us deliver excellent content.

Download a free PDF copy of this book

Thanks for purchasing this book!

Do you like to read on the go but are unable to carry your print books everywhere? Is your eBook purchase not compatible with the device of your choice?

Don’t worry, now with every Packt book you get a DRM-free PDF version of that book at no cost.

Read anywhere, any place, on any device. Search, copy, and paste code from your favorite technical books directly into your application.

The perks don’t stop there, you can get exclusive access to discounts, newsletters, and great free content in your inbox daily.

Follow these simple steps to get the benefits:

1. Scan the QR code or visit the link below:

   https://packt.link/free-ebook/9781801813570

2. Submit your proof of purchase
3. That’s it! We’ll send your free PDF and other benefits to your email directly

1
The Principles of Quantum Mechanics

Quantum mechanics is a framework for the development of physical theories; it is not itself a physical theory [80]. Actual physical theories are built upon a foundation of quantum mechanics. This is why quantum mechanics plays such an important role in all natural sciences. Information theory is no exception and also derives inspiration from the ideas and methods of quantum mechanics.

Understanding quantum computing requires some familiarity with the basic principles of quantum mechanics. This book does not assume any prior knowledge of quantum mechanics and provides all the necessary definitions and explanations when needed. At the same time, the reader is encouraged to learn more about this fascinating subject at the level of mathematical formalism that she is comfortable with. Out of the extensive universe of textbooks on quantum mechanics that provide an introduction to this discipline, it is necessary to mention the classical book by Landau and Lifshitz [182] as well as the equally classical book on quantum computing by Nielsen and Chuang [223], which covers the most relevant aspects of quantum mechanics from the quantum computing perspective. For someone taking their first steps in quantum computing who would like to get the overall picture and some historical perspective, the excellent book by Bernhardt [32] provides both without the heavy usage of complex mathematical apparatus. Readers looking for a more formal modern take on the subject of quantum mechanics

may find it in the book by Robinett [249]. The practical aspects of quantum computing are covered in great detail in the book by Sutor [278], and anyone looking for a python quantum computing programming textbook will find it in the work by Loredo [195].

1.1 Linear Algebra for Quantum Mechanics

Quantum computing and quantum mechanics rely on a specific notational formalism, due to Dirac, and are supported by classical linear algebra, in particular Hermitian structures of matrices and tensor products. We provide here a self-contained review of these tools to facilitate the understanding of the rest of the book. We start with basic linear algebra principles before introducing Dirac notations and the quantum counterparts of linear algebra tools. Sections 1.1.1 to 1.1.4 concentrate on standard definitions of finite-dimensional Hilbert spaces and matrices, while Sections 1.1.5 to 1.1.7 review the key details and properties of complex matrices (decompositions, Hermitian property, and rotations). Sections 1.1.9 to 1.1.11 introduce Dirac's formalism and the essential aspects of quantum operators.

1.1.1 Basic definitions and notations

We let $\mathbb{F}$ denote either the real field $\mathbb{R}$ or the complex one $\mathbb{C}$. For a complex number $z = x+\mathrm{i}y \in \mathbb{C}$, with $x,y \in \mathbb{R}$, we write the conjugate $z^* := x-\mathrm{i}y$. We let $\mathcal{M}_{m,n}(\mathbb{F})$ denote the space of matrices of dimension $m \times n$ with entries in $\mathbb{F}$ and $\mathcal{M}_n(\mathbb{F})$ whenever $m = n$. For $A := (a_{ij})_{1\le i\le m;\ 1\le j\le n} \in \mathcal{M}_{m,n}(\mathbb{F})$, $A^* := (a_{ij}{}^*)_{1\le i\le m;\ 1\le j\le n}$ is the complex conjugate. If $A \in \mathcal{M}_n(\mathbb{F})$, we write $A^\top$ for its transpose and $A^\dagger :=$

$(A^*)^\top$ for its *Hermitian conjugate*. We finally denote I the identity matrix and write I_n whenever we wish to emphasise the dimension, and $0_{m,n}$ the null matrix in $\mathcal{M}_{m,n}(\mathbb{F})$. Recall that a matrix $A \in \mathcal{M}_n(\mathbb{F})$ is invertible (or non-singular) if there exists $B \in \mathcal{M}_n(\mathbb{F})$ such that $AB = BA = I_n$. Given two matrices $A \in \mathcal{M}_{p,m}(\mathbb{F})$ and $B \in \mathcal{M}_{q,n}(\mathbb{F})$, we define their tensor product as

$$A \otimes B := \begin{bmatrix} a_{11}B & \dots & a_{1m}B \\ \vdots & \ddots & \vdots \\ a_{p1}B & \dots & a_{pm}B \end{bmatrix} \in \mathcal{M}_{pq,mn}(\mathbb{F}).$$

Since a vector is a particular case of a matrix, for $u \in \mathbb{F}^m$ and $v \in \mathbb{F}^n$, we can write

$$u \otimes v = \begin{bmatrix} u_1 \\ \vdots \\ u_m \end{bmatrix} \otimes \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} u_1 v_1 \\ \vdots \\ u_1 v_n \\ u_2 v_1 \\ \vdots \\ u_m v_n \end{bmatrix} \in \mathbb{F}^{mn}.$$

1.1.2 Inner products

A vector space V over the field $\mathbb{F}$ is a set endowed with

- a commutative, associative addition operation,
- an operation of multiplication by a scalar.

The addition and the multiplication by a scalar have the following properties (for scalars $\alpha, \beta \in \mathbb{F}$ and vectors $u, v \in V$):

- $v + 0 = v$;
- $v + (-v) = 0$;
- $\alpha(\beta v) = (\alpha\beta)v$;
- $(\alpha + \beta)v = \alpha v + \beta v$;
- $\alpha(u + v) = \alpha u + \alpha v$;
- $1 \cdot v = v$.

Armed with this, we can now define an inner product on V:

Definition 1. A map $\langle \cdot,\cdot \rangle : V \times V \to \mathbb{F}$ is called an *inner product* if, for $u,v,w \in V$ and $\alpha \in \mathbb{F}$,

- *(Positive definiteness)* $\langle u,u \rangle \geq 0$ and $\langle u,u \rangle = 0$ if and only if $u = 0$;
- *(Conjugate symmetry)* $\langle u,v \rangle = \langle v,u \rangle^*$;
- *(Linear in the first argument)* $\langle u + v,w \rangle = \langle u,w \rangle + \langle v,w \rangle$ and $\langle \alpha u,v \rangle = \alpha \langle u,v \rangle$;
- *(Antilinear in the second argument)* $\langle u,v + w \rangle = \langle u,v \rangle + \langle u,w \rangle$ and $\langle u,\alpha v \rangle = \alpha^* \langle u,v \rangle$.

The inner product is further called non-degenerate if $\langle u,v \rangle = 0$ for all $v \in V \setminus \{0\}$ implies $u = 0$.

For example, the following spaces carry a natural inner product:

- The vector space $\mathbb{C}^n$ with the inner product $\langle u,v \rangle := u^\dagger v = \sum_{i=1}^{n} u_i^* v_i$;
- The space of complex-valued continuous functions on $[0,1]$ with $\langle f,g \rangle := \int_0^1 f(t)^* g(t) \mathrm{d}t$;
- If $X,Y \in \mathcal{M}_{m,n}(\mathbb{R})$, then $\langle X,Y \rangle := \mathrm{Tr}(X^\mathsf{T} Y) = \sum_{i=1}^{m} \sum_{j=1}^{n} X_{ij} Y_{ij}$ defines an inner product on the space of (real) matrices.

Projection matrices are particularly useful for geometric purposes:

Definition 2. A matrix $P \in \mathcal{M}_n(\mathbb{F})$ is called a (orthogonal) *projection* if $P^2 = P$.

In particular, if W is a vector subspace of $\mathbb{F}^n$ with some orthonormal basis $(w_1,\ldots,w_d)$, it is then easy to check that the map $\mathcal{P}_W : \mathbb{F}^n \to \mathbb{F}^n$ onto W satisfying

$$\mathcal{P}_{\mathbf{W}}(v) := \sum_{i=1}^{d} \langle v, w_i \rangle w_i, \quad \text{for any } v \in \mathbb{F}^n,$$

defines an orthogonal projection.

1.1.3 From linear operators to matrices

Let V be a finite-dimensional vector space over $\mathbb{F}$ and $\langle \cdot,\cdot \rangle$ a non-degenerate inner product on V. Given a linear operator $\mathcal{A} : V \to V$, then, by the Riesz representation theorem [309, Section III-6], there exists a unique linear operator $\mathcal{A}^\dagger : V \to V$, called the adjoint operator, such that

$$\langle \mathcal{A}u, v \rangle = \langle u, \mathcal{A}^\dagger v \rangle, \quad \text{for all } u, v \in \mathbf{V}.$$

Indeed, for any $v \in V$, the map $u \in V \mapsto \langle \mathcal{A}u, v \rangle$ is a linear functional, hence an element of the dual space $V^\dagger$ (the space of bounded linear functionals on V), therefore for each $v \in V$, there exists $v' \in V$ such that $\langle \mathcal{A}u, v \rangle = \langle u, v' \rangle$. It is then easy to show that the map $v \mapsto v'$ is linear, proving that the adjoint operator is uniquely defined. In the particular case where $\mathcal{A} = \mathcal{A}^\dagger$, the operator $\mathcal{A}$ is called Hermitian, a key requirement in quantum mechanics:

Definition 3. The operator $\mathcal{A}$ is called *Hermitian*, or *self-adjoint*, if $\mathcal{A} = \mathcal{A}^\dagger$.

For a Hermitian operator $\mathcal{A}$, we then have, for any u ∈V,

$$\langle \mathcal{A}u, u\rangle = \langle u, \mathcal{A}^\dagger u\rangle = \langle u, \mathcal{A}u\rangle = \langle \mathcal{A}u, u\rangle^*$$

by conjugate symmetry (Definition 1), and therefore ⟨$\mathcal{A}$u,u⟩ is real. Conversely, if ⟨$\mathcal{A}$u,u⟩ is real, then

$$\langle \mathcal{A}u, u\rangle = \langle \mathcal{A}u, u\rangle^* = \langle u, \mathcal{A}u\rangle = \langle \mathcal{A}^\dagger u, u\rangle.$$

Therefore, $\langle (\mathcal{A} - \mathcal{A}^\dagger)\, u, u\rangle = 0$; since this is true for all u ∈V, then $\mathcal{A} = \mathcal{A}^\dagger$.

The following property of operators shall be useful to ensure that systems driven by operators preserve distances, or norms:

Definition 4. The linear operator $\mathcal{A} : V \to V$ is called *unitary* if it is surjective and

$$\langle \mathcal{A}u, \mathcal{A}v\rangle = \langle u, v\rangle, \quad \text{for all } u, v \in \mathbf{V}.$$

Recall that a linear operator between two finite-dimensional normed spaces is bounded, and therefore continuous. For any u ∈V, this implies that ‖$\mathcal{A}$u‖ = ‖u‖, so that a unitary operator $\mathcal{A}$ preserves the norm. In that case, $\mathcal{A}$ is an isometry, therefore injective. Being also surjective, it is bijective and therefore its inverse exists. For a unitary operator $\mathcal{A}$ and any u,v ∈V, we have

$$\langle u, v\rangle = \langle \mathcal{A}u, \mathcal{A}v\rangle = \langle u, \mathcal{A}^\dagger \mathcal{A}v\rangle$$

by definition of the adjoint, implying that

$$\mathcal{A}^\dagger \mathcal{A} = \mathcal{I} = \mathcal{A}\mathcal{A}^\dagger,$$

where $\mathcal{I}$ is the identity operator.

Example (Real Matrices): If $V = \mathbb{R}^n$ with inner product $\langle u,v\rangle := u^\top v$ for $u,v \in \mathbb{R}^n$, the linear operator $\mathcal{A}$ can now be viewed as a matrix A in $\mathcal{M}_n(\mathbb{R})$. Its adjoint is nothing other than the transpose $A^\top$, and therefore A is self-adjoint if and only if it is symmetric. In this case, if A is unitary (or orthogonal), then it is invertible with $A^{-1} = A^\top$. Rotation matrices in $\mathbb{R}^2$, which will play an important role later when constructing quantum circuits, are the only unitary maps of $\mathbb{R}^2$ onto itself and are of the form

$$\begin{bmatrix} \cos(\theta) & \delta \sin(\theta) \\ \sin(\theta) & -\delta \cos(\theta) \end{bmatrix},$$

for $\theta \in [0,2\pi)$ and $\delta \in \{-1,+1\}$.

Example (Complex Matrices): If $V = \mathbb{C}^n$ with inner product $\langle u,v\rangle := v^\dagger u$ for $u,v \in \mathbb{C}^n$, the linear operator $\mathcal{A}$ can now be viewed as a matrix in $\mathcal{M}_n(\mathbb{C})$. The adjoint of such a matrix is then the Hermitian conjugate $A^\dagger$ and A is called Hermitian if $A = A^\dagger$ and unitary if $A^\dagger A = I_n$. We shall denote by $\mathcal{U}_n(\mathbb{C})$ the set of unitary matrices in $\mathcal{M}_n(\mathbb{C})$. We will discuss Hermitian matrices over $\mathbb{C}$ in more detail in Section 1.1.6.

1.1.4 Condition number

In order to manipulate matrices and measure them, we require matrix norms:

Definition 5. A matrix norm $\|\cdot\| : \mathcal{M}_{m,n}(\mathbb{F}) \to \mathbb{R}$ is a function satisfying, for any $\alpha \in \mathbb{F}$ and $A,B \in \mathcal{M}_{m,n}(\mathbb{F})$,

- (positively valued) $\|A\| \geq 0$;

- (definite) $\|A\| = 0$ if and only if $A = 0_{m,n}$;
- (absolutely homogeneous) $\|\alpha A\| = |\alpha| \|A\|$;
- (triangle inequality) $\|A + B\| \leq \|A\| + \|B\|$.

The norm is further called sub-multiplicative if $\|AB\| \leq \|A\| \|B\|$.

The condition number of a matrix is an important tool to understand the stability of linear equations of the form $Ax = b$, for $A \in \mathcal{M}_n(\mathbb{F})$, $b \in \mathbb{F}^n$. Assuming A to be non-singular, the true solution is clearly $x_* := A^{-1}b$. Suppose, however, that the vector b is only known up to some (not necessarily quantum) measurement error, and one observes instead $b + \Delta_b$. The solution is then $A^{-1}(b + \Delta_b) = x_* + \Delta_x$, with $\Delta_x := A^{-1}\Delta_b$. In particular, we can write, for any (sub-multiplicative) matrix norm $\|\cdot\|$,

$$\frac{\|\Delta_x\|}{\|x\|} = \frac{\|A^{-1}\Delta_b\|}{\|A^{-1}b\|} \leq \|A^{-1}\| \frac{\|b\|}{\|A^{-1}b\|} \frac{\|\Delta_b\|}{\|b\|} \leq \|A^{-1}\| \|A\| \frac{\|\Delta_b\|}{\|b\|}.$$

From this inequality, we see that the quantity $\|A^{-1}\|\|A\|$ bounds the relative error in the solution with respect to the relative error in the measurement of the input vector b. This leads to the following terminology:

Definition 6. Given a matrix $A \in \mathcal{M}_n(\mathbb{F})$ and a sub-multiplicative norm $\|\cdot\|$, we call

$$\kappa_{\|\cdot\|}(A) := \|A^{-1}\| \|A\|$$

the *condition number* (with respect to the norm $\|\cdot\|$) of the matrix A (and assign to it infinite value if A is singular).

Remark: The definition of the condition number above holds for any matrix norm $\|\cdot\|$, but admits a more explicit representation in

the particular case of the spectral norm $\|\cdot\|_2$, defined as

$$\|\mathrm{A}\|_2 := \sup_{\mathrm{x}\neq 0} \frac{\|\mathrm{Ax}\|_2}{\|\mathrm{x}\|_2},$$

where $\|\mathrm{x}\|_2 := \left(\sum_{i=1}^{n} |x_i|^2\right)^{\frac{1}{2}}$ is the L_2 norm for vectors. If the matrix A is not singular, then

$$\kappa(\mathrm{A}) := \frac{|\lambda_{\max}(\mathrm{A})|}{|\lambda_{\min}(\mathrm{A})|},$$

where $\lambda_{\max}(\mathrm{A})$ and $\lambda_{\min}(\mathrm{A})$ denote the largest and smallest eigenvalues of A.

1.1.5 Matrix decompositions and spectral theorem

Having defined essential properties of (complex) matrices, we now introduce several essential tools that allow us to gain a better understanding of their properties.

The Singular Value Decomposition is a key tool to analyse the properties and behaviours of matrices. It is ubiquitous in applied statistics and machine learning and allows us to reduce the explanatory dimension of a large matrix into a small number of meaningful components.

Theorem 1 (Singular Value Decomposition)**.** *Let* $\mathrm{A} \in \mathcal{M}_{m,n}(\mathbb{F})$ *and* $p := \min(m,n)$. *There exist* $\mathrm{U} \in \mathcal{U}_m(\mathbb{F})$, $\mathrm{V} \in \mathcal{U}_n(\mathbb{F})$ *and* $\sigma_1 \geq \cdots \geq \sigma_p \geq 0$ *such that* $\mathrm{A} = \mathrm{U}\Sigma\mathrm{V}^{\dagger}$, *where* $\Sigma \in \mathcal{M}_{m,n}(\mathbb{F})$ *is diagonal with* $\Sigma_{ii} = \sigma_i$ *for* $i = 1, \ldots, p$ *and* $\Sigma_{ii} = 0$ *for* $i > p$.

The numbers $\{\sigma_1,\ldots,\sigma_p\}$ are called the singular values of A and are uniquely defined. The columns of U and V are the left-singular and right-singular vectors of A, in the sense that, if $\sigma \in \{\sigma_1,\ldots,\sigma_p\}$, then there exist a column u of U and a column v of V such that $Av = \sigma u$ and $A^\dagger u = \sigma v$. Recall that the rank of a matrix is defined as the dimension of the span of its columns. As a corollary of the Singular Value Decomposition theorem, the rank of a matrix is therefore equal to the number of non-zero singular values. The Singular Value Decomposition is general in the sense that it holds for any matrix. In the particular case of square matrices, the Schur decomposition and the Spectral Theorem provide refinements.

The Spectral Theorem is a cornerstone result in the theory of linear operators, and in particular for (finite-dimensional) matrices. Recall that an operator $\mathcal{A} : V \to V$ is called normal if it commutes with its adjoint, namely if $\mathcal{A}\mathcal{A}^\dagger = \mathcal{A}^\dagger\mathcal{A}$. Self-adjoint (or Hermitian) operators are clearly normal, yet the converse is not true in general. Recall further that an eigenvector of $\mathcal{A}$ is a non-zero vector $u \in V$ such that $\mathcal{A}u = \lambda u$ for some $\lambda \in \mathbb{C}$, and we denote by $\sigma(\mathcal{A})$ the set of eigenvalues of $\mathcal{A}$.

The following result, which is more general than the subsequent spectral theorem, allows us to decompose any arbitrary complex square matrix.

Theorem 2 (Schur Decomposition)**.** *For any* $A \in \mathcal{M}_n(\mathbb{C})$ *there exits a unitary matrix* $U \in \mathcal{U}_n(\mathbb{C})$ *and an upper triangular matrix* T *such that* $A = UTU^{-1}$.

Note that since U is unitary, then $U^{-1} = U^\dagger$. We call the matrix T the Schur transform of A and the identity in the theorem means that A and T are similar, so in particular, possess the same eigenvalues,

all located on the diagonal of T. If A is a normal matrix, then so is T, and therefore T must be diagonal and we write T = D for clarity. In this case, we say that the matrix A is diagonalisable with $A = UDU^\dagger$, where the diagonal entries of D are the eigenvalues of A and the column vectors of U are the orthonormal eigenvectors of A.

Theorem 3 (Spectral Theorem)**.** *The linear operator* $\mathcal{A} : V \to V$ *is normal if and only if there exists an orthonormal basis of* V *consisting of eigenvectors of* A.

For each eigenvalue $\lambda \in \sigma(\mathcal{A})$, denote the corresponding eigenspace

$$\mathcal{V}_\lambda := \{u \in \mathbf{V} : \mathcal{A}u = \lambda u\}.$$

Since the vector space V is the orthogonal direct sum of the eigenspaces (indexed by the eigenvalues of $\mathcal{A}$), we can then write the spectral decomposition

$$\mathcal{A} = \sum_{\lambda \in \sigma(\mathcal{A})} \lambda \mathcal{P}_\lambda,$$

where $\mathcal{P}_\lambda$ is the orthogonal projection operator onto $\mathcal{V}_\lambda$. Note that such an operator is naturally self-adjoint [309, Theorem 2, Section III-1].

1.1.6 Hermitian matrices

We introduced above Hermitian matrices as the set of matrices A over the complex field $\mathbb{C}$ such that $A = A^\dagger$. As fundamental building blocks of quantum computing, we investigate their

properties further. Clearly, a real matrix is Hermitian if and only if it is symmetric, in which case $A^\top = A$.

Proposition 1. *The eigenvalues of a Hermitian matrix are real.*

Proof. If $Ax = \lambda x$ for $\lambda \in \mathbb{C}$ and $x \in \mathbb{C}^n$, then

$$\begin{aligned} \langle Ax, x \rangle &= x^\dagger Ax = \lambda x^\dagger x = \lambda \|x\|^2, \\ \langle x, Ax \rangle &= (Ax)^\dagger x = (\lambda x)^\dagger x = \lambda^* x^\dagger x = \lambda^* \|x\|^2. \end{aligned}$$

Since both are equal by the Hermitian property, then $\lambda = \lambda^*$, proving the proposition. □

The Singular Value Decomposition (Theorem 1) takes a particular flavour in the case of Hermitian matrices:

Theorem 4. *With the notations of Theorem 1, if* $A \in \mathcal{M}_n(\mathbb{C})$ *is Hermitian, then the matrices* U *and* V *are equal and the matrix* Σ *is diagonal with real entries.*

Theorem 5. *For a Hermitian matrix* $A \in \mathcal{M}_n(\mathbb{C})$*, the following are equivalent:*

1. *The eigenvalues are non-negative.*
2. *There exists a Hermitian matrix* $B \in \mathcal{M}_n(\mathbb{C})$ *such that* $A = B^2$.
3. *There exists a matrix* $B \in \mathcal{M}_n(\mathbb{C})$ *such that* $A = B^\dagger B$.
4. *For every* $x \in \mathbb{C}^n$, $\langle Ax, x \rangle \geq 0$.

Such a matrix is called positive semi-definite.

Proof. The Spectral Theorem shows that there exist a unitary matrix $U \in \mathcal{U}_n(\mathbb{C})$ and a diagonal matrix $\Sigma \in \mathcal{M}_n(\mathbb{C})$ such that $A = U\Sigma U^\dagger$,

where the diagonal elements of Σ are the eigenvalues of A. Assuming (i), we can define $B = U\sqrt{\Sigma}U^{\dagger} \in \mathcal{M}_n(\mathbb{C})$. Then clearly

$$B^{\dagger} = B \quad \text{and} \quad B^2 = \left(U\sqrt{\Sigma}U^{\dagger}\right)\left(U\sqrt{\Sigma}U^{\dagger}\right) = A,$$

since U is unitary. The equality $A = B^{\dagger}B$ is also obvious. The latter implies that

$$\langle Ax, x\rangle = \langle B^{\dagger}Bx, x\rangle = \langle Bx, Bx\rangle = \|Bx\|^2 \geq 0, \quad \text{for any } x \in \mathbb{C}^n.$$

Finally, assume (iv) and let λ be an eigenvalue of A with eigenvector u. Then

$$\langle Au, u\rangle = \langle \lambda u, u\rangle = \lambda\langle u, u\rangle = \lambda\|u\|^2.$$

Since the latter is strictly positive, then clearly $\lambda \geq 0$.

The following property lies at the core of Hamiltonian simulation of quantum systems:

Theorem 6. *If* $A \in \mathcal{M}_n(\mathbb{C})$ *is Hermitian, then, for any* $t \in \mathbb{R}$, e^{itA} *is unitary; conversely, every unitary matrix has the form* e^{itA} *for some Hermitian matrix* A.

Recall that for a matrix $A \in \mathcal{M}_n(\mathbb{C})$, its exponential is given by

$$e^{A} = \sum_{k \geq 0} \frac{A^k}{k!}.$$

In practice, though, given a Hermitian matrix A, finding the corresponding unitary matrix U is not easy. The Hamiltonian simulation problem is defined as follows.

Hamiltonian Problem: Given a Hermitian matrix $A \in \mathcal{M}_n(\mathbb{C})$, a time $t > 0$, a tolerance level $\varepsilon > 0$, and some matrix norm $\|\cdot\|$, find a unitary matrix U such that $\left\| U - e^{itA} \right\| \leq \varepsilon$.

1.1.7 Rotation matrices

Rotation matrices, and later their quantum gate equivalents will play a key role in building quantum circuits. Let us start with the following lemma:

Lemma 1. *If a matrix* $A \in \mathcal{M}_n(\mathbb{C})$ *is such that* $A^2 = I$, *then for any* $\theta \in \mathbb{R}$,

$$e^{i\theta A} = \cos(\theta)\mathbf{I} + i\sin(\theta)A.$$

Proof. This follows directly from the series expansion

$$e^x = \sum_{k \geq 0} \frac{x^k}{k!},$$

which has an infinite radius of convergence. □

Lemma 1 will prove essential for computational purposes. As simple examples, consider the following:

Exercise: Compute $e^{i\theta A}$ for $A \in \{X, Y, Z\}$ and $\theta \in \mathbb{R}$, where

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

For any $\alpha \in [0, 2\pi)$, consider now the map $\mathcal{R}_\alpha : \mathbb{R}^2 \to \mathbb{R}^2$ such that

$$\mathcal{R}_\alpha\Big(r\cos(\theta), r\sin(\theta)\Big) := \Big(r\cos(\theta + \alpha), r\sin(\theta + \alpha)\Big),$$

$$\text{for any } r \in \mathbb{R} \text{ and } \theta \in [0, 2\pi),$$

which is basically a rotation of angle α and does not affect the norm of the input vector. To the map $\mathcal{R}_\alpha$, we can associate the (rotation) matrix R_α such that $\mathcal{R}_\alpha(u) = R_\alpha u$ for any $u \in \mathbb{R}^2$. It is easy (exercise) to show the following:

Lemma 2. *The matrix* R_α *has the form*

$$R_\alpha = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix}.$$

This representation is the general form of a rotation matrix in $\mathbb{R}^2$ (introduced in (1.1.3)).

Exercise: Write the matrices $e^{i\theta A}$ for $A \in \{X,Y,Z\}$ from the previous exercise as rotation matrices.

1.1.8 Polar coordinates

Recall that a point $z = x + iy$, with $x,y \in \mathbb{R}$, lying on the unit circle can be written as $z = e^{i\theta}$ with $\theta \in [0,2\pi)$. Indeed, simply let $x = r\cos(\theta)$, $y = r\sin(\theta)$ and add the constraint $r = 1$. Consider now a general vector $u \in \mathbb{C}^2$ of the form

$$u = \alpha e_1 + \beta e_2,$$

with $\alpha,\beta \in \mathbb{C}$ such that $|\alpha|^2 + |\beta|^2 = 1$. Here, (e_1,e_2) forms a basis of $\mathbb{R}^2$:

$$e_1 := \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad e_2 := \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

In polar coordinates, we can then write

$$\mathrm{u} = r_\alpha \mathrm{e}^{\mathrm{i}\theta_\alpha}\mathrm{e}_1 + r_\beta \mathrm{e}^{\mathrm{i}\theta_\beta}\mathrm{e}_2.$$

Note that arbitrary multiplication phases have no influence – a fact of key importance in quantum mechanics – because, for any $\gamma \in \mathbb{R}$,

$$|\mathrm{e}^{\mathrm{i}\gamma}\alpha|^2 = (\mathrm{e}^{\mathrm{i}\gamma}\alpha)^*\mathrm{e}^{\mathrm{i}\gamma}\alpha = \alpha^*\mathrm{e}^{-\mathrm{i}\gamma}\mathrm{e}^{\mathrm{i}\gamma}\alpha = \alpha^*\alpha = |\alpha|^2,$$

so that in fact, multiplying u by the global phase $\mathrm{e}^{-\mathrm{i}\theta_\alpha}$ and letting $\theta := \theta_\beta - \theta_\alpha$, we consider

$$\mathrm{u} = r_\alpha \mathrm{e}_1 + r_\beta \mathrm{e}^{\mathrm{i}\theta}\mathrm{e}_2.$$

Write temporarily $r_\beta \mathrm{e}^{\mathrm{i}\theta} = x + \mathrm{i}y$. Insisting on u being on the unit sphere further imposes $\|\mathrm{u}\|^2 = 1$, namely

$$\begin{aligned} 1 &= \|\mathrm{u}\|^2 = \left(r_\alpha \mathrm{e}_1 + (x + \mathrm{i}y)\mathrm{e}_2\right)^\dagger \left(r_\alpha \mathrm{e}_1 + (x + \mathrm{i}y)\mathrm{e}_2\right) \\ &= \left(r_\alpha \mathrm{e}_1{}^\top + (x - \mathrm{i}y)\mathrm{e}_2{}^\top\right)\left(r_\alpha \mathrm{e}_1 + (x + \mathrm{i}y)\mathrm{e}_2\right) \\ &= r_\alpha{}^2 + x^2 + y^2, \end{aligned}$$

since $(\mathrm{e}_1, \mathrm{e}_2)$ is orthonormal. This is nothing more than the equation of the unit sphere. In polar coordinates, we can write

$$x = r\sin(\theta)\cos(\phi), \quad y = r\sin(\theta)\sin(\phi), \quad r_\alpha = r\cos(\theta),$$

and clearly $r = 1$ since we are on the unit sphere. Therefore

$$\begin{aligned} \mathrm{u} &= \cos(\theta)\mathrm{e}_1 + \left(\sin(\theta)\cos(\varphi) + \mathrm{i}\sin(\theta)\sin(\varphi)\right)\mathrm{e}_2 \\ &= \cos(\theta)\mathrm{e}_1 + \sin(\theta)\mathrm{e}^{\mathrm{i}\varphi}\mathrm{e}_2. \end{aligned}$$

1.1.9 Dirac notations

Given a vector v $\in \mathbb{C}^n$, Dirac's *ket* and *bra* notations read

$$|\mathrm{v}\rangle := \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \quad \text{and} \quad \langle \mathrm{v}| := [v_1^*, v_2^*, \ldots, v_n^*].$$

With these notations, the operation $\langle \mathrm{u}, \mathrm{v} \rangle := \langle \mathrm{u} | \mathrm{v} \rangle$ defines an inner product on $\mathbb{C}^n$. The notation for the standard orthonormal basis in $\mathbb{C}^n$ is $(|i\rangle)_{i=0,\ldots,n-1}$, i.e.,

$$|0\rangle := \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad |1\rangle := \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \quad \ldots \quad |n-1\rangle := \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}.$$

In coordinates, we can write, for any $u,v \in \mathbb{C}^n$,

$$|\mathrm{u}\rangle = \sum_i u_i \, |i\rangle \quad \text{and} \quad |\mathrm{v}\rangle = \sum_i v_i \, |i\rangle \,,$$

and therefore,

$$\langle \mathrm{u}, \mathrm{v} \rangle = \sum_i u_i^* v_i.$$

1.1.10 Quantum operators

In the language of Dirac's notations, we can define the *outer product* $|\mathrm{u}\rangle\langle \mathrm{v}|$ (for u $\in$U and v $\in$V) as a linear operator from V to U, two

vector spaces, as

$$\Big(|\mathrm{u}\rangle \langle \mathrm{v}| \Big) |\mathrm{w}\rangle := \langle \mathrm{v}|\mathrm{w}\rangle |\mathrm{u}\rangle , \quad \text{for any } \mathrm{w} \in \mathbf{V}.$$

In particular, $|\mathrm{v}\rangle\langle \mathrm{v}|$ is the projection on the one-dimensional space generated by $|\mathrm{v}\rangle$. Any linear operator can be expressed as a linear combination of outer products as

$$\mathcal{A} = \sum_{ij} A_{ij} |i\rangle \langle j| ,$$

where $|i\rangle$ and $|j\rangle$ are the standard basis vectors (1.1.9).

Similarly to the linear algebra setting above, we can define an eigenvector of a linear operator $\mathcal{A} : \mathrm{V} \to \mathrm{V}$ as a non-zero vector $|\mathrm{v}\rangle$ such that

$$\mathcal{A} |\mathrm{v}\rangle = \lambda |\mathrm{v}\rangle$$

for some complex eigenvalue λ. Associated with any linear operator $\mathcal{A}$, the adjoint operator $\mathcal{A}^\dagger$ satisfies

$$\langle \mathrm{u}|\mathcal{A}\mathrm{v}\rangle = \left\langle \mathcal{A}^\dagger \mathrm{u}|\mathrm{v}\right\rangle .$$

Indeed, in the language of linear operators above, we have

$$\langle \mathrm{u}, \mathcal{A}\mathrm{v}\rangle = \langle \mathcal{A}\mathrm{v}, \mathrm{u}\rangle^* = \langle \mathrm{v}, \mathcal{A}^\dagger \mathrm{u}\rangle^* = \langle \mathcal{A}^\dagger \mathrm{u}, \mathrm{v}\rangle,$$

by definition of the inner product (Definition 1).

1.1.11 Tensor product

Given two vector spaces U and V of dimensions m and n, the tensor product U $\otimes$V is a vector space of dimension mn. For u $\in$U and v $\in$V, we can form the vector $|uv\rangle := |u\rangle \otimes |v\rangle \in U \otimes V$ with the following properties:

- $|(u + u')v\rangle = |uv\rangle + |u'v\rangle$, for any u′$\in$U;
- $|u(v + v')\rangle = |uv\rangle + |uv'\rangle$, for any v′$\in$V;
- $\alpha|uv\rangle = |(\alpha u)v\rangle = |u(\alpha v)\rangle$, for any $\alpha \in \mathbb{C}$.

Given the linear operators $\mathcal{A} : U \to U$ and $\mathcal{B} : V \to V$, we can then define their tensor product as an operator $\mathcal{A} \otimes \mathcal{B}$ on U $\otimes$V:

$$\left(\mathcal{A} \otimes \mathcal{B}\right) |uv\rangle := |(\mathcal{A}u), (\mathcal{B}v)\rangle ,$$

which can be represented in matrix form as A $\otimes$ B $\in \mathcal{M}_{mn,mn}(\mathbb{C})$.

This Dirac formalism, fully anchored in (classical) linear algebra, now opens the gates to a proper dive into the foundations of quantum mechanics.

1.2 Postulates of Quantum Mechanics

Quantum mechanics states several mathematical postulates that a physical theory must satisfy. It turns out that the mathematics of quantum mechanics allows for more general *computation*: more general definition of the *memory state* in comparison with classical digital computing and a wider range of possible *transformations* of such memory states. A natural question arises: what is the reason for this superior mode of computation not being used until very recently? The answer is that although quantum mechanics was formulated almost a century ago (Paul Dirac's seminal work "The

Principles of Quantum Mechanics" [86] was published in 1930), the realisation of the rules of quantum mechanics in the computational protocol performed on classical digital computers requires an enormous amount of memory. Exponential gains in computing power are offset by exponential memory requirements.

In order to perform quantum computations efficiently, we need to use actual quantum mechanical systems, with their ability to encode information in their states. To illustrate this point, the state of a quantum system consisting of n quantum bits (qubits) can be described by specifying 2^n probability amplitudes – this is a huge amount of information even for very small systems ($n \sim 100$) and it would be impossible to store this information in classical memory. It took decades of technological progress before quantum processing units (QPUs) – devices that control quantum mechanical systems performing computations – became feasible.

Let us now proceed with the formulation of the mathematical postulates that lie at the foundation of quantum mechanics. These postulates specify a general framework for describing the behaviour of a physical system [80, 182, 249]:

1. How to describe the state of a closed system.
2. How to describe the evolution of a closed system.
3. How to describe the interactions of a system with external systems.
4. How to describe observables of a system.
5. How to describe the state of a composite system in terms of its component parts.

1.2.1 First postulate - Statics

Postulate 1. *Associated to any physical system is a complex inner product space known as the state space of the system. The system is completely described at any given point in time by its state vector, which is a unit vector in its state space.*

What is the importance of the first postulate from the quantum computing point of view? The answer is that quantum mechanics offers us a straightforward generalisation of the classical *binary digit* (bit). The classical bit is a two-state system with controlled transitions between them. As an example, we can use an electrical switch that can exist in one of the two discrete, stable states ("on" and "off"). Although electrical switches may seem an odd physical realisation of bits in the age of transistors, they illustrate an important point about computation in general: it is substrate independent. Exactly the same computational results can be obtained using electrical relays and CMOS transistors.

The quantum mechanical version of a bit, called a *quantum binary digit* (qubit), is a quantum mechanical two-state system. The first postulate of quantum mechanics tells us that the state of such a system can be represented mathematically by a unit vector in the two-dimensional complex vector space. This also means that such a system can exist in a superposition of basis states. Indeed, any vector $|\mathrm{v}\rangle$ in the two-dimensional complex vector space,

$$|\mathrm{v}\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix},$$

can be represented as a linear combination of the standard basis vectors:

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \alpha \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \beta \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad |\mathrm{v}\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,.$$

Since the state vector is a unit vector, the coefficients α and β must satisfy

$$|\alpha|^2 + |\beta|^2 = 1.$$

The coefficients α and β are *probability amplitudes*. Even though a qubit can exist in a superposition of basis states, once *measured* (see Postulate 3), its state *collapses* to one of the basis states: $|\alpha|^2$ and $|\beta|^2$ give us the probability of finding the qubit, respectively, in states $|0\rangle$ and $|1\rangle$ after measurement.

One can draw an analogy with how the space of natural numbers, $\mathbb{N}$, can be extended to the space of real numbers, $\mathbb{R}$, and then to the space of complex numbers, $\mathbb{C}$. We have a much wider range of functions that can operate on and take values in $\mathbb{R}$ and $\mathbb{C}$ than in $\mathbb{N}$. Similarly, allowing the two-state system to exist in a superposition of states significantly extends the range of possible operators that can transform such states (i.e., perform computation).

For example, there is no Boolean function f that, when applied twice to a classical bit, would result in a NOT gate: $f(f(0)) = 1$ and $f(f(1)) = 0$. But there is such an operator in quantum computing. We can easily verify by direct calculations that the matrix

$$\mathrm{M} := \frac{1}{2} \begin{bmatrix} 1+\mathrm{i} & 1-\mathrm{i} \\ 1-\mathrm{i} & 1+\mathrm{i} \end{bmatrix},$$

applied twice to the basis vector $|0\rangle$ would transform it to the basis vector $|1\rangle$, and applied twice to the basis vector $|1\rangle$ would transform it to the basis vector $|0\rangle$. M is an example of a *quantum logic gate* – an operator that transforms the state of a qubit, thus implementing the computation.

Remark: The state space of a physical system can be infinite-dimensional. The quantum computing paradigm based on infinite-dimensional Hilbert spaces is called *continuous-variable quantum computing*, which is realised in, e.g., some photonic quantum computing systems. However, in the context of digital quantum computing, we will restrict our analysis to finite-dimensional state spaces.



The state of a qubit (the fundamental memory unit of quantum computing that generalises the concept of a classical bit) can be described mathematically as a unit vector in the two-dimensional complex vector space. Any physical system whose state space can be described by $\mathbb{C}^2$ can serve as an implementation of a qubit.

1.2.2 Second postulate - Dynamics

Postulate 2. *The time evolution of a closed quantum system is described by the Schrödinger equation*

$$i\hbar \frac{\mathrm{d}\,|\psi(t)\rangle}{\mathrm{d}t} = \mathcal{H}\,|\psi(t)\rangle\,,$$

where $\hbar$ is Planck's constant and $\mathcal{H}$ is a time-independent Hermitian operator known as the Hamiltonian of the system.

The Hamiltonian of a quantum system is an operator corresponding to the total energy of that system, and its eigenvalues are the possible energy levels of the system. The knowledge of the Hamiltonian provides all the necessary information about system dynamics.

In the Schrödinger equation (1.2.2), the state $|\psi(t_1)\rangle$ of a closed quantum system at time t_1 is related to the state $|\psi(t_2)\rangle$ at time t_2 by a unitary operator $\mathcal{U}(t_1,t_2)$ that depends only on t_1 and t_2 via

$$|\psi(t_2)\rangle = \mathcal{U}(t_1, t_2)\ |\psi(t_1)\rangle\,,$$

where $\mathcal{U}(t_1,t_2)$ is obtained from the Hamiltonian $\mathcal{H}$ as

$$\mathcal{U}(t_1, t_2) = \exp\left(-\frac{\mathrm{i}\mathcal{H}(t_2 - t_1)}{\hbar}\right).$$

Unitary operators preserve the inner product (and therefore norms, lengths, and distances), which means that for two vectors $|\mathrm{u}\rangle$ and $|\mathrm{v}\rangle$, if $\mathcal{U}$ is a unitary operator, then the inner product between $\mathcal{U}|\mathrm{u}\rangle$ and $\mathcal{U}|\mathrm{v}\rangle$ is the same as the inner product between $|\mathrm{u}\rangle$ and $|\mathrm{v}\rangle$:

$$\langle \mathrm{u}|\mathcal{U}^\dagger\mathcal{U}\ |\mathrm{v}\rangle = \langle \mathrm{u}|\mathrm{v}\rangle\,.$$

A unitary operator is a complex generalisation of a rotation: unitary operators take an orthonormal basis to another orthonormal basis, and any operator with this property is unitary. In quantum mechanics, physical transformations such as rotations,

translations and time evolution correspond to maps that take quantum states to other quantum states. These maps should be linear and preserve the inner product. This allows us to look at the unitary operators as the *quantum logic gates* implementing quantum computation protocols. Furthermore, unitary operators are invertible, a key property that ensures that quantum computing is *reversible*.



Quantum logic gates (quantum counterparts of the Boolean logic gates in classical computing) are unitary operators that transform quantum states, thus implementing the computation.

1.2.3 Third postulate - Measurement

Given a Hermitian operator $\mathcal{A}$, the spectral theorem implies that the state $|\psi\rangle$ of a system can be written as a superposition

$$|\psi\rangle = \sum_{i=1}^{N} \alpha_i \, |\psi_i\rangle \, ,$$

where the coefficients $(\alpha_i)_{i=1,\ldots,N}$ are complex *probability amplitudes*, assumed to be normalised with $\sum_{i=1}^{N} |\alpha_i|^2 = 1$, and where $(|\psi_i\rangle)_{i=1,\ldots,N}$ are eigenfunctions of $\mathcal{A}$. The measurement postulate then reads as follows:

Postulate 3. *If we measure the Hermitian operator $\mathcal{A}$ in the state $|\psi\rangle$ given in (1.2.3), the possible outcomes for the measurement are the eigenvalues $(\lambda_i)_{i=1,\ldots,N}$ of $\mathcal{A}$, and the probability p_i to measure λ_i is given by $p_i = |\alpha_i|^2$. After the outcome λ_i, the state of the system becomes*

$$|\psi\rangle = |\psi_i\rangle\,.$$

An immediate measurement in the same computational basis will deliver the same result without any uncertainty.

The quantum measurements are described by *measurement operators* $(\mathcal{P}_i)_{i=1,\ldots,N}$, acting on the state space of the system with N possible outcomes. If the state of the system is $|\psi\rangle$ before the measurement, then the probability of outcome i is

$$\mathbb{P}(i) = \langle\psi|\,\mathcal{P}_i^\dagger \mathcal{P}_i\,|\psi\rangle\,.$$

The measurement operators should also satisfy the *completeness condition*

$$\sum_{i=1}^{N} \mathcal{P}_i^\dagger \mathcal{P}_i = \mathcal{I},$$

where $\mathcal{I}$ is the identity operator. This ensures that the sum of the probabilities of all outcomes adds up to 1.

These measurement operators are linear but not unitary. From the quantum computing perspective, we are interested in measurement operators that are *projections* (Definition 2) onto the *computational basis*, such as the standard orthonormal basis given by (1.1.9).

For example, the measurement operators for a single qubit can be defined as

$$\mathcal{P}_0 := |0\rangle\langle 0| = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \quad \text{and} \quad \mathcal{P}_1 := |1\rangle\langle 1| = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}.$$

We can easily verify that $\mathcal{P}_0^2 = \mathcal{P}_0$ and $\mathcal{P}_1^2 = \mathcal{P}_1$, as should be the case for projection operators, and that the completeness condition (1.2.3) is satisfied. If the qubit is in state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, then the measurement operator $\mathcal{P}_0$ will give us $|0\rangle$ with probability $|\alpha|^2$, and the measurement operator $\mathcal{P}_1$ will give us $|1\rangle$ with probability $|\beta|^2$. Indeed,

$$\begin{aligned}\mathcal{P}_0|\psi\rangle &= |0\rangle\langle 0|\left(\alpha|0\rangle + \beta|1\rangle\right) = \alpha|0\rangle\langle 0|0\rangle + \beta|0\rangle\langle 0|1\rangle = \alpha|0\rangle,\\ \mathcal{P}_1|\psi\rangle &= |1\rangle\langle 1|\left(\alpha|0\rangle + \beta|1\rangle\right) = \alpha|1\rangle\langle 1|0\rangle + \beta|1\rangle\langle 1|1\rangle = \beta|1\rangle.\end{aligned}$$

The measurement postulate of quantum mechanics states that an immediate measurement in the same computational basis will deliver the same result without any uncertainty. The key words here are "the same computational basis". What would happen if the subsequent measurement is performed in another basis (the basis specified by another set of linearly independent unit vectors from the state space)? For example, assume that the qubit is in state

$$|\psi\rangle = \frac{1}{\sqrt{2}}\,|0\rangle + \frac{1}{\sqrt{2}}\,|1\rangle\,.$$

Measuring $|\psi\rangle$ in $\{|0\rangle,|1\rangle\}$ computational basis will result in observing states $|0\rangle$ and $|1\rangle$ with equal probability 1/2. Let us assume that we measured $|0\rangle$. The qubit state is now

$$|\psi'\rangle = 1\cdot|0\rangle + 0\cdot|1\rangle\,.$$

If we repeat the measurement in the same $\{|0\rangle,|1\rangle\}$ computational basis, we obtain state $|0\rangle$ with probability 1 in accordance with the measurement postulate. However, had we measured state $|\psi'\rangle$ in the *Hadamard basis* $\{|+\rangle,|-\rangle\}$, given by

$$|+\rangle := \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad \text{and} \quad |-\rangle := \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle),$$

we would have equal probabilities of $|+\rangle$ and $|-\rangle$ outcomes. Let us assume that we measured $|-\rangle$ and the state of the qubit is now

$$|\psi''\rangle = 0 \cdot |+\rangle + 1 \cdot |-\rangle .$$

If we repeat the measurement of state $|\psi''\rangle$ in the Hadamard basis $\{|+\rangle,|-\rangle\}$, we obtain state $|-\rangle$ with probability 1. But the state of the qubit is an equal superposition of states $|0\rangle$ and $|1\rangle$ from the $\{|0\rangle,|1\rangle\}$ computational basis perspective and we have an equal chance of measuring either $|0\rangle$ or $|1\rangle$ in this basis.

Remark: The basis vectors $|0\rangle$ and $|1\rangle$ that form the standard computational basis can be transformed into the basis vectors $|+\rangle$ and $|-\rangle$ that form the Hadamard basis by applying the following unitary operator (rotation), called the *Hadamard gate*:

$$\mathtt{H} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

Chapters 6, 10 and 11 provide examples of applications of the Hadamard gate.

The measurement plays a crucial role in quantum computing. This is the process of *collapsing* a quantum state and reading out the classical information: measuring qubits encoding a quantum state will produce a classical bit string. The measurement process generates probabilistic outcomes. Therefore, we need to perform measurements on the same quantum state multiple times to

generate a sufficiently large number of classical bit strings to produce reliable statistics.



The process of measurement describes the collapse of the quantum state due to contact with the environment. After measurement, the states of the qubits are known without any uncertainty. It is possible to extract at most 1 bit of information from a qubit. In order to extract more information about the probability distribution encoded in a given quantum state, it is necessary to perform measurement of the same state multiple times.

1.2.4 Fourth postulate - Observable

Postulate 4. *For every measurable property of a physical system, there exists a corresponding Hermitian operator. The values of the physical observables correspond to the expectation values of Hermitian operators. The expectation value* $\langle \mathcal{A} \rangle$ *of the Hermitian operator* $\mathcal{A}$ *in the normalised state* $|\psi\rangle$ *is given by*

$$\langle \mathcal{A} \rangle := \langle \psi | \, \mathcal{A} \, | \psi \rangle \, .$$

Let us consider the general case where the expectation value of a Hermitian operator $\mathcal{A}$ is calculated in state $|\psi\rangle$, which is not an eigenfunction of $\mathcal{A}$. By the Spectral Theorem 3 (see also (1.2.3)), the state $|\psi\rangle$ of a system can be represented as the superposition

$$|\psi\rangle = \sum_{i=1}^{N} \alpha_i \, |\psi_i\rangle \, ,$$

where $(|\psi_i\rangle)_{i=1,\ldots,N}$ are the eigenfunctions of $\mathcal{A}$ and $(\alpha_i)_{i=1,\ldots,N}$ the corresponding probability amplitudes.

Therefore, the *expectation value* of $\mathcal{A}$ in state $|\psi\rangle$, given in (1.2.4), is calculated as

$$\langle \mathcal{A} \rangle = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i^* \alpha_j \langle \psi_i | \mathcal{A} | \psi_j \rangle = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i^* \alpha_j \lambda_j \langle \psi_i | \psi_j \rangle ,$$

where $(\lambda_i)_{i=1,\ldots,N}$ are the eigenvalues of $\mathcal{A}$. The only terms that survive in the expression for $\langle \mathcal{A} \rangle$ are those with $i = j$ due to the orthogonality of the eigenfunctions, so that

$$\langle \mathcal{A} \rangle = \sum_{i=1}^{N} \alpha_i^* \alpha_i \lambda_i = \sum_{i=1}^{N} |\alpha_i|^2 \lambda_i .$$

Therefore, the value of the observable is a weighted average of the eigenvalues of the corresponding Hermitian operator. The weights are the coefficients $(|\alpha_i|^2)_{i=1,\ldots,N}$, which are the probabilities of measuring the corresponding eigenstate of $\mathcal{A}$.



Hermitian operators play an exceptionally important role in quantum mechanics since their expectation values correspond to physical observables.

1.2.5 Fifth postulate - Composite System

Postulate 5. *The state space of a composite physical system is the tensor product of the state spaces of the individual component physical systems.*

If the first component physical system is in state $|\psi_A\rangle$ and the second component physical system is in state $|\psi_B\rangle$, then the state of the combined system, $|\psi\rangle$, is given by the tensor product

$$|\psi\rangle = |\psi_A\rangle \otimes |\psi_B\rangle .$$

Not all states of a combined system can be separated into the tensor product of states of individual components. If the state of a system cannot be separated into component parts, we say that the component parts are *entangled*.

The entanglement of quantum systems is one of the major sources of computational power of quantum computing. It allows us to store exponentially more information in the correlations between the states of individual subsystems (in the limit – individual qubits) than directly in the states of individual subsystems.

To illustrate this point, we can look at the number of probability amplitudes needed to describe the state of an n-qubit system. An individual qubit can be found in one of the two possible states after measurement – one of the two basis states, $|0\rangle$ or $|1\rangle$. This means that we need to specify two probability amplitudes to fully describe the state of the qubit before measurement. If all our qubits are independent and the state of the system can be represented as a tensor product of individual qubit states,

$$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes \ldots \otimes |\psi_n\rangle ,$$

then we need to specify $2n$ probability amplitudes (two for each individual quantum states) to describe the state $|\psi\rangle$ of the system. If, however, all individual qubits are entangled and the tensor product representation of the system state $|\psi\rangle$ does not exist, we

need to specify 2^n probability amplitudes – this is an effective measure of useful information that can be stored in the system.



The power of quantum computing is derived from the principles of superposition and entanglement. Entanglement allows us to store most of the information in correlations between the qubit states.

1.3 Pure and Mixed States

There are situations where the state of a quantum mechanical system cannot be described with the help of a state vector. Here, we look at such situations and provide a mathematical tool for describing them.

1.3.1 Density matrix

Let us start with the state of a combined two-component physical system given by (1.2.5). Let $(|i\rangle)_{i=1,\ldots,N}$ and $(|j\rangle)_{j=1,\ldots,M}$ denote, respectively, the standard orthonormal bases of the Hilbert spaces of systems *A* and *B*:

$$|\psi_A\rangle = \sum_{i=1}^{N} \alpha_i \, |i\rangle \,, \quad |\psi_B\rangle = \sum_{j=1}^{M} \beta_j \, |j\rangle \,,$$

where $(\alpha_i)_{i=1,\ldots,N}$ and $(\beta_j)_{j=1,\ldots,M}$ are some probability amplitudes. The states that allow the state vector representation (1.3.1) are called *pure* states. In this case, the state of the combined system is

$$|\psi\rangle = |\psi_A\rangle \otimes |\psi_B\rangle = \sum_{i=1}^{N} \sum_{j=1}^{M} \alpha_i \beta_j \, |i\rangle \otimes |j\rangle \, .$$

However, in general, the state of the combined system would look like

$$|\psi\rangle = \sum_{i=1}^{N} \sum_{j=1}^{M} \gamma_{ij} \, |i\rangle \otimes |j\rangle \, ,$$

where γ_{ij} are probability amplitudes that may not necessarily be factorised as the product of probability amplitudes $(\alpha_i)_{i=1,\ldots,N}$ and $(\beta_j)_{j=1,\ldots,M}$. If γ_{ij} cannot be factorised as $\alpha_i\beta_j$, then the component systems *A* and *B* are entangled and their states cannot be represented by the state vectors (1.3.1). Such states of systems *A* and *B* are called *mixed* states.

The more general setup is that of an *ensemble* of states of the form $\{p_k, |\psi_k\rangle\}_{k=1,\ldots,N}$, where each $|\psi_i\rangle$ is a quantum state whose wavefunction is known with certainty (although this does not necessarily provide full knowledge of the measurement statistics), and each p_k is the associated probability (not amplitude) in [0,1]. In order to define properly pure and mixed states, introduce the *density operator* as follows:

Definition 7. A density operator ρ is a positive semidefinite Hermitian operator with unit trace and takes the form

$$\rho := \sum_{k=1}^{N} p_k \, |\psi_k\rangle \, \langle\psi_k| \, ,$$

where $\sum_{k=1}^{N} p_k = 1$ and $\langle\psi_k|\psi_l\rangle$ equals 1 if $k = l$ and zero otherwise.

Mathematically, such a density operator ρ corresponds to a *density matrix* $(\rho_{kl})_{k,l=1,\dots,N}$ such that

$$\rho = \rho^\dagger, \quad \mathrm{Tr}(\rho) \equiv \sum_{k=1}^{N} \rho_{kk} = 1, \quad \rho_{kk} \geq 0, \quad \text{for all } k = 1, \dots, N.$$

1.3.2 Pure state

A pure state is one that can be represented by a state vector

$$|\psi\rangle = \sum_{i=1}^{N} \alpha_i \, |i\rangle \,,$$

where $(\alpha_i)_{i=1,\dots,N}$ are probability amplitudes in $\mathbb{C}$ such that $\sum_{i=1}^{N} |\alpha_i|^2 = 1$. In the ensemble setup above, this means that there exists $k^* \in \{1,\dots,N\}$ such that $p_{k^*} = 1$ and hence $|\psi\rangle = |\psi_{k^*}\rangle$ and therefore $\rho = |\psi\rangle\langle\psi|$. The density matrix also allows us to compute expectations of the form (1.2.4):

Lemma 3. *Let ρ be the density matrix associated to the pure state (1.3.2) and let $\mathcal{A}$ be an observable (Hermitian operator), then*

$$\langle \mathcal{A} \rangle := \langle \psi | \, \mathcal{A} \, | \psi \rangle = \mathrm{Tr}(\rho \mathcal{A}).$$

Proof. The lemma follows from the immediate computation

$$\begin{aligned} \langle \psi | \mathcal{A} | \psi \rangle &= \langle \psi | \mathcal{A} \sum_{i=1}^{N} \alpha_i |i\rangle \\ &= \sum_{i=1}^{N} \alpha_i \langle \psi | \mathcal{A} | i \rangle \\ &= \sum_{i=1}^{N} \langle i | \psi \rangle \langle \psi | \mathcal{A} | i \rangle \\ &= \sum_{i=1}^{N} \langle i | \rho \mathcal{A} | i \rangle = \mathrm{Tr}(\rho \mathcal{A}). \end{aligned}$$

□

With the state $|\psi\rangle$ given by (1.3.2), we obtain

$$\langle \mathcal{A} \rangle = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j^* \langle j | \mathcal{A} | i \rangle .$$

At the same time we have

$$\langle \mathcal{A} \rangle = \mathrm{Tr}(\rho \mathcal{A}) = \sum_{i=1}^{N} \sum_{j=1}^{N} \rho_{ij} \langle j | \mathcal{A} | i \rangle .$$

Comparison of (1.3.2) and (1.3.2) yields the following expression for the density matrix of a pure state:

$$\rho_{ij} = \alpha_i \alpha_j^*, \quad \rho = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j^* |i\rangle \langle j| = |\psi\rangle \langle \psi| .$$

Example: An example of a pure state is the Hadamard state

$$|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

with corresponding density matrix

$$\rho = |+\rangle \langle +| = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}.$$

1.3.3 Mixed state

A mixed state is one that cannot be represented by a single pure state vector, and is therefore represented as a statistical distribution of pure states in the form of an ensemble of quantum states $\{p_k, |\psi_k\rangle\}_{k=1,\ldots,N}$, where $\sum_{k=1}^{N} p_k = 1$ and $p_k \in [0,1]$ for each k. The density of a mixed state therefore reads

$$\rho = \sum_{k=1}^{N} p_k \, |\psi_k\rangle \, \langle\psi_k| \, .$$

Similarly to Lemma 3, we can write expectations of observables with respect to mixed states using the density matrix:

Lemma 4. *Let ρ be the density matrix associated to the mixed state (1.3.3) and let $\mathcal{A}$ be an observable (Hermitian operator), then*

$$\mathrm{Tr}(\rho\mathcal{A}) = \sum_{k=1}^{N} p_k \, \langle\psi_k| \, \mathcal{A} \, |\psi_k\rangle \, .$$

Proof. The lemma follows from the immediate computation

$$\begin{aligned}
\mathrm{Tr}(\rho\mathcal{A}) &= \sum_{i=1}^{N} \langle i| \, \rho\mathcal{A} |i\rangle \\
&= \sum_{i=1}^{N} \langle i| \left(\sum_{k=1}^{N} p_k \, |\psi_k\rangle \, \langle\psi_k| \right)_{\mathcal{A}} |i\rangle \\
&= \sum_{k=1}^{N} p_k \left(\sum_{i=1}^{N} \langle i|\psi_k\rangle \, \langle\psi_k| \, \mathcal{A} \, |i\rangle \right) \\
&= \sum_{k=1}^{N} p_k \, \langle\psi_k| \, \mathcal{A} |\psi_k\rangle .
\end{aligned}$$

□

Let us see now how the density matrix formalism can help us describe the state of a combined system. Consider an entangled state of two systems, A and B, given by (1.3.1), and a Hermitian operator $\mathcal{A}$ that only acts within the Hilbert space of system A. What would be the expectation value of $\mathcal{A}$ in this state? Starting with (1.2.4), we obtain

$$\langle \mathcal{A} \rangle = \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{k=1}^{N} \sum_{l=1}^{M} \gamma_{ij} \gamma_{kl}^{*} \langle k | \mathcal{A} | i \rangle \langle l | j \rangle .$$

Since only terms with $l = j$ survive in (1.3.3) due to the orthogonality of the basis states, we have

$$\langle \mathcal{A} \rangle = \sum_{i=1}^{N} \sum_{k=1}^{N} \left(\sum_{j=1}^{M} \gamma_{ij} \gamma_{kj}^{*} \right) \langle k | \mathcal{A} | i \rangle .$$

Thus, the density matrix that describes the mixed state of system A is

$$\rho_{ik} = \sum_{j=1}^{M} \gamma_{ij} \gamma_{kj}^{*}.$$

Note that in the case where the probability amplitudes γ_{ij} can be factorised as the product of probability amplitudes $(\alpha_i)_{i=1,\ldots,N}$ and $(\beta_j)_{j=1,\ldots,M}$, we obtain

$$\rho_{ik} = \sum_{j=1}^{M} \alpha_i \beta_j \alpha_k^{*} \beta_j^{*} = \alpha_i \alpha_k^{*} \sum_{j=1}^{M} |\beta_j|^2 = \alpha_i \alpha_k^{*},$$

which describes a pure state.

A simple criterion to distinguish a pure state from a mixed state is the following:

Lemma 5. *Let* ρ *be a density matrix. The inequality* $\mathrm{Tr}(\rho^2) \leq 1$ *always holds and* $\mathrm{Tr}(\rho^2) = 1$ *if and only if* ρ *corresponds to a pure state.*

Proof. Consider an ensemble of pure states $\{p_i, |\psi_i\rangle\}_{i=1,\ldots,N}$, with density matrix given by (1.3.3). Therefore

$$\begin{aligned}\mathrm{Tr}(\rho^2) &= \mathrm{Tr}\left(\left(\sum_{i=1}^{N} p_i\, |\psi_i\rangle\langle\psi_i|\right)\left(\sum_{j=1}^{N} p_j\, |\psi_j\rangle\langle\psi_j|\right)\right)\\ &= \mathrm{Tr}\left(\sum_{i=1}^{N}\sum_{j=1}^{N} p_i p_j\, |\psi_i\rangle\langle\psi_i|\,|\psi_j\rangle\langle\psi_j|\right)\\ &= \mathrm{Tr}\left(\sum_{i=1}^{N} p_i^2\, |\psi_i\rangle\langle\psi_i|\right) = \sum_{i=1}^{N} p_i^2\,\mathrm{Tr}\left(|\psi_i\rangle\langle\psi_i|\right) = \sum_{i=1}^{N} p_i^2\langle\psi_i|\psi_i\rangle = \sum_{i=1}^{N} p_i^2,\end{aligned}$$

which is smaller than 1 since the p_i are probabilities in [0,1] summing up to 1. Assume now that $\mathrm{Tr}(\rho^2)$ equals one, then so does $\sum_{i=1}^{N} p_i^2$. If $p_i \in (0,1)$ for all $i = 1,\ldots,N$, then

$$1 = \sum_{i=1}^{N} p_i^2 < \sum_{i=1}^{N} p_i = 1,$$

which is a contradiction, and therefore there exists $i^* \in \{1,\ldots,N\}$ such that $p_{i^*} = 1$, so that $\rho = |\psi_{i^*}\rangle\langle\psi_{i^*}|$ is a pure state. Conversely, if $\rho = |\psi_i\rangle\langle\psi_i|$ for some $i \in \{1,\ldots,N\}$ represents a pure state, then

$$\mathrm{Tr}(\rho^2) = \mathrm{Tr}(|\psi_i\rangle\langle\psi_i|\,|\psi_i\rangle\langle\psi_i|) = \mathrm{Tr}(|\psi_i\rangle\langle\psi_i|) = \langle\psi_i|\psi_i\rangle = 1.$$

□

Example: An example of a mixed state is a *statistical ensemble* of states $|0\rangle$ and $|1\rangle$. If a physical system is prepared to be either in state $|0\rangle$ or state $|1\rangle$ with equal probability, it can be described by the mixed state

$$\rho = \frac{1}{2}|0\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1| = \frac{1}{2}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Note that this is different from the density matrix of the pure state

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle),$$

which reads

$$\rho_\psi = |\psi\rangle\langle\psi| = \frac{1}{2}(|0\rangle+|1\rangle)(\langle 0|+\langle 1|) = \frac{1}{2}(|0\rangle\langle 0|+|1\rangle\langle 0|+|0\rangle\langle 1|+|1\rangle\langle 1|) = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}.$$



Unlike pure quantum states, mixed quantum states cannot be described by a single state vector. However, the pure states and the mixed states can be described by the density matrix.

Summary

In this chapter, we learned the key principles of quantum mechanics, starting with a review of the basic elements of linear algebra, followed by an introduction to Dirac notations.

We then covered the main postulates of quantum mechanics and their relevance to quantum computing. We learned how to describe the state (statics) and the evolution (dynamics) of a closed system, the interactions of a system with external systems (measurement), observables, as well as the state of a composite system in terms of its component parts.

We finally introduced the density operator, which allows us to describe both pure and mixed quantum states, contrasting with the state vector, which can only represent pure quantum states.

In the next chapter, we will look at an application of the principles of quantum mechanics to analog quantum computing – quantum annealing.

Join our book's Discord space

Join our Discord community to meet like-minded people and learn alongside more than 2000 members at:
https://packt.link/quantum



Part I
Analog Quantum Computing – Quantum Annealing

2
Adiabatic Quantum Computing

Search algorithms are among the most important and fundamental algorithms in computer science, the most basic example being that of finding one special item among a list of N items. Classical algorithms are known to solve this problem in time proportional to the problem size, N, which becomes highly untractable when the latter grows large. In 1996, Grover [117] devised a quantum algorithm to solve such search problems with a quadratic speedup, with the obvious caveat that quantum computers did not exist at the time. Soon after, Farhi, Goldstone, Gutmann and Sipser [98] recast the Grover problem as a satisfiability problem in the context of quantum computation by adiabatic evolution.

Another class of problems hard to solve classically is that of combinatorial optimisation problems. The truck dispatching problem, originally proposed by Dantzig and Ramser [78], searches the optimal routing of delivery trucks, and is a generalisation of the famous travelling salesman problem. Other well-known optimisation problems of the same class include partitioning problems and binary integer linear programming, to name just a few. The exact algorithms are only efficient for small-scale instances. Heuristics and metaheuristics (e.g., evolutionary search heuristics such as Genetic Algorithm and Particle Swarm Optimisation) are often more suitable for practical applications, because real-world problems are considerably larger in scale. Since search time scales exponentially with problem size, there has always been a strong motivation for finding alternative approaches to solve such problems with a wide range of practical applications,

not least in finance. Adiabatic Quantum Computing (AQC) was ready to enter the stage.

2.1 Complexity of Computational Problems

In this and the following chapters, we will often speak about computational problems that are hard for classical computers but can be solved efficiently using quantum algorithms and hardware. How can we quantify the hardness of a computational problem? One way to answer this is to analyse problems from the computational resource perspective: how much time and memory are needed to solve them? This leads to the concept of *complexity classes*. Important examples are as follows:

- The class P (*polynomial*) is the set of decision problems solvable by a deterministic Turing machine in polynomial time.
- The class NP (*non-deterministic polynomial*) is the set of decision problems solvable by a non-deterministic Turing machine in polynomial time.

These definitions, in turn, require us to specify the following objects:

- *A decision problem* is a computational problem that can be posed as a Yes-No question of the input values.
- *Polynomial time* means that the running time of the algorithm is bounded above by a polynomial expression in the size of the input for the algorithm.
- *A Turing machine* is an abstract model of computation that is general enough to embody any computer problem.

- *A deterministic Turing machine* is the most basic type of Turing machine that uses a fixed set of rules to determine its future actions.
- A *non-deterministic Turing machine* is a Turing machine that is able to explore multiple alternative future actions from a given state.

In terms of computational hardness, we will pay special attention to the problems that are *NP-complete* and *NP-hard*.

A problem is NP-complete when

1. the correctness of each solution can be verified in polynomial time and a brute-force search algorithm can find a solution by trying all possible solutions;
2. it can be used to simulate every other problem for which we can verify in polynomial time that a solution is correct.

NP-complete problems are the hardest, for which solutions can be verified quickly (in polynomial time). If we could find solutions to some NP-complete problems quickly, we could quickly find the solutions to every other problem to which a given solution can be easily verified.

A problem is NP-hard when every problem in NP can be reduced to it in polynomial time. Alternatively, a problem is NP-hard when every NP-complete problem can be reduced to it in polynomial time. Since every problem in NP reduces to an NP-complete problem in polynomial time, the second definition implies the first one.

The NP-hard class is not restricted to decision problems and also includes search problems and optimisation problems. This means

that NP-hard problems do not have to be elements of the complexity class NP.

Arguably, the most important open problem in computer science is whether P = NP. It is widely believed, although it has not been proven yet, that P ≠ NP. Figure 2.1 shows the relationship between complexity classes for both scenarios.

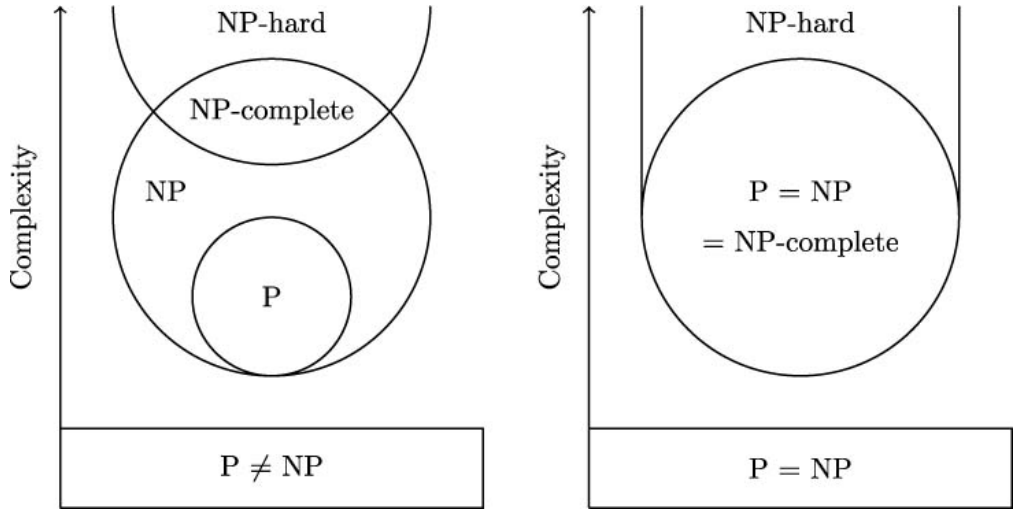


Figure 2.1: Schematic illustration of the relationship between P, NP, NP-complete and NP-hard sets of problems.

2.2 Principles of Adiabatic Quantum Computing

Adiabatic quantum optimisation is a promising approach to solving NP-complete and NP-hard problems [97]. Assume that a solution to the optimisation problem is encoded in the ground state (i.e., the quantum state corresponding to the lowest eigenvalue) of a quantum Hamiltonian $\mathcal{H}_F$. By the second postulate of quantum mechanics (Section 1.2.2), the dynamics of a quantum system is fully specified by its Hamiltonian. If we know how to encode the objective function that we want to minimise in

the Hamiltonian of a quantum system, then finding the ground state of the Hamiltonian is equivalent to finding the set of decision variables that minimises the objective function.

As a simple example of equivalence between the minimum of a function and the ground state of a Hamiltonian, consider a function $f : \{0,1\}^n \to \mathbb{R}$ that needs to be minimised and take the Hamiltonian

$$\mathcal{H}_F := \sum_{z \in \{0,1\}^n} f(z) \, |z\rangle \langle z| \, .$$

Clearly, for any $z_0 \in \{0,1\}^n$,

$$\mathcal{H}_F \, |z_0\rangle = \left(\sum_{z \in \{0,1\}^n} f(z) \, |z\rangle \langle z| \right) |z_0\rangle = f(z_0) \, |z_0\rangle \langle z_0|z_0\rangle = f(z_0) \, |z_0\rangle \, ,$$

since the computational basis $(|z\rangle)_{z \in \{0,1\}^n}$ is orthonormal. Therefore any $z_0 \in \{0,1\}^n$ is an eigenstate of $\mathcal{H}_F$ with eigenvalue $f(z_0)$. Minimising f is thus clearly equivalent to finding the lowest eigenvalue of the Hamiltonian $\mathcal{H}_F$.

Let us further assume that we have another quantum Hamiltonian, $\mathcal{H}_0$, whose ground state is easy to find and easy to prepare in an experimental setup. Then, if we prepare a quantum system to be in the ground state of $\mathcal{H}_0$, and then adiabatically (slowly) change the system Hamiltonian, $\mathcal{H}(t)$, from $\mathcal{H}_0$ at $t = 0$ to $\mathcal{H}_F$ at $t = \tau$ according the following time evolution:

$$\mathcal{H}(t) = \left(1 - \frac{t}{\tau}\right) \mathcal{H}_0 + \frac{t}{\tau} \mathcal{H}_F,$$

then if τ is large enough, and $\mathcal{H}_0$ and $\mathcal{H}_F$ do not commute, the quantum system will remain in the ground state at all times according to the quantum adiabatic theorem. Measuring the quantum state at $t = \tau$ will produce a solution of our problem (a bitstring that encodes an optimal configuration of binary decision variables).

We provide a detailed explanation of the adiabatic quantum optimisation algorithm as well as the quantum adiabatic theorem in Section 2.2.1. In terms of potentially achievable quantum speedup in comparison with the best classical algorithms, for a problem of size N, quantum optimisers solve NP-hard combinatorial optimisation problems in time proportional to

$$\exp(\beta N^{\gamma}),$$

as N tends to infinity, for positive coefficients β and γ, which may be smaller than known classical algorithms [197]. In fact, early experiments on quantum annealers that realised the principles of adiabatic quantum computing demonstrated several orders of magnitude quantum speedup (ignoring various computational overheads that are likely to be reduced as the quantum annealing technology matures) [175, 201, 296].

The coefficient γ is by far the most important. This can be illustrated by the following table that provides the estimates of the computation time as a function of problem size. Assuming that a single operation takes 1 microsecond, and the number of operations scales as either 2^N ("classical benchmark") or $e^{\sqrt{N}}$ ("quantum optimisation" with $\gamma = 0.5$), we obtain the following results:

N	2^N	$e^{\sqrt{N}}$
10	1 millisecond	0.024 milliseconds
50	35.7 years	1.2 milliseconds
100	4×10^{16} years	22 milliseconds
500	10^{137} years	1.4 hours

Table 2.1: Computational time as a function of the problem size.

The asymptotic estimate of the duration of adiabatic system evolution, T, which is exponential in problem size, is a consequence of the requirement that the system should always stay in the ground state of the local Hamiltonian. As the gap between the ground state and the first excited state becomes small at some point, the system evolution process should slow down accordingly.

However, if we only are interested in an approximate solution (and are willing to accept some deterioration in the quality of the obtained solution), we can expect the NP-hard combinatorial problems to be solved in polynomial time proportional to N^γ for $\gamma > 0$, as N becomes large [23, 253].



The power of adiabatic quantum computing lies in its ability to solve hard computational problems through the natural evolution of a physical system.

2.2.1 The Quantum Adiabatic Theorem

In the Schrödinger equation (1.2.2) (normalised with $\hbar = 1$) with constant Hamiltonian $\mathcal{H}$, if the system starts in $|\psi(0)\rangle$ then the

solution moves to

$$|\psi(t)\rangle = \mathrm{e}^{-\mathrm{i}\mathcal{H}t}\,|\psi(0)\rangle$$

at time $t \geq 0$. This in particular implies that any eigenstate $|\psi_0\rangle$ of $\mathcal{H}$, satisfying $\mathcal{H}|\psi_0\rangle = \lambda_0|\psi_0\rangle$ for some eigenvalue λ_0, will evolve through the Schrödinger equation from $|\psi_0\rangle$ to

$$|\psi(t)\rangle = \mathrm{e}^{-\mathrm{i}\mathcal{H}t}\,|\psi_0\rangle = \mathrm{e}^{-\mathrm{i}\lambda_0 t}\,|\psi_0\rangle\,,$$

namely the eigenstate only gains a phase $\mathrm{e}^{-\mathrm{i}\lambda_0 t}$ and there is no transition over time between different eigenstates. The more interesting case, which we consider now, is that of a time-dependent Hamiltonian.

Consider again the Schrödinger equation (1.2.2) (normalised with $\hbar = 1$) over the time interval $[0,\tau]$, where now the Hamiltonian $\mathcal{H}$ is a function of time. The time change $t(\cdot)$, such that $t(0) = 1$ and $t(1) = \tau$, yields

$$\mathrm{i}\frac{\mathrm{d}\,|\psi(s)\rangle}{\mathrm{d}s} = t'(s)\mathcal{H}(s)\,|\psi(s)\rangle$$

over the unit time interval $[0,1]$. It is important to note here that the Hamiltonian $\mathcal{H}$ has no dependence on the time horizon τ itself. This excludes in particular Hamiltonians with multiple timescales as in [206]. Here, we are chiefly interested in Hamiltonians of the (slightly generalised) form (2.2),

$$\mathcal{H}(s) = r(s)\mathcal{H}_0 + (1 - r(s))\mathcal{H}_F,$$

for two given Hamiltonians $\mathcal{H}_0$ and $\mathcal{H}_F$, where $r(\cdot)$ is a continuous adiabatic evolution path decreasing from $r(0) = 1$ to $r(1) = 0$. The

standard adiabatic schedule is given by $r(s) = 1 - s$.

The gist of the quantum adiabatic theorem is the following. Assume that the system starts from the ground state of $\mathcal{H}_0$. If the time evolution of the Hamiltonian is sufficiently slow, then the system remains in the ground state of the evolving Hamiltonian up to time 1. It was originally proposed by Born and Fock [41] and generalised by Kato [156] using the theory of perturbation of linear operators.

In order to state it properly, denote by $|\psi(\cdot)\rangle$ the solution to the Schrödinger equation (2.2.1), so that for any $s \in [0,1]$, there exists a unitary operator $\mathcal{U}$ for which

$$|\psi(s)\rangle = \mathcal{U}(s)\ |\psi(0)\rangle\ .$$

Consider a Hamiltonian of the form (2.2.1) and a time-change $t(s) = s\tau$ (hence $t'(s) = \tau$), so that the Schrödinger evolution dynamics reads

$$\mathrm{i}\frac{\mathrm{d}\ |\psi(t)\rangle}{\mathrm{d}t} = \tau\mathcal{H}(t)\ |\psi(t)\rangle\ ,$$

over the interval [0,1]. For each $t \in [0,1]$, we denote $|\phi(t)\rangle$ the ground state of $\mathcal{H}_t$.

We finally present the following version of the quantum adiabatic theorem, due to Jansen, Seiler and Ruskai [149]. We recall that an eigenvalue is called non-degenerate if there exists only one eigenstate associated with this eigenvalue. For any $t \in [0,1]$, given the Hamiltonian $\mathcal{H}(t)$, we denote by Δ_t the (strictly positive) energy gap between the lowest eigenvalue of $\mathcal{H}_t$ and the next one.

Theorem 7 (Quantum Adiabatic Theorem)**.** *Assume that, for any* $t \in [0,1]$ *the Hamiltonian* $\mathcal{H}(t)$ *admits a non-degenerate ground state and that there exists* $\varepsilon > 0$ *such that*

$$\frac{2}{\varepsilon}\left\{c_0\frac{\|\mathcal{H}'(0)\|}{\Delta_0^2}+c_1\frac{\|\mathcal{H}'(1)\|}{\Delta_1^2}+\int_0^1\left[\left(3c_1^2+c_1+c_3\right)\frac{\|\mathcal{H}'(s)\|^2}{\Delta_s^3}+c_2\frac{\|\mathcal{H}''(s)\|}{\Delta_s^2}\right]\mathrm{d}s\right\}\leq\tau.$$

Then, starting the system (2.2.1) *in the state* $|\psi(0)\rangle = |\phi(0)\rangle$*, the Schrödinger evolution yields at time* 1 *a state* $|\psi(1)\rangle$ *satisfying*

$$\| |\phi(1)\rangle - |\psi(1)\rangle\| \leq \varepsilon.$$

This quantitative version of the adiabatic theorem provides an estimate on how large the time horizon τ needs to be be in order to achieve sufficient accuracy ε. Consider, for example, the interpolation scheme (2.2.1) with $r(s) = 1 - s$, so that

$$\mathcal{H}(s) = (1-s)\mathcal{H}_0 + s\mathcal{H}_F,$$

and, therefore, $\mathcal{H}'(s) = -\mathcal{H}_0 + \mathcal{H}_F$ and $\mathcal{H}''(s) = 0$. In that case, the quantitative estimate in Theorem 7 simplifies to the following:

Corollary 1. *With the same assumptions as in Theorem 7 and with the interpolation scheme above, the quantitative estimate*

$$\tau \geq \frac{2}{\varepsilon}\left\{c_0\frac{\|\mathcal{H}_F-\mathcal{H}_0\|}{\Delta^2}+\left(3c_1^2+c_1+c_3\right)\frac{\|\mathcal{H}_F-\mathcal{H}_0\|^2}{\Delta^3}\right\},$$

with $\Delta := \min_{s\in[0,1]}\Delta_s$*, ensures again that* $\| |\phi(1)\rangle - |\psi(1)\rangle\| \leq \varepsilon$.

This corollary in particular highlights the importance of the spectral gap Δ. The smaller it is, the longer one has to wait to see the adiabatic property become efficient. There exist different versions of the quantum adiabatic theorem, each with slightly

different assumptions. A weak form was proved by Avron and Elgart [19] and by Bornemann [40] without the gap condition.

The proof of the quantum adiabatic theorem is rather technical and many versions exist, with slightly different proofs. They all rely, however, on analysing the evolution operator $\mathcal{U}(\cdot)$ corresponding to the Hamiltonian, which clearly solves

$$\frac{\mathrm{d}\mathcal{U}(t)}{\mathrm{d}t} = -\mathrm{i}\tau\mathcal{H}(t)\mathcal{U}(t)$$

starting from the identity. In particular, one needs to construct an adiabatic operator $\mathcal{U}_{\mathrm{A}}(\cdot)$ solving the same Schrödinger equation, replacing $\mathcal{H}(\cdot)$ by its adiabatic version

$$\mathcal{H}_A(t) = \mathcal{H}(t) + \frac{\mathrm{i}}{\tau}[\mathcal{P}'(t), \mathcal{P}(t)],$$

where $\mathcal{P}(t)$ denotes the projection operator onto the desired eigenstate of $\mathcal{H}(t)$. The proof then follows from showing that $\mathcal{U}_{\mathrm{A}}$ and $\mathcal{U}$ are close enough as τ becomes large.

By connecting the geometric properties of the adiabatic limit of the quantum system to parallel transport in a vector bundle, Berry's [33] and Simon's [267] works gave rise to Geometric Quantum Computing. Van Dam, Mosca and Vazirani [291] showed that it is possible to construct a discrete-time approximation of the evolution operator $\mathcal{U}$ with only polynomial time overheads.



The quantum adiabatic theorem provides the theoretical background for adiabatic quantum

computing.

2.2.2 Optimisation and metaheuristics

Metaheuristics are used to find "good" approximate solutions to general optimisation problems. In plain terms, a metaheuristic is a search policy that explores the optimisation function $f(\cdot)$ by evaluating it at certain points. There are myriad metaheuristic algorithms that decide where next (at which value of x) to evaluate $f(x)$ given the history of function evaluations, but all are based on the same essential principle that good solutions are likely to be near other good solutions, or in other words that the optimisation surface has some smoothness. This in turn reveals the exploration versus exploitation trade-off that all metaheuristics must make.

A metaheuristic can exploit its "current" position, by descending incrementally. The risk is that this returns a (possibly not very good) local minimum. Alternatively, a metaheuristic can explore the optimisation surface by making "large movements" to discover whether another part of the optimisation surface returns smaller values of $f(x)$. In this case, the global minimum may be found, but the value of x returned may only be a fairly poor approximation of the actual global minimum.

Simulated annealing

Simulated annealing is a metaheuristic inspired by thermal annealing. Consider the minimisation of a given function $f : D \to \mathbb{R}$ over some domain $D \subset \mathbb{R}^n$. The algorithm works as follows:

1. Start with an initial value $x \in D$ and compute $f(x)$.

2. Randomly choose a neighbour y of x and evaluate $f(y)$.
3. If $f(y) < f(x)$, then set x = y.
4. Else, either keep x as is or set x = y.
5. Repeat until an end criterion is attained.

The crucial step is the random choice in Step 4, designed to avoid being stuck in a local minimum and to favour, at least at the beginning of the algorithm, exploration rather than exploitation. In the case where $f(y) \geq f(x)$, we shall hence make the switch $y \mapsto x$ with the probability

$$\mathbb{P}(\text{switch}) = \exp\left(-\frac{f(y) - f(x)}{\tau}\right),$$

where τ plays the role of the thermal annealing temperature: when the system is hot, particles move (exploration), and it cools down when refinement (exploitation) is required.

Quantum annealing and quantum tunnelling

Quantum annealing combines the idea of simulated annealing with the quantum adiabatic theorem by considering the time-dependent Hamiltonian

$$\mathcal{H}(t) = \mathcal{H}_F + \Gamma(t)\mathcal{H}_0,$$

where $\mathcal{H}_F$ is the final, *longitudinal field* Hamiltonian whose ground state encodes the optimal solution of the minimisation problem and $\mathcal{H}_0$ is the initial, *transverse field* Hamiltonian, assumed not to commute with $\mathcal{H}_F$. The function Γ is the *transverse field coefficient*, playing the role of the thermal temperature, namely a continuous decreasing function of t converging to zero as t approaches the

final time horizon. By the quantum adiabatic theorem (Theorem 7), if the system is in the ground state of $\mathcal{H}(0)$, and $\mathcal{H}(\cdot)$ evolves slowly with time, then it will remain in the ground state of $\mathcal{H}(t)$ for each t, and therefore will converge to the ground state of $\mathcal{H}_{\mathrm{F}}$ (hence to the optimal solution of the problem). We note in passing that the quantum equivalent of the jump over the local hills in the simulated annealing framework is now the quantum tunnelling through the hill.

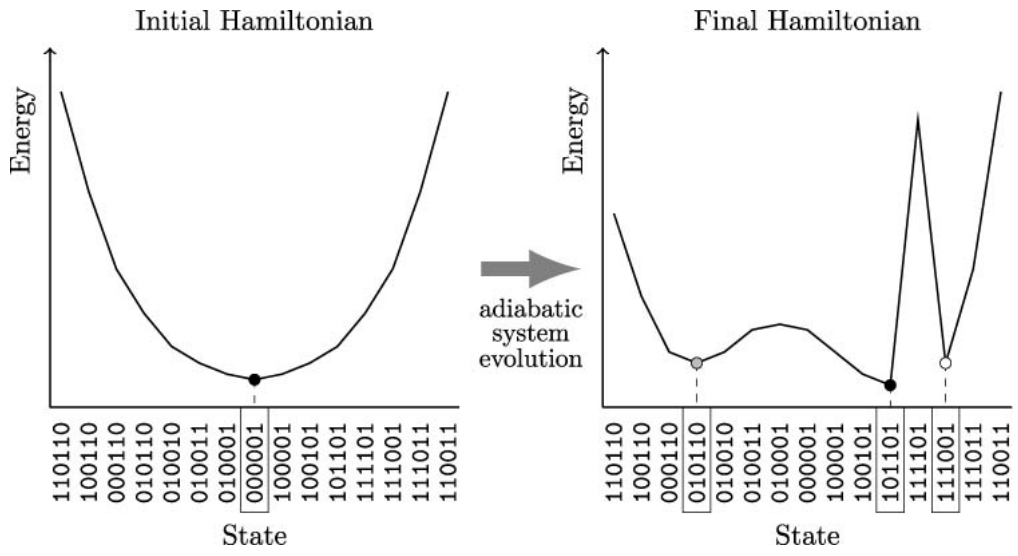


Figure 2.2: Schematic illustration of solving optimisation problems with quantum annealing. The objective is to keep the system in the ground state of $\mathcal{H}(t)$ through adiabatic evolution.

Figure 2.2 illustrates the practical application of quantum annealing to solving optimisation problems. We start by putting the system in the ground state of some easy-to-prepare transverse field Hamiltonian $\mathcal{H}_0$. This initial Hamiltonian is then slowly (adiabatically) transformed into the final longitudinal field Hamiltonian $\mathcal{H}_{\mathrm{F}}$. If the system stays in the ground state of the local Hamiltonian throughout the quantum annealing process, then the readout will give us the optimal configuration of the binary decision variables (qubit values) that correspond to the

global minimum of the objective function encoded in the final Hamiltonian.

Recall the following definition of the Hamming distance between two sets of (bit) strings:

Definition 8. Let a := $(a_1,\ldots,a_n)$ and b := $(b_1,\ldots,b_n)$ denote two bit strings in $\{0,1\}^n$. The Hamming distance between a and b is defined as

$$\sum_{i=1}^{n} |a_i - b_i|.$$

There are two important mechanisms in finding the global minimum: thermal annealing and quantum tunnelling. In Figure 2.2, the final Hamiltonian has a global minimum at state 101101 and two local minima at states `010110` and `111001`. Although the energy of both local minimum states is the same, the Hamming distances from the global minimum are different. The Hamming distance between state `010110` and the global minimum state `101101` is 5: it is necessary to flip 5 bits to get from this local minimum state to the global minimum. The Hamming distance between state `111001` and the global minimum is 2. However, it is more difficult to get from state `111001` to the global minimum state `101101` because they are separated by the tall energy barrier, while getting from state `010110` to the global minimum is straightforward: a series of sequential bit flips over a relatively low and flat energy barrier can be achieved with the help of thermal annealing. Fortunately, we also have a quantum tunnelling effect that allows the system to go through the tall but narrow energy barriers. Flipping 5 bits (wide barrier) in one go could be a challenging task for the quantum tunnelling but flipping 2 bits

(narrow barrier) is something that can happen with sufficiently high probability.



Quantum annealing is a practical implementation of the principles of adiabatic quantum computing and can be benchmarked against its classical counterpart – simulated annealing. Quantum annealing derives its power from two sources: thermal annealing and quantum tunnelling. It is the combination of these classical and quantum effects that allows quantum annealing to achieve superior performance.

2.3 Implementations of AQC

How do we build quantum annealers? What is their physical realisation? Can we find a suitable metric that would objectively quantify their performance? Do we observe steady progress in their development? In this section, we provide answers to these questions that are the results of two decades of intensive research.

2.3.1 The short history of quantum annealing

The first major patent was devised by Amin and Steininger [11] for D-Wave Systems, leading to the seminal paper [151]. Since then, numerous works investigated the value of D-Wave computers, in particular McGeoch and Wang [211], who proved significant speedup of specific problems on one of the earlier D-Wave processors (D-Wave Two with 512 qubits). More experiments followed this early research with two results that, arguably, stand out from many other interesting findings.

First, the contribution of quantum tunnelling to the performance of D-Wave quantum annealers has been established through experiments on D-Wave Two [38]. Multi-qubit tunnelling has been observed and has been experimentally shown to play a computational role in a programmable quantum annealer.

Then an 8-order of magnitude speedup has been established on a 1,152-qubit D-Wave 2X processor relative to the classical benchmark (simulated annealing) for a crafted problem designed to have tall and narrow energy barriers separating local minima by Denchev *et al.* [83]. For instances with 945 variables, the D-Wave 2X quantum annealer achieved time-to-99%-success-probability that is 100 million times faster than simulated annealing running on a single processor core.

Additionally, a fully connected graph problem has been addressed with the help of forward and reverse quantum annealing performed on the more recent 2,048-qubit D-Wave 2000Q processor by Venturelli and Kondratyev [175, 296]. Chapter 3 provides a detailed description of this use case.

At the time of writing, the most advanced D-Wave machine is the 5,760-qubit Advantage system, which also boasts lower noise and much better qubit connectivity – a physical qubit can be directly connected with up to 15 other physical qubits in comparison with D-Wave 2000Q where a physical qubit could only be connected with at most 6 other physical qubits [92]. Therefore, logical qubits that represent binary variables consist of shorter and more stable chains of physical qubits. For example, for the problem size $N = 64$ (which corresponds to the largest fully connected graph that can be embedded on D-Wave 2000Q), the chain length is 17 for the D-

Wave 2000Q *Chimera* graph but only 7 for the Advantage *Pegasus* graph.

Interestingly, quantum computing algorithms prompted intensive improvement of their classical counterparts and recent developments in optimised classical algorithms proved [259] to be able to at least match D-Wave machines.

2.3.2 Inter-generational comparison of D-Wave quantum annealers

The technological progress in the development of D-Wave quantum annealers has been investigated by Pokharel *et al* [235]. The performance of four generations of D-Wave quantum annealers has been studied on the task of solving an identical ensemble of a parameterised family of scheduling problems. These problems are NP-hard and find numerous practical applications.

The performance of quantum annealers was measured by a metric known as Time-to-Solution (TTS). In benchmarking studies, the data collected from multiple runs of the quantum annealer was used to calculate the probability of finding the ground state solution for the given configuration of adjustable parameters. This probability is given by the expression

$$p = \frac{\text{Number of ground state solutions}}{\text{Total number of runs}}.$$

The equivalent TTS is defined as the expected time to obtain the ground state solution at least once with α success probability and is computed as

$$\mathrm{TTS} = t_{\mathrm{run}} \frac{\log(1-\alpha)}{\log(1-p)},$$

where t_{run} is the annealing time for a single run of the quantum annealer, and α is by default taken equal to 99%.

Since scheduling problems are NP-hard, TTS should scale exponentially with the problem size N in the asymptotic limit as shown by (2.2) with $\gamma = 1$. It is a question of what value of parameter β (the scaling exponent) would best fit the experimental results

$$\mathrm{TTS} = T_0 \exp(\beta N),$$

for some constant $T_0 > 0$. As established in [235], the scaling exponent ranges from 1.01 for D-Wave Two to 0.17 for D-Wave Advantage.

To illustrate the magnitude of this improvement let us set $N = 10$ and TTS = 100 microseconds for both cases ($\beta = 1.01$ and $\beta = 0.17$). This produces the following impressive results for TTS as a function of N as shown in Table 2.2:

N	TTS ($\beta = 1.01$)	TTS ($\beta = 0.17$)
10	100 microseconds	100 microseconds
15	16 milliseconds	234 microseconds
20	2.4 seconds	547 microseconds
25	6.3 minutes	1.3 milliseconds
30	16.5 hours	3.0 milliseconds

Table 2.2: TTS as a function of problem size, N, for different values of scaling exponent.

2.3.3 Physical realisations of quantum annealers

Being adiabatic, the quantum annealer is a device that works by first specifying an initial Hamiltonian with an easily computable ground state, and then evolving it slowly to the final problem Hamiltonian. On the hardware side, to ensure quantum coherence, the system lives in a closed environment, away from external magnetic sources, and is kept at a very low temperature. The states of the system are viewed as superpositions of qubits, each represented as a superconducting loop in such a way that the state of the qubit is defined as the electric current in the loop. During computation, the direction of the current is unknown, but when the system decoheres (due to observation or noise), it becomes known. The direction of the spin of each qubit is physically represented by a magnetic field applied to the loop, acting as a noise that may stir the qubit from its current spin. The qubits are also connected to each other via ferromagnetic (antiferromagnetic) couplings forcing their spins in the same or in opposite directions.

The D-Wave quantum annealers rely on the Ising formulation of the Hamiltonian. This is justified by the fact that Barahona [24] showed that finding the ground state of an Ising spin glass is NP-hard. It means that any NP-complete problem can be reduced to an Ising spin glass problem with polynomial overhead (see Section 2.1). Mathematically, the optimisation problem is formulated as the minimisation of the cost function

$$L_{\text{Ising}}(\mathrm{s}) = \sum_{i=1}^{N} g_i s_i + \sum_{i=1}^{N} \sum_{j=i+1}^{N} J_{ij} s_i s_j,$$

where s := (s_1,…,s_N) is a vector of binary decision variables (classical spin variables taking values {−1,+1}), and $(g_i)_{i=1,\ldots,N}$ and $(J_{ij})_{1\le i<j\le N}$ are coefficients encoding the parameters of the optimisation problem.

According to the quantum mechanical description of a spin, we need to replace the classical spin variables with the corresponding operators – the Pauli operators σ_x, σ_y and σ_z (see Section 6.3.3 for their matrix representations). The *problem* or *final* Hamiltonian then takes the following Ising form [147]:

$$\mathcal{H}_F = \sum_{i=1}^{N} g_i \sigma_z^i + \sum_{i=1}^{N} \sum_{j=i+1}^{N} J_{ij} \sigma_z^i \sigma_z^j,$$

where g_i is the bias applied to qubit i and J_{ij} is the coupling between qubits i and j.

The *initial* Hamiltonian has the form

$$\mathcal{H}_0 = \sum_{i=1}^{N} \sigma_x^i,$$

where the operator σ_x (see Section 6.3.3) is the quantum NOT gate that flips the qubit state:

$$\texttt{NOT}\,|0\rangle = |1\rangle \quad \text{and} \quad \texttt{NOT}\,|1\rangle = |0\rangle\,.$$

Recalling the quantum states $|+\rangle$ and $|-\rangle$ defined in (1.2.3), we have

$$\text{NOT}|+\rangle = \frac{1}{\sqrt{2}}\text{NOT}\big(|0\rangle + |1\rangle\big) = \frac{1}{\sqrt{2}}\big(|1\rangle + |0\rangle\big) = |+\rangle$$

$$\text{NOT}|-\rangle = \frac{1}{\sqrt{2}}\text{NOT}\left(|0\rangle - |1\rangle\right) = \frac{1}{\sqrt{2}}\left(|1\rangle - |0\rangle\right) = -|-\rangle,$$

so that $|+\rangle$ and $|-\rangle$ are the two eigenstates.

The eigenstate of the initial Hamiltonian (2.3.3) is the equal superposition of the states $|0\rangle$ and $|1\rangle$ of all individual qubits:

$$\left[\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right]^{\otimes N}.$$

The Hamiltonian $\mathcal{H}_0$ is the *transverse field* Hamiltonian. Its role is to create disorder and to prevent spins from aligning with each other (along the z-axis). The Hamiltonian $\mathcal{H}_F$ is the *longitudinal field* Hamiltonian. At the end of the annealing process, when the transverse field and the σ_x terms go to zero, spins should be aligned either parallel or anti-parallel with each other along the z-direction (depending on the values of the corresponding coupling factors and their individual biases).

The architecture of D-Wave quantum annealers is based on a particular graph (the *Chimera* graph in the case of D-Wave 2000Q or the *Pegasus* graph in the case of the Advantage system) that realises a particular connectivity pattern between the physical qubits. These graphs are not fully connected as mentioned earlier, which means that a concrete optimisation problem to be solved on the quantum annealer must first be transformed (*embedded*) into a graph. We refer the reader to [63, 64] for a theoretical description of these embeddings. Here, we only present a schematic rendering of the *Chimera* graph and show how logical qubits can be constructed from the chains of multiple physical qubits.

Apart from D-Wave, Steffen [273] built an adiabatic quantum computation using nuclear magnetic resonance, available at room temperature, based on the discrete-time approximation of the quantum adiabatic theorem. Another nuclear magnetic resonance implementation was developed in [305] for integer factoring.

Finally, it is necessary to mention a perspective approach to building analog quantum computers based on the neutral atoms technology [131, 163]. Neutral atom quantum annealers have potential to outperform quantum annealers built using other technologies and to bridge the gap between NISQ and fault-tolerant advantage.

2.3.4 Chimera graph and embedding of the logical qubits

The *Chimera* graph topology [76, 91] has a recurring structure of a bipartite graph, called a *unit cell*, as shown in Figure 2.3. The unit cell consists of two groups of four qubits each with pairwise connections between qubits from different groups and no connections between qubits from the same group. Thus, each unit cell graph consists of 8 vertices (physical qubits) and 16 edges (connections between physical qubits). These physical qubits (superconducting loops) are connected via *internal couplers*. Connections between unit cells are achieved via *external couplers*.

Each physical qubit in a *Chimera* graph is connected to six other physical qubits (via four internal couplers and two external couplers). This puts severe limitations on the type of problems that can be embedded in the graph if we would like to establish a one-to-one mapping between the binary decision variable (logical qubit) and the corresponding physical qubit. However, the

effective connectivity between logical qubits can be improved if the logical qubit can be represented by a chain of physical qubits. This would allow us to reach distant corners of the graph and solve the fully connected graph problems. Figure 2.3 displays an example of a qubit chain formed by qubits A, B, C and D (dark shaded qubits connected by bold lines). While individual qubits have only six connections, the qubit chain A-B-C-D has 18 external connections.

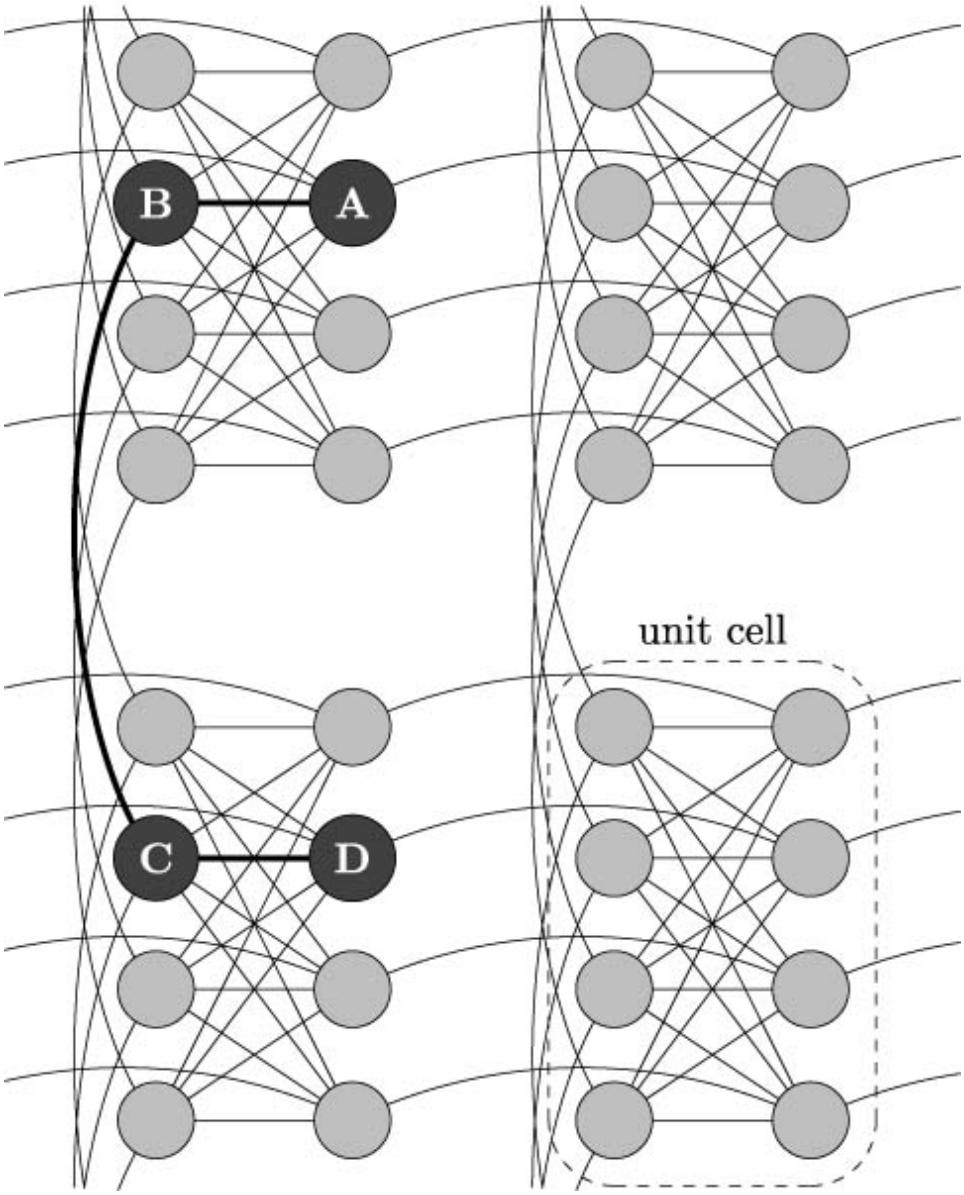
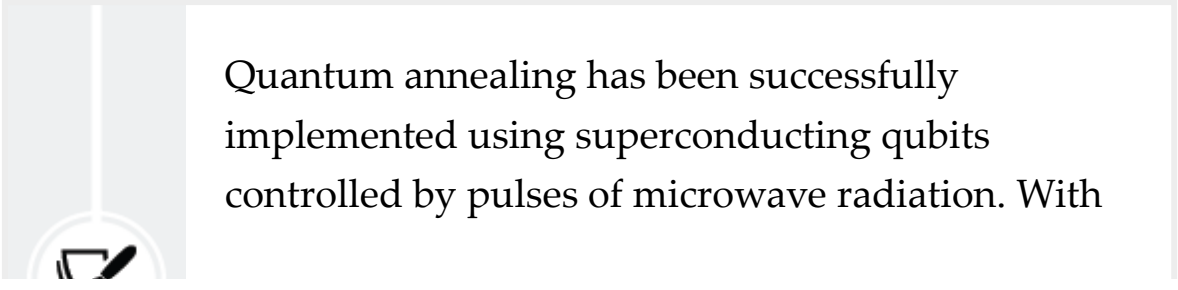


Figure 2.3: Chimera graph. The figure displays a cropped view of four unit cells. Individual physical qubits are shown as circles, internal couplers are shown as straight

lines, and external couplers are shown as curved lines. Qubits A, B, C and D form a qubit chain (logical qubit).

To ensure that qubits A, B, C and D operate as a single logical qubit, we need to make connections between them strong enough such that their states are aligned at all times and they would flip simultaneously if the logical qubit is flipped. Since now they represent a single binary decision variable (logical qubit), the corresponding bias g (in (2.3.3) and (2.3.3)) will be shared equally across all four physical qubits in the chain.

We expect all physical qubits in the chain to have the same value after measurement but this is not always the case. When the qubit chain becomes too long with too many internal and external couplers between the physical qubits, it is likely that the chain would be broken at some point. This will be seen as a disagreement between the physical qubits forming the chain – some of them will be measured as −1 and some as +1. The solution is to apply the *majority voting* rule: the value of a logical qubit is given by the mode of all constituent physical qubit values. For example, if physical qubits A, B and C are measured as −1 and qubit D is measured as +1, the value of the logical qubit is assumed to be −1. The longer the chain, the easier it is to break it. This is why the improved connectivity of the new *Pegasus* graph in comparison with the old *Chimera* graph that results in shorter chains for the fully connected graph problems leads to a significant reduction in TTS.



Quantum annealing has been successfully implemented using superconducting qubits controlled by pulses of microwave radiation. With



significant progress in improving connectivity and scaling up the systems observed over the last several years, quantum annealers have demonstrated their potential as powerful optimisers.

2.4 Universality of AQC

Quantum computing has the theoretical advantage of being able to represent an exponential number of states at once, thereby proving exponential speedup compared to classical algorithms. The historically important examples are Shor's integer factorisation [265] and Grover's database search [117]. On a practical level though, it remains unclear whether this speedup is actually within reach, in particular for NP-hard problems.

Adiabatic quantum computing is a completely different paradigm to gate model quantum computing – although they bear some resemblance – and rests upon the quantum adiabatic theorem. Contrary to gate model quantum computing, AQC has inherent fault tolerance, as proved in [61]. Since its performance depends on the spectral gap from the adiabatic theorem, environmental decoherence can be minimised by running an AQC device at temperatures much lower than this gap.

Key insights connecting gate model quantum computing and its adiabatic version were provided by Aharonov *et al.* [7], and by Farhi *et al.* [97]. The former proved that AQC can simulate any algorithm with only polynomial overheads compared to gate model quantum computing while the latter showed that gate model quantum computers can reproduce any AQC computation. These two results therefore imply that AQC and gate model

quantum computing are in fact polynomially equivalent. The proof in [7] assumes that the initial and final Hamiltonians in the adiabatic formulation are so-called *k*-local, meaning that they can be represented as sums of independent Hamiltonians each acting on *k* qubits only. Existing AQC algorithms rely on a certain number of assumptions regarding the final Hamiltonian, the main one being that the latter has null off-diagonal elements with a 2-local connectivity structure, as in the case of the D-Wave quantum annealer. Unfortunately, this currently falls outside the scope of [7], leaving a question mark about the universality of this setup. Other AQC algorithms exist, in particular involving stoquastic Hamiltonians [45] with real and non-positive off-diagonal elements or even more restrictive considerations [35].

The standard background for comparing algorithms – classical, quantum, or adiabatic – is that of complexity classes. We shall not delve into too many technicalities here though, but refer to [197] for details. There, the mathematical formulations of Ising problems (the standard problems solved by AQC) for a large number of NP-complete and NP-hard problems are presented, including precise formulations of Karp's 21 fundamental NP-complete problems [155], a perfect playground to compare quantum algorithms.

Summary

At the beginning of this chapter, we introduced several basic complexity classes and discussed their relationships. The time needed to solve NP-hard problems grows exponentially with the problem size, which is a strong motivation for exploring alternative approaches, such as analog adiabatic quantum

computing. Even though quantum optimisers also solve NP-hard combinatorial optimisation problems in time that grows exponentially with problem size, the prefactor in the exponent may be smaller than for known classical algorithms. Additionally, we can expect to find an approximate solution in polynomial time, which provides a strong motivation for many practical applications of adiabatic quantum computing.

We then introduced the principles of AQC based on the adiabatic quantum theorem. The physical realisation of AQC – quantum annealing – has been contrasted with its classical counterpart – simulated annealing. We highlighted two main sources of computational power of quantum annealing: thermal annealing, which helps us find the minima of the objective function separated by wide but moderate energy barriers; and quantum tunnelling, which allows us to penetrate through the narrow and steep energy barriers. Their combination ensures efficient convergence to the global minimum of the objective function for many practical problems.

Having established the principles of AQC and the corresponding theoretical framework, we looked at the practical implementation of AQC in the form of quantum annealers based on superconducting qubits. We also introduced the Ising model, which provides a mathematical description of the problems solvable on quantum annealers.

We learned about the importance of the quantum chip layout (graph) and how several physical qubits can be coupled together to form a chain representing a single logical qubit. Finally, we touched on the universality of AQC.

In the next chapter, we will learn how quantum annealing can be used to solve practical NP-hard optimisation problems such as discrete portfolio optimisation.

Join our book's Discord space

Join our Discord community to meet like-minded people and learn alongside more than 2000 members at:
https://packt.link/quantum



3
Quadratic Unconstrained Binary Optimisation

Undoubtedly, Quadratic Unconstrained Binary Optimisation (QUBO) is a flagship use case of quantum annealing. We only need to have a closer look at the name of this class of optimisation problems to see why:

- Quantum annealers operate on binary spin variables. It is straightforward to perform mapping between binary decision variables (represented by the logical qubits) and spin variables.
- The objective functions of quadratic optimisation problems have only linear and quadratic terms. This significantly simplifies the models and allows their embedding on existing quantum annealing hardware.
- Unconstrained optimisation means that although QUBO allows us to specify conditions that must be satisfied, they are not hard constraints. The violation of constraints is penalised through the additional terms in the QUBO objective function, but it is still possible to find solutions that violate specified constraints.

All these features make QUBO problems solvable on quantum annealers. At the same time, the QUBO formulation exists for many important NP-hard combinatorial optimisation problems, such as graph partitioning, job-shop scheduling, binary integer linear programming, and many others. This class also includes the discrete portfolio optimisation problem, which we consider in this chapter. We should also mention here some of the recent attempts to address the problem of discrete portfolio optimisation using classical methods, such as the knapsack problem formulation by Vaezi *et al.* [289] and the application of evolutionary search methods such as genetic algorithms

by Anagnostopoulos and Mamanis [12]. Both the knapsack problem with integer weights and genetic algorithms are discussed in this chapter.

3.1 Principles of Quadratic Unconstrained Binary Optimisation

QUBO represents optimisation problems in which a quadratic function of N binary variables, $q_1,\ldots,q_N$, has to be minimised over all possible 2^N assignments of its variables. The function to be minimised is referred to as the *cost function*, and it can be written as

$$L_{\mathrm{QUBO}}(\mathrm{q}) = \sum_{i=1}^{N} a_i q_i + \sum_{i=1}^{N} \sum_{j=i+1}^{N} b_{ij} q_i q_j,$$

where $\mathrm{q} := (q_1,\ldots,q_N) \in \{0,1\}^N$ represents the assignment of N binary decision variables.

A broad class of optimisation problems with many practical applications admits a QUBO formulation [197]. To solve hard QUBO instances in an exact way, known classical algorithms require exponential time (in the problem size defined as the number of binary decision variables, N) [120]. Several approximate classical methods have been devised to reduce the computational cost; however, it is the fast maturing quantum annealing that aspires to demonstrate material speedup on the hardest QUBO instances, such as NP-hard discrete portfolio optimisation problems [175, 296].

3.1.1 QUBO to Ising transformation

A QUBO problem can be easily translated into a corresponding Ising problem solvable on a quantum annealer. The Ising cost function of N spin variables $\mathrm{s} := (s_1,\ldots,s_N) \in \{-1,+1\}^N$ is given by

$$L_{\text{Ising}}(\mathbf{s}) = \sum_{i=1}^{N} g_i s_i + \sum_{i=1}^{N} \sum_{j=i+1}^{N} J_{ij} s_i s_j.$$

The Ising and QUBO models are related through the transformation $s_i = 2q_i - 1$, hence the relationship with (3.1) is

$$J_{ij} = \frac{1}{4} b_{ij} \quad \text{and} \quad g_i = \frac{1}{2} a_i + \frac{1}{4} \left(\sum_{k=1}^{i-1} b_{ki} + \sum_{l=i+1}^{N} b_{il} \right)$$

disregarding an unimportant constant offset for the optimisation.

3.1.2 QUBO problem examples

There are many examples of important QUBO problems that can be directly applied to practical use cases arising in finance. Here, following Lucas [197], we mention a few of them in their traditional formulation.

Number Partitioning

Given a set of N positive numbers $\{n_1,\ldots,n_N\}$, is there a partition of this set of numbers into two disjoint subsets such that the sum of the elements in both sets is the same? For example, we can think about the set $\{n_1,\ldots,n_N\}$ as a collection of assets that must be divided equally between two parties. The Ising formulation of this problem is

$$L(\mathbf{s}) = \left(\sum_{i=1}^{N} s_i n_i \right)^2,$$

where spin variables $(s_i)_{i=1,\ldots,N} \in \{-1,+1\}$ are the decision variables. If there is a solution to the Ising model with $L = 0$, then there is a configuration of spins where the sum of the n_i for the +1 spins is the same for the sum of the n_i for the −1 spins. The number partitioning

problem finds numerous applications in economics and finance, from routing and scheduling problems [71] to signal detection and time series analysis [148].

Graph Partitioning

Consider an undirected graph

$$G = (V, E),$$

where V stands for the set of vertices and E stands for the set of edges, with an even number N of vertices. The task is to partition the set V into two subsets of equal size $N/2$, such that the number of edges connecting the two subsets is minimised. This problem has many applications: finding these partitions can allow us to run some graph algorithms in parallel on the two partitions, and then make some modifications due to the few connecting edges at the end [36]. The spin variables represent the graph vertices, with values +1 and −1 denoting the vertex being in either the {+} set or the {−} set. The problem is solved with the cost function consisting of two components:

$$L(\mathrm{s}) = L_A(\mathrm{s}) + L_B(\mathrm{s}),$$

where

$$L_A(\mathrm{s}) = A \sum_{i=1}^{N} s_i$$

provides a penalty if the number of elements in the {+} set is not equal to the number in the {−} set, and

$$L_B(\mathrm{s}) = B \sum_{(u,v)\in E} \frac{1 - s_u s_v}{2}$$

is a term that provides a penalty for each time that an edge connects vertices from different subsets. If $B > 0$, then we wish to minimise the number of edges between the two subsets; if $B < 0$, we will choose to maximise this number. Should we choose $B < 0$, we must ensure that B is small enough so that it is never favorable to violate the L_A constraint. The graph partitioning problem can be applied to studying clustering in financial markets [271].

Both problems, number and graph partitioning, are NP-hard problems [155].

Binary Integer Linear Programming

Let $q := (q_1,\ldots,q_N)$ be a vector of N binary variables. The task is to maximise $c \cdot q$, for some vector c, given the constraint

$$Sq = b$$

with $S \in \mathcal{M}_{m,N}(\mathbb{R})$ and $b \in \mathbb{R}^m$. Many problems can be posed as binary integer linear programming, for example, a maximisation of profit subject to regulatory constraints [255].

The cost function $L(q)$ corresponding to this problem can be constructed as a sum of two terms, $L(q) = L_A(q) + L_B(q)$, where the first term is

$$L_A(q) = A\sum_{j=1}^{m}\left(b_j - \sum_{i=1}^{N} S_{ij}q_i\right)^2,$$

for some constant $A > 0$. Note that $L_A = 0$ enforces the constraint $Sq = b$. The second term is

$$L_B(\mathrm{q}) = -B \sum_{i=1}^{N} c_i q_i,$$

with another positive constant $B < A$.

Knapsack with Integer Weights

We have a list of N objects, labelled by indices $i = 1,\ldots,N$, with the weight of each object given by $w_\mathrm{i} \in \mathbb{N}$, and its value given by c_i, and we have a knapsack that can only carry some weight up to $W_\mathrm{max} \in \mathbb{N}$. If q_i is a binary variable denoting whether ($q_\mathrm{i} = 1$) or not ($q_\mathrm{i} = 0$) object i is contained in the knapsack, the total weight in the knapsack is

$$W = \sum_{i=1}^{N} w_i q_i,$$

and the total value is

$$C = \sum_{i=1}^{N} c_i q_i.$$

The task is to maximise C subject to the constraint that $W \leq W_\mathrm{max}$. The knapsack problem finds multiple applications in economics and finance [159, 289].

We introduce a binary variable, y_n, for each $n = 1,\ldots,W_\mathrm{max}$, which is 1 if the final weight of the knapsack is n, and 0 otherwise. As before, the cost function consists of two terms, $L(\mathrm{q}) = L_\mathrm{A}(\mathrm{q}) + L_\mathrm{B}(\mathrm{q})$, with

$$L_A(\mathrm{q}) = A\left(1 - \sum_{n=1}^{W_\mathrm{max}} y_n\right)^2 + A\left(\sum_{n=1}^{W_\mathrm{max}} n y_n - \sum_{i=1}^{N} w_i q_i\right)^2,$$

which enforces that the weight can only take on one value and that the weight of the objects in the knapsack equals the value we claimed it did, and

$$L_B(\mathrm{q}) = -B\sum_{i=1}^{N} c_i q_i.$$

As we require that it should not be possible to find a solution where L_A is weakly violated at the expense of L_B becoming more negative, we require $0 < B \max_{i=1,\ldots,N} c_i < A$ (adding an item to the knapsack, which makes it too heavy, is not allowed).

Many other famous NP-hard optimisation problems can be solved on quantum annealers. Here, we can mention the Map Colouring Problem [76] and the Job-Shop Scheduling Problem [295], which were successfully solved on the D-Wave quantum annealers. In this chapter, we provide detailed description and analysis of an important finance-related QUBO problem – a discrete portfolio optimisation problem investigated by Venturelli and Kondratyev [175, 296]. This is a hard, fully connected graph problem that can be best addressed using a newly developed technique of reverse quantum annealing.



Many famous NP-hard problems can be solved with practical efficiency on quantum annealers in their QUBO formulation, up to a non-trivial number of variables. This makes quantum annealing a useful tool in dealing with classically hard optimisation problems.

3.2 Forward and Reverse Quantum Annealing

Having defined the QUBO problem, we now review how quantum annealing can act as an efficient solver.

3.2.1 Forward quantum annealing

The quantum annealing protocol, inspired by the adiabatic principle of quantum mechanics detailed in Chapter 2, dictates driving the system from an easy-to-prepare ground state of an initial Hamiltonian $\mathcal{H}_0$ to the unknown low-energy subspace of states of the problem Hamiltonian $\mathcal{H}_F$, ideally to the lowest energy state corresponding to the global minimum of the objective function. This *forward* quantum annealing procedure can be ideally described as attempting to drive the evolution of the time-dependent Hamiltonian

$$\mathcal{H}(t) = A(t)\mathcal{H}_0 + B(t)\mathcal{H}_F,$$

starting from

$$\mathcal{H}_0 = \sum_{i=1}^{N} \left(\sum_{c=1}^{N_c} \sigma_x^{ic} \right),$$

where $\mathcal{H}_0$ is a Hamiltonian describing an independent collection of local transverse fields for each spin of the system (σ_x is the Pauli X spin operator, or the quantum NOT gate, detailed in Section 6.3.3).

In expression (3.2.1), the first sum runs over all *logical qubits* representing binary decision variables in the QUBO formulation of the optimisation problem, and the second sum runs over all *physical qubits* in the chain that represents a logical qubit. The construction of a logical qubit from a chain of physical qubits is explained in Chapter 2 and a sample embedding of the qubit chain on the quantum chip is shown in Figure 2.3 for the *Chimera* graph. No physical qubit can be an element of more than one qubit chain. Assuming the most dense embedding scheme, we have $N \times N_c = K$, where K is the total number of physical

qubits in the quantum chip. However, in practice we have $N \times N_c < K$ for most problems of realistic size that require full connectivity between the logical qubits. For example, in the case of a D-Wave 2000Q *Chimera* graph consisting of $K = 2{,}048$ physical qubits, the maximum number of fully connected logical qubits is 64 and the maximum number of physical qubits in a chain is 17, meaning that $N \times N_c = 1{,}088$.

Figure 3.1 shows how $A(t)$ and $B(t)$ vary over the scale of total *annealing time* τ.

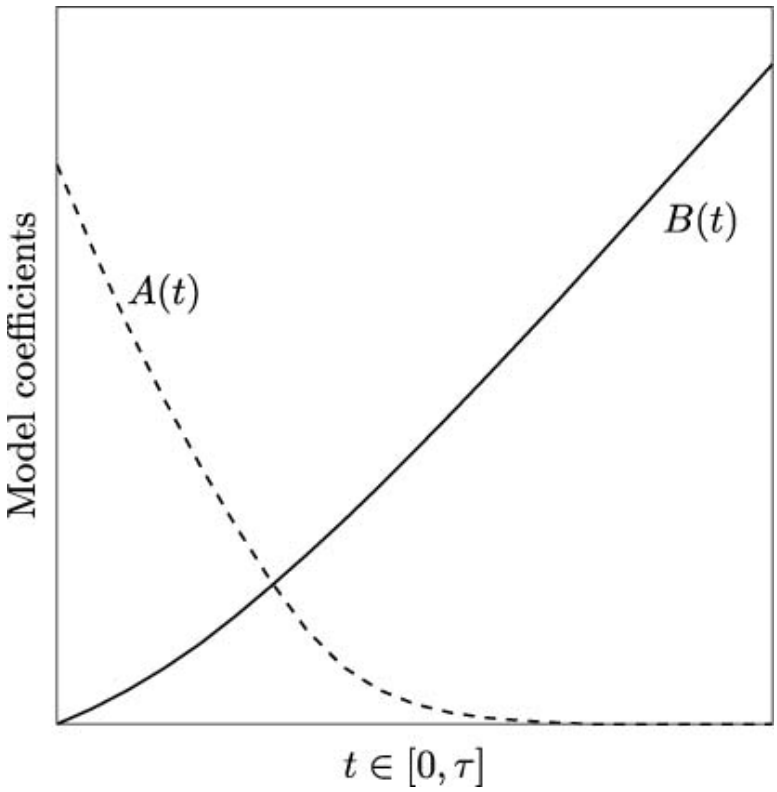


Figure 3.1: Schematic forward annealing schedule. A(t) indicates the strength of the transverse magnetic field and B(t) indicates the strength of the longitudinal magnetic field.

At the end of the annealing run, $A(\tau) = 0$ and the system is projected on the computational basis by a measurement of each qubit magnetisation. The duration of the anneal, τ, is a free parameter, hence it is often useful to define the fractional completion of the annealing schedule $s = t/\tau$.

3.2.2 Reverse quantum annealing

Figure 3.2 illustrates the quantum annealing protocol when the quantum annealer is set to operate as a reverse annealer.

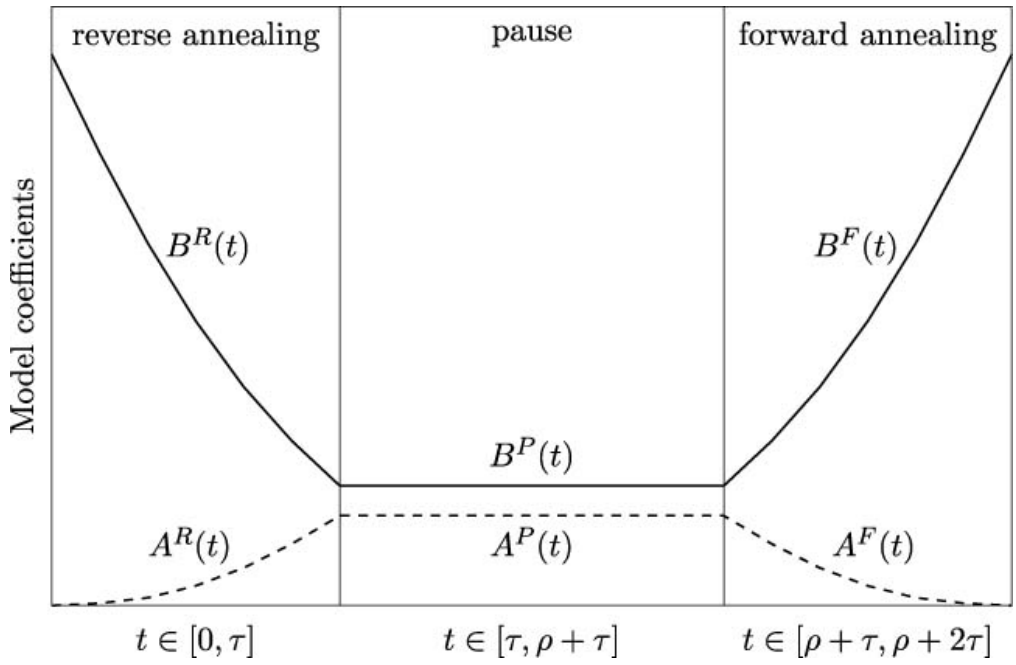


Figure 3.2: Schematic reverse annealing schedule. A(t) indicates the strength of the transverse magnetic field and B(t) indicates the strength of the longitudinal magnetic field.

The system is initialised with $B(0) = \max\{B\}$ and $A(0) = 0$, with spins set to a classical bitstring. The evolution then undergoes an inverse schedule up to a point where the Hamiltonian time-dependence is temporarily paused. With reference to the Hamiltonian evolution in (3.2.1), the transverse field evolution that we program for this protocol is the following three-phase function (analogous equations for $B(t)$):

$$\begin{aligned} A^R(t) &:= A(\tau + (s_p - 1)t) && \text{Reverse Annealing: } t \in [0, \tau], \\ A^P(t) &:= A(s_p \tau) && \text{Annealing Pause: } t \in [\tau, \tau + \rho], \\ A^F(t) &:= A((1 - s_p)(t - \rho) - (1 - 2s_p)\tau) && \text{Forward Annealing: } t \in [\tau + \rho, 2\tau + \rho], \end{aligned}$$

where ρ is the duration of the pause and $s_p \in [0,1]$ indicates the location of the forward schedule where the pause is implemented. The total

duration of the selected reverse anneal protocol is $2\tau + \rho$ as opposed to τ for the forward anneal. While the theory of reverse annealing is just starting to be investigated, the physics rationale of reverse annealing is to be found in the oversimplified idea that, if the system is initialised in a state s corresponding to a local minimum of the objective function, the interplay of quantum and thermal fluctuations might help the state *tunnel* out of the energy trap during the reverse annealing, while the annealing pause (and to some extent the final forward annealing) allows the system to thermalise and relax in the neighborhood of the newfound minimum. The quality of the initial state s is likely to influence dramatically the reverse annealing process. For the portfolio optimisation use case presented in this chapter, a classical greedy algorithm to set s can be used as described in [296].



The combination of reverse quantum annealing with a classical greedy search algorithm has the potential to massively speed up a QUBO solver, thus realising a promising hybrid quantum-classical algorithm.

3.3 Discrete Portfolio Optimisation

The optimal portfolio construction problem is one of the most extensively studied problems in quantitative finance. The Modern Portfolio Theory (MPT) [202] has laid the foundation for highly influential mean-variance portfolio optimisation approach. According to the MPT, a typical portfolio optimisation problem can be formulated as follows. Let N be the number of assets, μ_i be the expected return of asset i, Σ_{ij} be the covariance between the returns for assets i and j, and R be the target portfolio return. Then the decision variables are the weights w_i, i.e., the investment associated with the asset i ($w_i \in \mathbb{R}$). The

standard Markowitz mean-variance approach consists in the constrained, quadratic optimisation problem

$$\min \sum_{i=1}^{N} \sum_{j=1}^{N} w_i w_j \Sigma_{ij}, \quad \text{subject to} \quad \sum_{i=1}^{N} w_i = 1 \quad \text{and} \quad \sum_{i=1}^{N} w_i \mu_i = R.$$

Quadratic problems of this form are efficiently solvable by standard computational means (e.g., quadratic programming with linear constraints) if the covariance matrix is positive definite. However, related discrete portfolio optimisation problems (with discrete weights w_i) are much harder to solve. In fact, they are known to be NP-complete [158].

Interestingly, the problem can also be cast into an unconstrained quadratic optimisation problem, which is a suitable formulation for quantum annealers [116, 205, 234, 251]. The problem we are trying to solve here is a construction of the optimal portfolio from the universe of assets with known characteristics, such as asset returns, volatilities, and pairwise correlations. A stylised portfolio optimisation problem consists of selecting M assets from the universe of N investable assets. These M assets should ideally be the best possible choice according to some criteria.

The scenario we target is a Fund of Funds portfolio manager who is facing a task of selecting the best funds that follow particular trading strategies in order to maximise the risk-adjusted returns according to some model [175] with a constraint that the assets are selected with equal preference weights [82]. Should we want to generalise the portfolio with larger allocation to a given asset, we could allow for multiples of the reference weight by cloning an asset and treating it as a new one.

3.3.1 QUBO encoding

The task of encoding the relationship among the choices of M funds (*without replacement*) from the universe of N funds can then be formulated as a quadratic form:

$$L(\mathrm{q}) = \sum_{i=1}^{N} a_i q_i + \sum_{i=1}^{N} \sum_{j=i+1}^{N} b_{ij} q_i q_j,$$

where $q_\mathrm{i} = 1$ means that asset i is selected and $q_\mathrm{i} = 0$ means that asset i is not selected. The task is then to find a configuration of $\mathrm{q} := (q_1,\ldots,q_\mathrm{N})$ that minimises $L(\mathrm{q})$ subject to satisfying the cardinality constraint (i.e., selection of exactly M assets). A common way to deal with the cardinality constraint would be to add a term $L_\mathrm{penalty}(\mathrm{q})$ to the cost function given by (3.3.1) such that the unsatisfying selections would be penalised by a large value $P \gg 1$, which would force the global minimum to be such that $\sum_{\mathrm{i}=1}^{\mathrm{N}} q_\mathrm{i} = M$:

$$L_\mathrm{penalty}(\mathrm{q}) = P\left(M - \sum_{i=1}^{N} q_i\right)^2.$$

The coefficients $(a_\mathrm{i})_{1\le \mathrm{i} \le \mathrm{N}}$ reflect asset attractiveness on a standalone basis and can be derived from the individual assets' expected risk-adjusted returns. Assets with large expected risk-adjusted returns should be rewarded with negative values of a_i, while assets with small expected risk-adjusted returns should be penalised with positive values of a_i. The coefficients $(b_\mathrm{ij})_{1\le \mathrm{i} < \mathrm{j} \le \mathrm{N}}$ reflect pairwise diversification penalties (positive values) and rewards (negative values). These coefficients can be derived from the pairwise correlations.

The minimisation of the QUBO cost function given by (3.3.1) and (3.3.1) should optimise the risk-adjusted returns by the use of the metrics of the Sharpe ratio. The Sharpe ratio (excess asset return measured in the asset volatility units) is calculated as $(r - r_0)/\sigma$, where r

is the expected annualised asset return, r_0 is the applicable risk-free interest rate, and σ is asset volatility (annualised standard deviation of the asset returns). The higher the fund's Sharpe ratio, the better the fund's returns have been relative to the risk it has taken on. Volatility can be estimated as the historical annualised standard deviation of the net asset value returns (per share). Expected return can be either estimated as the historical return on fund investment or derived independently by the analyst/portfolio manager taking into account different considerations about the future fund performance.

3.3.2 The coarse-grained encoding scheme

Instead of using the raw real numbers obtained from the financial data for the QUBO coefficients, we opt to coarse-grain the individual funds Sharpe ratios and their mutual correlations down to integer values by grouping intervals in buckets (the sample mapping schemes are shown in Table 3.1). By using bucketed values, we define a scorecard, which is loosely based on the past fund performances but can be easily adjusted by portfolio managers according to their personal views and any new information not yet reflected in the funds reports.

Sharpe ratio bucket	Coefficient a_i		
Equally spaced buckets,	Mapping scheme:		
from worst to best	A	B	C
1st	15	25	30
2nd	12	16	24
3rd	9	9	18
4th	6	4	12
5th	3	1	6
6th	0	0	0

7th	−3	−1	−6
8th	−6	−4	−12
9th	−9	−9	−18
10th	−12	−16	−24
11th	−15	−25	−30

Correlation bucket	Coefficient b_{ij}		
	Mapping scheme:		
	D	E	F
$-1.00 \leq \varrho_{ij} < -0.25$	−5	−9	−10
$-0.25 \leq \varrho_{ij} < -0.15$	−3	−4	−6
$-0.15 \leq \varrho_{ij} < -0.05$	−1	−1	−2
$-0.05 \leq \varrho_{ij} < 0.05$	0	0	0
$0.05 \leq \varrho_{ij} < 0.15$	1	1	2
$0.15 \leq \varrho_{ij} < 0.25$	3	4	6
$0.25 \leq \varrho_{ij} \leq 1.00$	5	9	10

Table 3.1: Specification of the sample QUBO coefficients from NAV time series market data.

The choice of QUBO coefficients as small integer numbers is dictated by the technical realisation of the existing quantum annealer architecture (precision of the superconducting chip circuitry). Within this restriction, the portfolio manager may choose any linear or non-linear scale for QUBO coefficients. For example, the quadratic mapping scheme B strongly penalises low Sharpe ratio funds and strongly rewards high Sharpe ratio funds. The linear mapping schemes A and C distinguish better between funds with average performances. Similarly, the mapping scheme E penalises large positive correlations and

rewards large negative correlations stronger than the mapping scheme D.

3.3.3 Construction of the instance set for numerical experiments

The instance set used for our case study is obtained by simulating asset values with the help of correlated geometric Brownian motion processes with a constant correlation ρ, drift μ, and log-normal volatility σ. The specific values of these parameters were derived from a wide range of fund industry researches (see [79] for the Sharpe ratio distributions) and, therefore, can be viewed as representative of the industry. The simulation time horizon was chosen to be one year and the time step was set at one month.

Every simulated (or "realised") portfolio scenario consists of 12 monthly returns for each asset. From these returns, we calculated the total realised return and realised volatility for each asset (which, obviously, differ from their expected values μ and σ) and for the portfolio as a whole. We also calculated realised pairwise correlations between all assets according to the input uniform correlation ρ. Finally, we calculated individual assets and portfolio Sharpe ratios. For reference, with $\rho = 0.1$, $\mu = 0.075$, $\sigma = 0.15$, and the constant risk-free interest rate set at $r_0 = 0.015$, the expected Sharpe ratio for each asset in the portfolio is 0.4. The expected Sharpe ratio for the portfolio of N assets is significantly larger due to the diversification and low correlation between the assets, e.g., for a 48-asset portfolio we would expect Sharpe ratio values from 0.5 (25th percentile) to 2.1 (75th percentile) with a mean around 1.4.

3.3.4 Classical benchmark – Genetic Algorithm

We selected Genetic Algorithm (GA) as a classical benchmark heuristic, which is a popular choice for solving hard combinatorial optimisation

problems. GAs are adaptive methods of searching a solution space by applying operators modelled after natural genetic inheritance and simulating the Darwinian struggle for survival. There is a rich history of GA applications to portfolio optimisation problems, including [172, 179] recently.

In the case of portfolio optimisation, the solution (chromosome) is a vector q := (q_1,…,q_N) consisting of N elements (genes) that can take binary values in {0,1}. The task is to find a combination of genes that minimises the cost (fitness) function L(q). Due to the solution being represented by a relatively short bit string, we do not use the crossover recombination mechanism as it provides very little value in improving the algorithm convergence. Algorithm 1 provides a detailed description of the GA procedure.

Algorithm 1: GA – portfolio optimisation with cardinality constraint

Result: Optimal portfolio.

1: Generate L initial solutions by populating solutions with random draws from a pool of possible element values $\{0, 1\}$, with the restriction that "1" is assigned to the values of exactly M elements and "0" is assigned to the values of remaining $N - M$ elements.

2: Evaluate the cost (fitness) function for each solution.

3: Rank the solutions from "best" to "worst" according to the cost function evaluation results.

for i **from** *0* **to** *number of iterations* $-$ *1* **do**

a) Select K best solutions from the previous generation and produce L new solutions by randomly swapping the values of two elements with opposite values. With $L = mK$, every one of the "best" solutions will be used to produce m new solutions.

b) Evaluate the cost (fitness) function for each solution.

c) Rank the solutions from "best" to "worst" according to the cost function evaluation results.

end

The optimal values of the parameters L and K depend on the problem size and the specific QUBO coefficient mapping scheme, and can be found through trial and error. The objective here is to achieve the target convergence with the smallest number of objective function calls.

Our first task is to verify that the proposed mapping schemes are sensible in the sense that the minimisation of the objective functions (3.3.1) and (3.3.1) indeed leads to the construction of

optimal portfolios. One possible approach to the problem of selecting M best assets from the universe of N investable assets is to select M individually best assets according to their individual Sharpe ratios. This approach would ignore the potential negative impact on diversification due to a probable large positive correlation between some of individually best assets, and there is no reason to believe that such a portfolio would be optimal. Therefore, we should demand that the optimal portfolio constructed by minimising $L(q) + L_{penalty}(q)$ should outperform the portfolio of M individually best assets.

For example, we can compare the results for the optimal 24-asset portfolio selected from the universe of 48 assets (for 10,000 portfolio instances simulated with $\rho = 0.1$, $\mu = 0.075$, $\sigma = 0.15$; as before, we assumed a constant risk-free interest rate $r_0 = 0.015$). Table 3.2 displays the Sharpe ratio distribution statistics obtained for the sample QUBO coefficient mapping schemes provided by Table 3.1 [296]. The results are presented in the format: **mean** (25th percentile; 75th percentile).

		Mapping schemes for b_{ij}		
		D	E	F
Mapping schemes for a_i	A	4.7 (2.5; 6.4)	4.5 (2.1; 6.1)	4.0 (1.7; 5.5)
	B	4.8 (2.7; 6.5)	4.3 (2.0; 5.9)	4.2 (2.0; 5.8)
	C	5.0 (3.0; 6.7)	4.8 (2.6; 6.3)	4.6 (2.5; 6.1)

Table 3.2: Sharpe ratio distribution statistics for different mapping schemes.

For the portfolio of individually best assets, the Sharpe ratio distribution statistics look as follows:

mean: 3.8 25th percentile: 2.6 75th percentile: 4.7

On average, the Sharpe ratio of the optimal portfolio is larger than that of the portfolio of individually best assets by 0.8, although some QUBO coefficient mapping schemes produce better results than others. Figure 3.3 illustrates a better performance of the optimal portfolio found through the minimisation of the cost function $L(q) + L_{penalty}(q)$ in comparison with the portfolio consisting of 24 individually best assets for the mapping schemes A and D.

Our second task is to understand how the time to solution scales with the problem size and whether quantum annealing can demonstrate a material speedup compared with the classical algorithm. It would be interesting to see what happens when we remove the constraint on the number of assets in the optimal portfolio. The portfolio optimisation results shown in Figure 3.3 were obtained for $M = N/2$, which is arguably the hardest combinatorial problem with a constraint on the number of assets, based on the size of the search space [175]. From a brute force approach perspective, the problem becomes even harder computationally if we remove this constraint due to the solution space growing with N as 2^N instead of $\frac{N!}{M!(N-M)!}$.

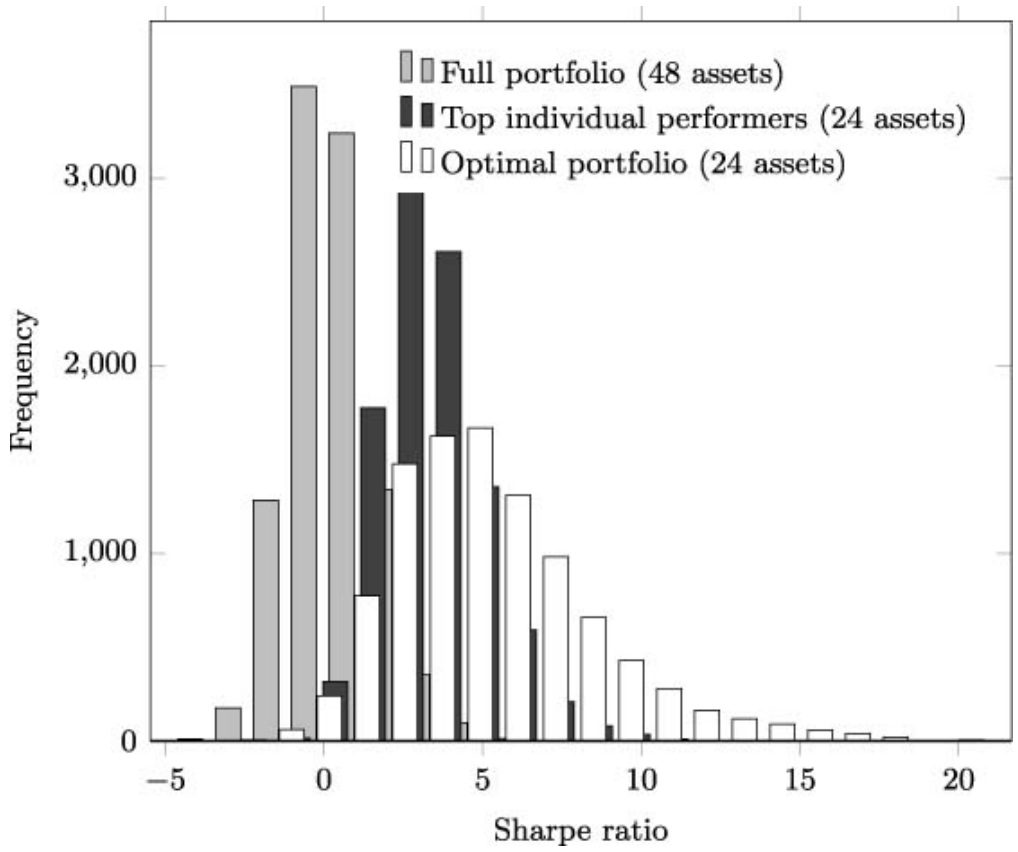


Figure 3.3: Sharpe ratio histogram (QUBO coefficients mapping schemes A and D). M = N/2.

The removal of the cardinality constraint is also warranted by the fact that a large energy scale P in expression (3.3.1) is typically associated with precision issues connected to the analog nature of the quantum annealing machine and the fact that there is a physical maximum to the energy that can be controllably programmed on local elements of the quantum chip. However, several hybrid quantum-classical strategies can be put in place to overcome this limitation.

For instance, we observe that shifting artificially the Sharpe ratio values by a constant amount $\pm\Delta$ (and adding buckets according to the prescription chosen, e.g., Table 3.1) will essentially amount to forcing the ground state solution of the unconstrained problem to have more or less a desired number of assets selected. Hence, while not solving the same problem, we could imagine a solver of a similarly constrained problem that will enclose the quantum annealing runs in a classical loop that checks for the number of selected assets $m(\Delta)$ in the best

found solution with $\Delta = 0$, then increases or decreases the individual desirability of the assets according to whether m is larger or smaller than M and runs again until $m(\Delta) = M$ for $\Delta = \Delta^\star$. While in this case it is an approximation of the original problem, this sort of hybridisation scheme is not uncommon for quantum-assisted solvers [285] and the number of expected rounds of runs should scale proportionally to $\log_2(\Delta^\star)$ as per a binary search, introducing a prefactor over the time-to-solution complexity that should stay manageable. Other hybrid approaches could also be put forward to deal with the constraint, such as fixing some asset selections in pre-processing via sample persistence [154].

As per the preceding arguments, in our benchmark case study, we focus on running the unconstrained problem, setting $\Delta = 0$. Table 3.3 provides the characteristics of the benchmark instance set [296]. For a problem of the given size, the table reports the median number of assets in the optimal portfolio (and the minimum and the maximum in parenthesis), over 30 instances, for the unconstrained portfolio optimisation problem.

Problem size N	Number of assets in the optimal portfolio (unconstrained problem)
42	16 (−7,+6)
48	17 (−6,+5)
54	19 (−7,+12)
60	23 (−13,+15)

Table 3.3: Benchmark instance set characterisation.

3.3.5 Establishing quantum speedup

The aim is to solve representative portfolio instances at the limit of programmability for the D-Wave 2000Q quantum annealer. D-Wave 2000Q features 2,048 physical qubits; however, due to the limited connectivity of the D-Wave 2000Q *Chimera* graph, we can embed a maximum of 64 logical binary variables on a fully connected graph. Practically, we limit ourselves by working with up to 60 logical qubits, which means that the largest search space for our benchmarks is around $60!/(30!)^2 \simeq 10^{17}$ if $M = N/2$. This constraint dictates the configuration of the instance set, which consists of 30 randomly generated instances for $N = \{42,48,54,60\}$ assets.

As mentioned in Chapter 2, a common metric to benchmark the performance of non-deterministic iterative heuristics against quantum annealing is the Time-to-Solution (TTS) [250]. The latter is defined as the expected number of independent runs of the annealer in order to find the ground state with probability (confidence level) $\alpha \in (0,1)$:

$$\mathrm{TTS} = t_{\mathrm{run}} \frac{\log(1-\alpha)}{\log(1-p)},$$

where t_{run} is the running time elapsed for a single run – either τ for forward annealing (see Section 3.2.1), or $2\tau + \rho$ for reverse annealing (see Section 3.2.2) – and p is the probability of finding the optimum of the objective function in that single shot.

Figure 3.4 displays the TTS results for the GA, the forward QA solver, and the reverse QA solver for the unconstrained portfolio optimisation problem encoded using the mapping schemes A and D [296]. In the figure, the markers are the median values and the error bars indicate the 30th and the 70th percentiles over the 30-instance set. All TTS are measured, not counting the time required to run the greedy descent that initialises the initial ansatz $\mathcal{S}$, nor the overhead times for operating the quantum annealer.

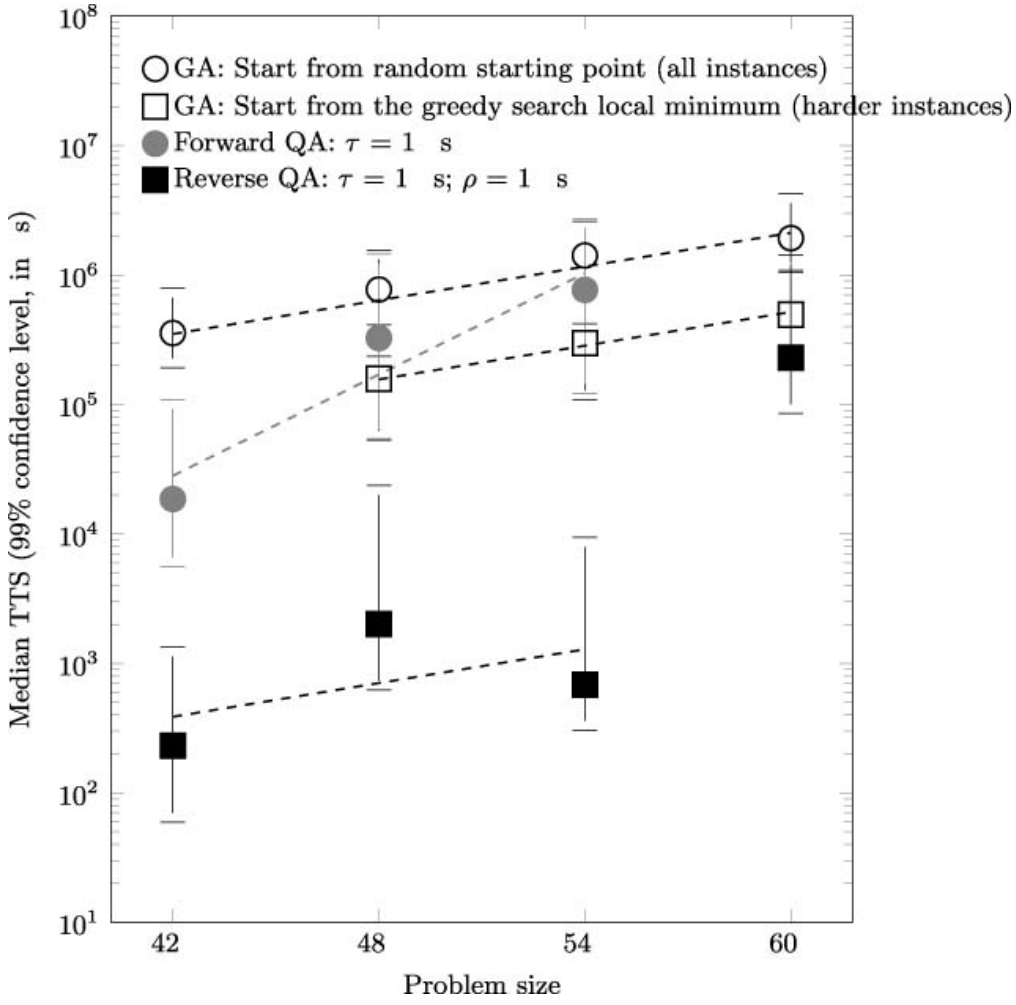


Figure 3.4: Time-to-solution (99% confidence level): GA, forward and reverse quantum annealing. Unconstrained portfolio optimisation problem encoded using the mapping schemes A and D.

The GA can also be initialised by the greedy search heuristics, and this also decreases the TTS required for the GA to find the global minimum. As was experimentally established in [296], the best results are obtained for the smallest possible annealing time and pause time (1 μs). In the median case, we observe one to three orders of magnitude in speedup when applying reverse quantum annealing with respect to forward quantum annealing or classical benchmark.

It is likely that the non-monotonic behaviour of the reverse quantum annealing TTS for $N = 54$ is not of fundamental significance but is due to the noise associated with the finite, small size of our instance set. Although the small size of the instance set makes it difficult to draw a

definite conclusion about the scaling of QA with the problem size, it appears that reverse quantum annealing displays similar scaling to GAs for portfolios of up to $N = 54$ assets – as illustrated by the dashed regression lines. A worse scaling for the limit case of $N = 60$ assets is probably due to the fact that, in this case, the physical qubit chains become too long and more likely to be broken. It is worth noting that for the same reason, $N = 60$ is also a very hard case for forward annealing. Reverse annealing displays significantly better scaling in comparison with forward annealing. While parameter β in (2.3.2) for forward annealing is equal to 0.3, it becomes 0.1 for reverse annealing.



Discrete portfolio optimisation is an NP-hard problem that can be solved on a quantum annealer using a hybrid quantum-classical reverse annealing technique with several orders of magnitude in quantum speedup (ignoring the measurement and system reset computational overheads). Although it is too early to say whether quantum annealing can become a widely adopted portfolio optimisation tool, there are indications that as technology and theory progress, it could represent a viable choice.

Summary

In this chapter, we applied quantum annealing to solving a discrete portfolio optimisation problem. We started with the principles of quadratic unconstrained binary optimisation and looked at several examples of NP-hard optimisation problems and their QUBO formulations.

Then, we introduced the concept of a quantum annealing protocol and specified two such protocols: forward annealing and reverse annealing.

We also specified the classical benchmark: the genetic algorithm, an evolutionary search heuristic ideally suited for operations on binary variables.

Once we had all the necessary building blocks, we translated a sample discrete portfolio optimisation problem into QUBO and experimented with solving its instances on a D-Wave 2000Q quantum annealer. We collected sufficient statistics on various problem sizes to compare the performance of forward and reverse quantum annealing with the classical benchmark. We obtained encouraging results in terms of TTS, especially for the reverse quantum annealing protocol.

In the next chapter, we will learn how to apply quantum annealing to the problem of construction of a robust classifier. The proposed solution – quantum boosting – is a hybrid quantum-classical classifier (trained quantumly and run classically) that performs on a par with the standard classical models and, sometimes, outperforms them.

Join our book's Discord space

Join our Discord community to meet like-minded people and learn alongside more than 2000 members at: https://packt.link/quantum



4
Quantum Boosting

In this chapter, we consider a quantum version of the classical boosting meta-algorithm – a family of machine learning algorithms that convert weak classifiers into strong ones. Classically, boosting consists of two main operations: i) adaptive (iterative) training of the weak classifiers, thus improving their individual performance, and ii) finding an optimal configuration of weights applied to the individual weak learners when combining them into a single strong one.

Adaptive learning consists of iterative re-weighting of the samples from the training dataset, forcing the model to improve its performance on the difficult-to-classify samples by giving them heavier weights. These weights are adjusted at each algorithm iteration. Arguably, the best-known and most successful example of such algorithms is the popular *adaptive boosting* (AdaBoost) model. It was first formulated in 1997 by Freund and Schapire [107], whose work has been recognised by the awarding of the prestigious Gödel Prize in 2003.

The main principle of AdaBoost is that the base classifiers (weak learners) are trained in sequence and each base classifier is trained using a weighted form of the dataset: the weighting coefficient associated with each sample depends on the performance of the previous classifiers. Samples that are misclassified by one of the base classifiers are given larger weights when used to train the next base classifier in the sequence. Once all base classifiers have been trained, their predictions are combined through some kind of weighted majority voting scheme [37]. Therefore, AdaBoost can be seen as a general framework that allows many possible realisations with various degrees of sophistication rather than a narrowly defined algorithm.

In contrast to AdaBoost, a boosting approach that consists of finding an optimal set of weights for the individual weak learners (with the weak learners being trained in the usual way) is straightforward to implement and relies on standard optimisation routines. However, this task becomes a hard combinatorial problem when an additional set of constraints is introduced. When the weights are only allowed to take binary values, the problem naturally lends itself to being formulated as a QUBO problem.

This is where quantum annealing has a role to play, as we have seen in Chapter 3. For a large enough number of weak classifiers, the search space becomes enormous, and classical algorithms (such as various evolutionary search heuristics) may take a non-trivial amount of time to find an optimal configuration of weights (or, at least, a good approximation). This is an ideal scenario for quantum annealing to demonstrate its strong points, including the possibility of achieving a material quantum speedup.



Quantum boosting is a QUBO-based technique combining individual weak learners into a single strong classifier by constructing an optimal linear combination of binary classifiers. It is transparent, easy to interpret, and resistant to overfitting.

4.1 Quantum Annealing for Machine Learning

Quantum boosting is the first QML algorithm we will consider in this book. This is also the algorithm that plays to the natural strengths of quantum annealing.

4.1.1 General principles of the QBoost algorithm

We start with the general principles of the Quantum Boosting (QBoost) algorithm before exploring a specific finance-related application. In the formulation of QBoost, we will be using the following definitions and notations:

Object	Definition
$x_\tau = (x_1(\tau), x_2(\tau), \ldots, x_N(\tau))$	Vector of N variables (features)
$y_\tau = \pm 1$	Binary label indicating whether x_τ corresponds to Class 0 (−1) or Class 1 (+1)
$\{x_\tau, y_\tau\}_{\tau=1,\ldots,M}$	Set of training events
$c_i(x_\tau) = \pm\frac{1}{N}$	Value of the weak classifier i on the event τ
$q := (q_1, q_2, \ldots, q_N)$	Vector of binary (0 or 1) weights associated with each weak classifier

Table 4.1: QBoost algorithm notations.

We first specify the classification error for sample τ, which is given by the squared error

$$\left(\sum_{i=1}^{N} c_i(x_\tau) q_i - y_\tau \right)^2 .$$

The total cost function to minimise is then the sum of squared errors across all samples:

$$
\begin{aligned}
L(\mathrm{q}) &= \sum_{\tau=1}^{M} \left(\sum_{i=1}^{N} c_i(\mathrm{x}_\tau) q_i - y_\tau \right)^2 \\
&= \sum_{\tau=1}^{M} \left(\sum_{i=1}^{N} c_i(\mathrm{x}_\tau) q_i \sum_{j=1}^{N} c_j(\mathrm{x}_\tau) q_j - 2 y_\tau \sum_{i=1}^{N} c_i(\mathrm{x}_\tau) q_i + y_\tau^2 \right).
\end{aligned}
$$

Note that y_τ^2 does not depend on q and therefore has no influence on the minimisation of L. Adding a penalty $\lambda > 0$ to prevent overfitting, the objective function to minimise is thus

$$
\begin{aligned}
\widetilde{L}(\mathrm{q}) &= \sum_{\tau=1}^{M} \left(\sum_{i=1}^{N} c_i(\mathrm{x}_\tau) q_i \sum_{j=1}^{N} c_j(\mathrm{x}_\tau) q_j - 2 y_\tau \sum_{i=1}^{N} c_i(\mathrm{x}_\tau) q_i \right) + \lambda \sum_{i=1}^{N} q_i \\
&= \sum_{\tau=1}^{M} \left(\sum_{i=1}^{N} \sum_{j=1}^{N} q_i q_j c_i(\mathrm{x}_\tau) c_j(\mathrm{x}_\tau) - 2 \sum_{i=1}^{N} q_i c_i(\mathrm{x}_\tau) y_\tau \right) + \lambda \sum_{i=1}^{N} q_i \\
&= \sum_{i=1}^{N} \sum_{j=1}^{N} \left(\sum_{\tau=1}^{M} c_i(\mathrm{x}_\tau) c_j(\mathrm{x}_\tau) \right) q_i q_j - 2 \sum_{i=1}^{N} \left(\sum_{\tau=1}^{M} c_i(\mathrm{x}_\tau) y_\tau \right) q_i + \lambda \sum_{i=1}^{N} q_i \\
&= \sum_{i=1}^{N} \sum_{j=1}^{N} C_{ij} q_i q_j + \sum_{i=1}^{N} (\lambda - 2C_i) q_i,
\end{aligned}
$$

with

$$
C_{ij} := \sum_{\tau=1}^{M} c_i(\mathrm{x}_\tau) c_j(\mathrm{x}_\tau) \quad \text{and} \quad C_i := \sum_{\tau=1}^{M} c_i(\mathrm{x}_\tau) y_\tau.
$$

Remark: Adding a penalty term controlled by the coefficient λ is analogous to the LASSO regression method [6] with L_1 penalty, which is ubiquitous in machine learning. Ridge regression [243] with L_2 penalty could also be used and would also lead to a QUBO problem.

4.1.2 QUBO to Ising

As developed in Chapter 3.1.1, we now perform a transformation from QUBO to Ising from the binary decision variables $q := (q_1,\ldots,q_N) \in \{0,1\}^N$ to spin variables $s := (s_1,\ldots,s_N) \in \{-1,+1\}^N$ using

$$s = 2q - 1 \quad \text{or} \quad q = \frac{1}{2}(s + 1).$$

Therefore, the Ising problem to be solved on the quantum annealer reads

$$\begin{aligned}\mathcal{H} = \widetilde{L}(s) &= \sum_{i,j=1}^{N} \left(\frac{1}{2}s_i + \frac{1}{2}\right)\left(\frac{1}{2}s_j + \frac{1}{2}\right) C_{ij} + \sum_{i=1}^{N} \left(\frac{1}{2}s_i + \frac{1}{2}\right)(\lambda - 2C_i) \\ &= \frac{1}{4}\sum_{i,j=1}^{N} s_i s_j C_{ij} + \frac{1}{2}\sum_{i,j=1}^{N} s_i C_{ij} + \frac{1}{4}\sum_{i,j=1}^{N} C_{ij} \\ &+ \frac{1}{2}\sum_{i=1}^{N} s_i \lambda + \frac{\lambda N}{2} - \sum_{i=1}^{N} s_i C_i - \sum_{i=1}^{N} C_i.\end{aligned}$$

Since the three terms

$$\frac{1}{4}\sum_{i,j=1}^{N} C_{ij}, \quad \frac{\lambda N}{2}, \quad \text{and} \quad \sum_{i=1}^{N} C_i$$

do not depend on s, they can be removed from the cost function. The substitution $\lambda = \frac{1}{2}\lambda$ then yields the final Ising problem

$$\mathcal{H} = \frac{1}{4}\sum_{i,j=1}^{N} s_i s_j C_{ij} + \frac{1}{2}\sum_{i,j=1}^{N} s_i C_{ij} + \sum_{i=1}^{N} s_i(\overline{\lambda} - C_i).$$

The problem that quantum annealing attempts to solve is to minimise $\mathcal{H}$ and to return the minimising, ground-state spin configuration $(s_i^g)_{i=1,\ldots,N}$. The strong classifier is then built as

$$R(\mathrm{x}) = \sum_{i=1}^{N} s_i^g c_i(\mathrm{x}) \in [-1, 1],$$

for each new event x that we wish to classify [218].

4.2 QBoost Applications in Finance

Quantum Annealing for Machine Learning (QAML) has been applied productively to a wide range of financial and non-financial use cases. It demonstrated a performance advantage in comparison with standard classical machine learning models such as the binary decision tree-based Extreme Gradient Boosting (XGBoost) and Deep Neural Network (DNN) classifiers, especially on relatively small datasets. The QAML use cases come from such diverse fields as high-energy physics (the Higgs boson detection [218]) and computational biology (the classification and ranking of transcription factor binding [186]). In finance, the most obvious application of QAML is to credit scoring and fraud detection as well as to the construction of strong trading signals from large numbers of weak binary (buy/sell) trading signals.

In this section, we analyse QBoost performance on the more conventional binary classification problem – forecasting credit card client defaults. We also provide classical benchmarks (gradient boosting and feedforward neural network classifiers) and analyse QBoost performance from different angles. The chosen dataset is relatively large, with tens of thousands of samples, which should help standard classical classifiers avoid overfitting and demonstrate their best qualities.

It has been established in [218] that the QBoost algorithm is resistant to overfitting because it involves an explicit linearisation of correlations (hence its better performance on the smaller dataset in comparison with classical benchmarks). Another useful aspect of the model is that it is

interpretable directly, with each weak classifier corresponding to a specific feature or a combination of features (or their functions), and the strong classifier being a simple linear combination thereof. This compares favourably with the "black box" machine learning discriminants, such as when using gradient boosting or DNNs. This is especially important for financial products aimed at retail customers.

4.2.1 Credit card defaults

The default of credit card clients (DCCC) dataset is available from the UCI Machine Learning Repository [307, 308]. The dataset consists of 30,000 samples with binary classification: a client defaults on the credit card payment (Class 1) and a client does not default (Class 0). There are 23 features (F1-F23) that have at least some predictive power and can be used for the classification decision:

- F1: Amount of the given credit (NT dollar): it includes both the individual consumer credit and his/her family (supplementary) credit.
- F2: Gender (1 = male; 2 = female).
- F3: Education (1 = graduate school; 2 = university; 3 = high school; 4 = others).
- F4: Marital status (1 = married; 2 = single; 3 = others).
- F5: Age (years).
- F6-F11: History of past payments. F6 – the repayment status in the previous month, F7 – the repayment status two months ago, and so on. The measurement scale for the repayment status is: −1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; …; 8 = payment delay for eight months; 9 = payment delay for nine months and above.
- F12-F17: Amount of bill statement (NT dollar). F12 – amount of bill statement previous month, F13 – amount of bill statement two months ago, and so on.

- F18-F23: Amount of previous payment (NT dollar). F18 – amount paid last month, F19 – amount paid two months ago, and so on.

The weak classifiers were constructed in the following way: each feature was used separately as an input into the logistic regression classifier with the aim of making a binary prediction: $-1/N$ for Class 0 (no default) and $+1/N$ for Class 1 (default), where $N = 23$ is the total number of weak classifiers (number of features in the dataset). Note that this is not the only possible approach. It is perfectly feasible to build the weak classifiers through some (possibly non-linear) combination of original features. This should be done every time we have a clear understanding of which combination of features would produce a more meaningful and insightful result. However, in this particular example, our objective is to illustrate the general principles of QBoost algorithm and we do not assume any subject matter expertise that would allow us to construct better derived features.

We have used `sklearn.linear_model.LogisticRegression` from the `scikit-learn` package [230] for the weak classifiers. The dataset was split into training and testing datasets at a 70:30 ratio with the help of the sklearn.model_selection.train_test_split module. The class labels were encoded as −1 for Class 0 and +1 for Class 1, as per the QBoost algorithm requirements.

The following configuration of the `LogisticRegression` model was used in constructing the weak classifiers dataset (all other parameters were set at their default values):

- penalty = 'l2'
- C = 1.0
- solver = 'lbfgs'
- max_iter = 1000

Therefore, we have a training dataset (21,000 samples) and a testing dataset (9,000 samples), each consisting of the predictions of 23 weak classifiers (taking values {−1⁄23, +1⁄23}) and class labels (taking values in {−1, +1}). If the prediction of the strong classifier is given by the sum of predictions of the weak classifiers (a simple majority voting approach), its value will be in the [−1,1] range, with the values of −1 and +1 achieved if all the weak classifiers are in perfect agreement with each other.

QBoost provides an improvement on this approach by finding an optimal configuration of the weak classifiers such that the majority voting is performed on a subset of available weak classifiers. In other words, a majority voting performed on all weak classifiers is just a special case of the QBoost (one of the possible configurations to explore). Therefore, it is necessary to compare QBoost performance with more advanced classical machine learning models such as gradient boosting and neural networks. We provide this comparison in Section 4.3.

4.2.2 QUBO classification results

Each feature in the DCCC dataset is uniquely mapped to the corresponding (weak) logistic regression classifier and associated binary decision variable $(q_i)_{i=1,\ldots,23}$. These decision variables are represented by the logical qubits/spin variables in the QUBO/Ising formulation of the optimisation problem.

The number of non-zero decision variables (weights) depends on the degree of regularisation we would like to impose. Table 4.2 shows the optimal configurations of the weights as a function of the penalty λ obtained for the training dataset. Given the relatively small number of weak classifiers in our example, the optimal configuration can be found by an exhaustive search. As one would expect, the larger the value of the penalty λ, the smaller the number of non-zero weights.

λ	non-zero weights
500	$\{q_1, q_6, q_7, q_8, q_9, q_{10}, q_{11}\}$
600	$\{q_6, q_7, q_8, q_9, q_{10}, q_{11}\}$
700	$\{q_6, q_7, q_{10}, q_{11}\}$
800	$\{q_6, q_{10}, q_{11}\}$
900	$\{q_6, q_{11}\}$
1000	$\{q_6\}$

Table 4.2: Optimal configurations of QUBO weights q for various values of the penalty λ. The optimal configurations list all non-zero weights.

Given a configuration of weights, we can build the strong classifier as per (4.1.2). Then, we can compare the performance of the obtained strong classifier on both training (in-sample) and testing (out-of-sample) datasets. The performance metrics of choice are *accuracy*, *precision*, and *recall*. The classifier performance can also be visualised with the help of a *confusion matrix*. Here are their definitions.

- **Accuracy** is the ratio of correctly predicted observations to the total observations. Accuracy is a good metric for classes of roughly the same size and equivalent importance. However, it is a poor metric for the dataset in our example: the Class 0 samples (no default) are far more numerous but the relative importance of Class 1 samples (default) is much higher.
- **Precision** is the ratio of correctly predicted positive observations to the total predicted positive observations. High precision corresponds with a low false positive rate. This is a metric we would like to maximise in the context of credit card defaults if there is a high cost associated with the incorrect default predictions.
- **Recall** is the ratio of correctly predicted positive observations to all observations in the positive class. In the context of credit card

defaults, this metric shows how many of the actual defaults were predicted by the classifier. We would like to maximise this metric from the risk management perspective.

- **Confusion matrix** for a binary classifier is a 2 × 2 matrix whose elements are the counts of the true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions of a classifier, as shown in Figure 4.1.

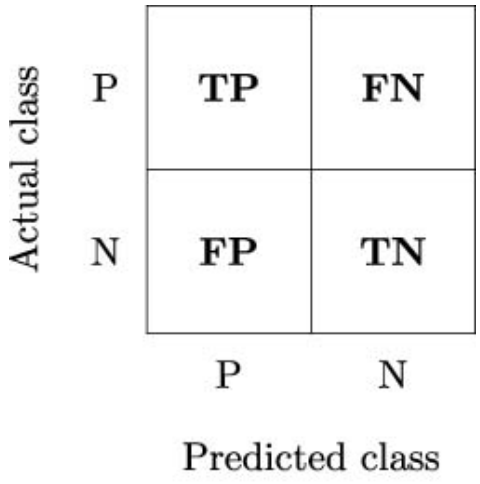


Figure 4.1: Confusion matrix for a binary classifier.

Accuracy, precision, and recall are then defined as follows:

$$\begin{aligned} \text{Accuracy} &:= \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \\ \text{Precision} &:= \frac{\text{TP}}{\text{TP} + \text{FP}}, \\ \text{Recall} &:= \frac{\text{TP}}{\text{FN} + \text{TP}}. \end{aligned}$$

Figure 4.2 displays in-sample and out-of-sample confusion matrices for the strong QBoost classifier assuming that Class 1 (default) is the positive class and Class 0 (no default) is the negative class. The penalty was set at $\lambda = 10^3$, thus enforcing strong regularisation.

The in-sample and out-of-sample results are quite close, as one would expect from a strongly regularised classifier. Table 4.3 summarises the results.

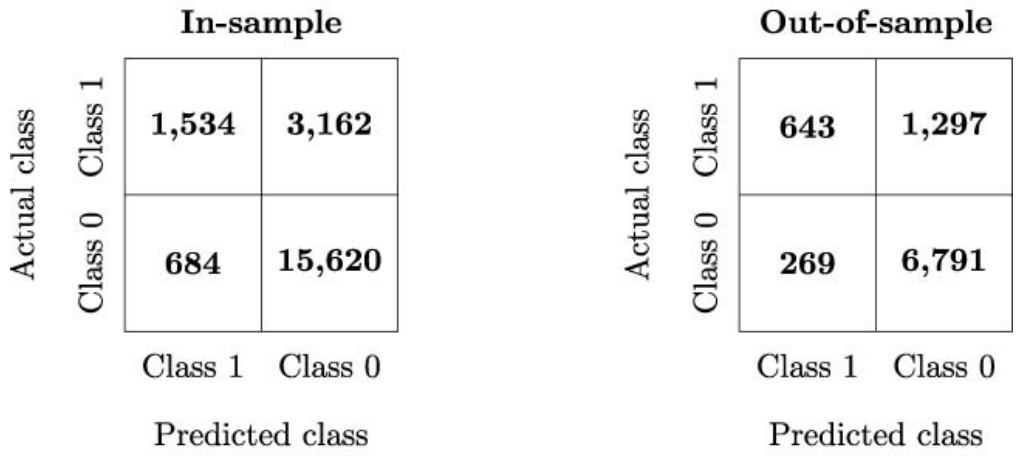


Figure 4.2: Confusion matrices for the QBoost classifier (DCCC dataset).

	Accuracy	Precision	Recall
In-sample	0.82	0.69	0.33
Out-of-sample	0.83	0.71	0.33

Table 4.3: Accuracy, precision, and recall for the QBoost classifier trained and tested on the DCCC dataset.

4.3 Classical Benchmarks

Classical benchmarking is an important element of the testing of quantum algorithms. Small-scale (or even stylised) problems are ideally suited for this task. Let us see how the QBoost model performs in comparison with the standard classical ML classifiers: neural networks and gradient boosting.

4.3.1 Artificial neural network

An Artificial Neural Network (ANN) is a network of interconnected *activation units* (or *artificial neurons*), where each activation unit

performs three main functions (Figure 4.3):

- Summation of the input signals $(x_i)_{i=1,\ldots,N}$, from all the upstream units to which it is connected with multiplication by the corresponding weights $(w_i)_{i=1,\ldots,N}$;
- Non-linear transformation of the aggregated input;
- Sending the result to the downstream units to which it is connected.

Sometimes the activation unit also performs binarisation (or, more generally, digitisation) of the output – typically, this is a task of the activation units in the output layer of an ANN trained as a classifier.

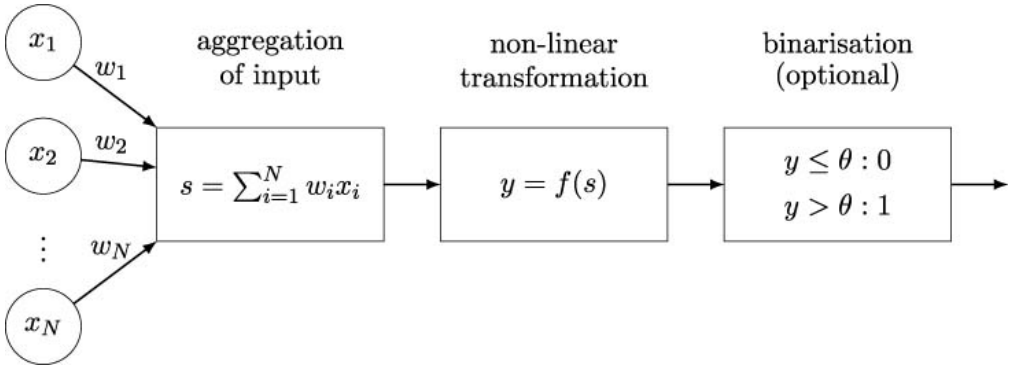


Figure 4.3: Schematic representation of an artificial neuron (perceptron).

In its simplest form, an ANN is organised as layers of activation units: an input layer, an output layer, and one or several hidden layers, as schematically pictured in Figure 4.4.

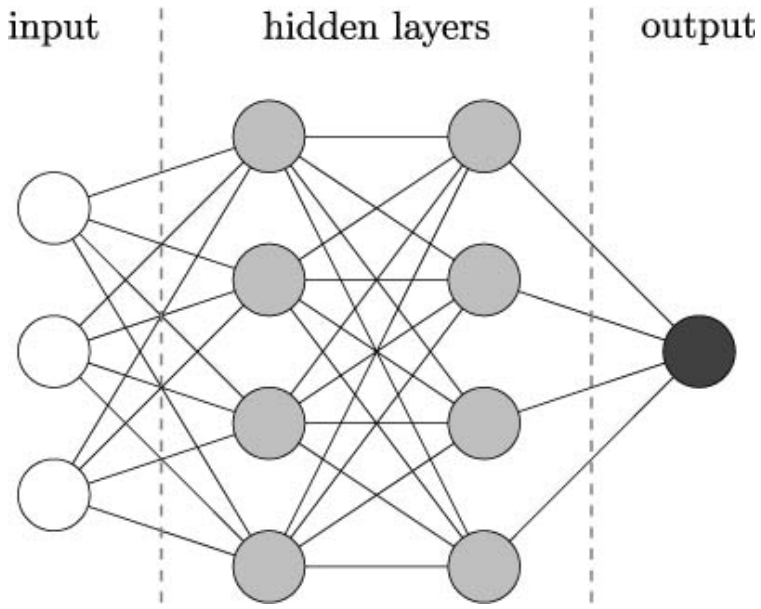


Figure 4.4: Schematic representation of a feedforward ANN.

The activation unit in Figure 4.3 is known as a *perceptron*, and the ANNs consisting of layers of perceptrons are known as Multi-Layer Perceptrons (MLPs). MLPs are *feedforward* neural networks: the signal travels in one direction from the input layer to the output layer. ANNs can be organised differently with signal travelling back and forth between the layers, and we will explore one such model in the next chapter. However, when it comes to building a classifier, the simple feedforward architecture works well in practice.

The practical approach to the ANN architecture is based on the fundamental result obtained by Cybenko [75]. It states that arbitrary decision regions can be arbitrarily well approximated by continuous feedforward neural networks with only a single hidden layer and any continuous sigmoidal non-linearity. This result was further generalised to the wider range of activation functions by Hornik, Stinchcombe, and White [141]. It was established that multilayer feedforward networks with only a single hidden layer and an appropriately smooth hidden layer activation function are capable of arbitrarily accurate approximating any arbitrary function and its derivatives. In fact, these

networks can even approximate functions that are not differentiable in the classical sense, but possess only generalised derivatives [224].

4.3.2 Training artificial neural networks

The process of training an ANN consists in finding an optimal configuration of network parameters (weights and biases) such that the new unseen input is transformed in the desired way. The network is trained on what is known as a *training dataset*. The samples from the training dataset can be *labelled* (each sample is assigned a class label, either numerical or categorical). In this case, we can perform *supervised learning*, where the network is tasked with learning the mapping between the features and the class labels – an ANN trained in the supervised learning mode becomes a classifier. When the samples are not labelled, we can train the network as a regressor. Although ANNs trained as classifiers may seem to be the most obvious practical decision-making tools, regressors too find numerous applications in various fields of quantitative finance, for example in learning the natural dynamics and transformations of interest rate curves [169].

However, we would like to focus here on the labelled datasets since our objective is to consider a classical counterpart of the QBoost classifier. The standard approach to training a feedforward ANN is the *backpropagation* of error with gradient descent [113]. We briefly explain the main idea of this method.

The starting point is the specification of some suitable cost function that indicates how far we are from the correct classification. Without loss of generality, assume that we work with a training dataset consisting of M samples, where each sample is a pair of an N-dimensional vector of features and a binary class label:

$$\{\mathrm{x}^j, y^j\}_{j=1,\ldots,M}, \quad \text{with} \quad \mathrm{x}^j := (x_1^j, \ldots, x_N^j) \quad \text{and} \quad (y^j)_{j=1,\ldots,M} \in \{0,1\}.$$

Let $(\hat{y}^j)_{j=1,\ldots,M}$ be the class labels assigned to the corresponding training samples by the ANN for some configuration of the network weights $w = (w_1,\ldots,w_K)$. Then, we can define the cost function as

$$L(\mathrm{w}) := \sum_{j=1}^{M} g\left(y^j, \hat{y}^j(\mathrm{w})\right),$$

where $g(y^j,\hat{y}^j(\mathrm{w}))$ is the estimation error for sample j. There are many possible ways of specifying the error function, the most popular being the squared error

$$g(y^j, \hat{y}^j) := \left(y^j - \hat{y}^j\right)^2.$$

Given the cost function $L(\cdot)$, we can calculate its sensitivities (derivatives) $\partial L(\mathrm{w})/\partial w_k$, for each $k = 1,\ldots,K$, with respect to the network weights. We can then *update* the weights by changing them in the direction that would reduce the estimation error, i.e., by moving in the opposite direction of the corresponding gradients:

$$w_k \longleftarrow w_k - \eta \frac{\partial L(\mathrm{w})}{\partial w_k},$$

where the coefficient η is called the *learning rate*, which can be either constant or dynamic. We then iterate the procedure given by (4.3.2), (4.3.2), and (4.3.2) until either the estimation error drops below a predefined threshold or a maximum number of iterations is reached. Often the learning rate is set initially at some relatively large value and then decays exponentially with the number of iterations.

The gradients can be calculated numerically (e.g., using the finite difference method) or analytically, the latter being obviously preferable. The most widely used non-linear activation functions and

their gradients are listed in Table 4.4 and their plots are shown in Figure 4.5:

$$\left(1+\mathrm{e}^{-x}\right)$$

Table 4.4: Activation functions.

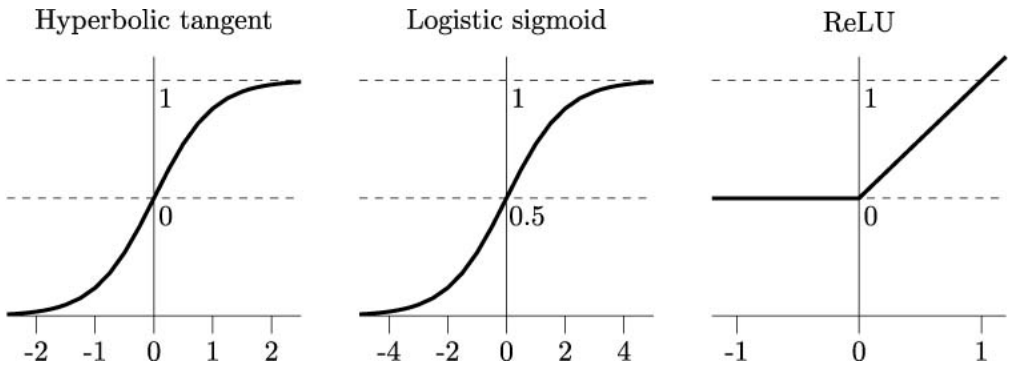


Figure 4.5: Activation functions.

Remark: The sigmoid activation functions, such as logistic sigmoid and hyperbolic tangent, are the activation functions of choice for shallow neural networks with only a couple of hidden layers. In this case, it is possible to exploit the smoothness of the sigmoid functions in order to achieve the best possible approximation of the function we are trying to learn. However, in the case of deep neural networks with a large number of hidden layers, we face the problem of vanishing gradients – gradients of $\sigma(x)$ and $\tanh(x)$ become null as $x \to \pm\infty$. At the same time, ReLU always has a non-zero gradient for all $x > 0$, which makes it the activation function of choice for deep neural networks whenever it makes sense to sacrifice the smoothness of the activation function for non-zero gradients.

Finally, the problem of overfitting can be addressed by adding a regularisation penalty term to (4.3.2), for example the following L_2 penalty, which discourages large network weights associated with strong non-linearity:

$$L(\mathrm{w}) := \sum_{j=1}^{M} g\left(y^j, \hat{y}^j(\mathrm{w})\right) + \lambda ||\mathrm{w}||^2,$$

where the parameter λ controls the degree of regularisation.

4.3.3 Decision trees and gradient boosting

The decision tree approach to classification is based on the concept of splitting a dataset on the available features in order to maximise the *information gain*, defined as

$$G(D, f) = I(D) - \sum_{j=1}^{M} \frac{N_j}{N} I(d_j),$$

where D is the dataset of the parent node, $(d_j)_{j=1,\ldots,M}$ are the datasets of the child nodes into which the parent node is split, N is the number of samples in the parent node, $(N_j)_{j=1,\ldots,M}$ are the number of samples in the child nodes, and I is the chosen *impurity measure*. The latter indicates the presence of the samples from the different classes in the same node: it is zero if the node holds samples from a single class and is maximal if the node holds an equal number of samples from the available classes. Therefore, maximisation of the information gain is achieved through minimisation of the child node impurities.

Figure 4.6 provides a schematic representation of a decision tree based on the binary ("rainy/not rainy") and continuous ("wind speed") features. The decision tree algorithm starts at the *root*, which is shown in the figure as a shaded box. Splitting the dataset on the root feature results in the largest information gain. The splitting leads to the creation of *branches* (shown in the figure as arrows going from the parent node to the child nodes) and *leaves* (shown in the figure as white boxes). The terminal leaves (classes) are represented as dashed boxes. The splitting continues until either no more branches can be created or

the maximum allowed depth is reached. It is good practice to avoid the construction of a too deep tree by imposing *pruning* – a strict limit on the maximum depth of the tree, in order to avoid overfitting.

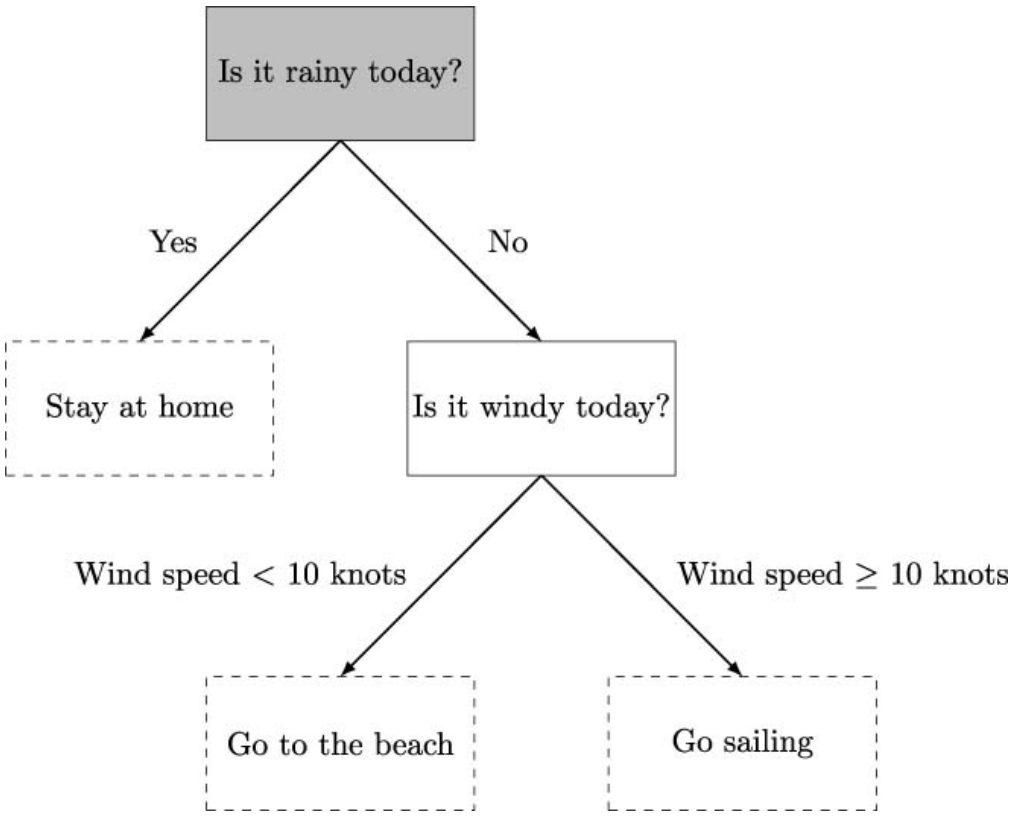


Figure 4.6: Schematic representation of a decision tree.

The most widely used impurity measures are *Gini impurity* and *entropy*. Let $(p_i^l)_{i=1,\dots,C}$ be the proportion of the samples that belong to class i for node l. Then the impurity measures are defined as

$$I_{\text{Gini}} := \sum_{i=1}^{C} p_i^l (1 - p_i^l) \quad \text{and} \quad I_{\text{Entropy}} := -\sum_{i=1}^{C} p_i^l \log_2(p_i^l).$$

Decision trees can be seen as weak learners that can be *boosted* to be strong learners. One of the most popular methods of combining weak classifiers into a single strong classifier is *gradient boosting*. The main principle of gradient boosting is as follows [185].

The objective is to improve the weak classifier through an iterative process with the improvement measured as a minimisation of the estimation error (for example, the squared error given by (4.3.2)). As before, without loss of generality, we assume that we deal with the binary classification problem (4.3.2). Further, assume that at the k-th iteration, the weak learner returns the estimate $\hat{y}_k(\mathrm{x}^j)$ for sample x^j. In order to improve the classification results, the algorithm should add some estimator h_k, such that for the given sample x^j we have

$$\hat{y}_{k+1}(\mathrm{x}^j) := \hat{y}_k(\mathrm{x}^j) + h_k(\mathrm{x}^j) = y^j,$$

where y^j is the correct class label for sample x^j. In other words, the task is to fit the new estimator h_k to the residuals $y^j - \hat{y}_k(\mathrm{x}^j)$, $j = 1,\ldots,M$. We also notice that the estimator h_k is proportional to the negative gradient of the squared error (4.3.2) with respect to $\hat{y}_k$:

$$h_k(\mathrm{x}^j) := y^j - \hat{y}_k(\mathrm{x}^j) = -\frac{1}{2}\frac{\partial g(y^j, \hat{y}_k(\mathrm{x}^j))}{\partial \hat{y}_k}.$$

Therefore, gradient boosting combines boosting with the gradient descent algorithm.

4.3.4 Benchmarking against standard classical classifiers

The classical benchmarks of choice are the MLP classifier (`sklearn.neural_network.MLPClassifier`) and the gradient boosting classifier (`sklearn.ensemble.GradientBoostingClassifier`). Table 4.5 holds weakly optimised model parameters: we did not search for the absolute best set of model parameters but tried just a few configurations. We can think of it as a very rough grid search method that produces a viable configuration of model parameters but is not necessarily optimal. All other model parameters were set at their default values.

Gradient Boosting Classifier	MLP Classifier
loss = 'deviance'	hidden_layer_sizes = (20)
learning_rate = 0.1	activation = 'tanh'
n_estimators = 1000	solver = 'adam'
criterion = 'friedman_mse'	alpha = 0.1
max_depth = 3	max_iter = 5000
	alpha = 0.01

Table 4.5: Model parameters for classical benchmarks.

Figure 4.7 displays out-of-sample confusion matrices for the classical benchmarks and Table 4.6 provides a direct comparison of the out-of-sample results for the QBoost and classical classifiers.

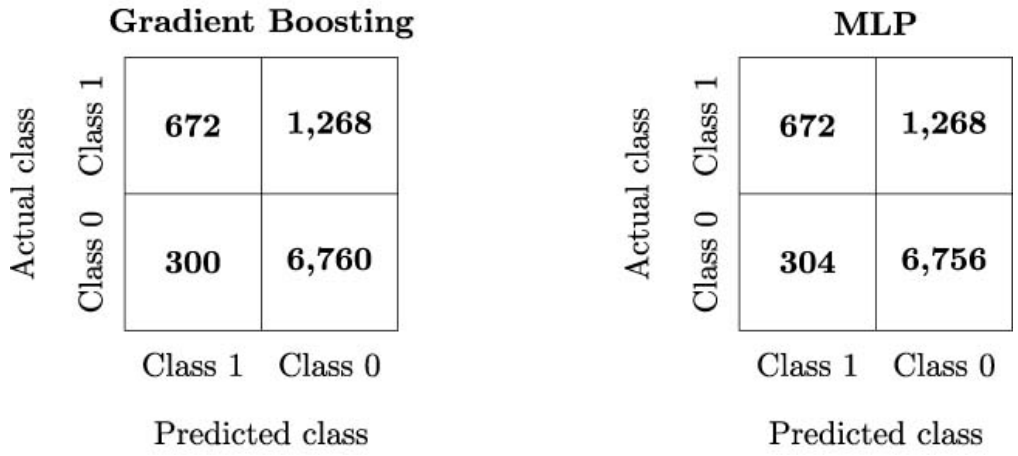


Figure 4.7: Confusion matrices for the gradient boosting and MLP classifiers (DCCC dataset, out-of-sample results).

	Accuracy	Precision	Recall
Gradient Boosting	0.83	0.69	0.35
MLP	0.83	0.69	0.35
QBoost	0.83	0.71	0.33

Table 4.6: Out-of-sample accuracy, precision, and recall for the QBoost, gradient boosting, and MLP classifiers (DCCC dataset).

QBoost achieves similar out-of-sample results to those of gradient boosting and MLP classifiers. A comparison of in-sample and out-of-sample QBoost performance confirms QBoost's ability to impose strong regularisation and avoid overfitting. At the same time, QBoost provides full transparency in terms of which features contribute to the strong classifier. We also obtain an explicit optimal configuration of features for any given degree of regularisation. This is not the case when we deal with the conventional machine learning models, which may require an extensive analysis of sensitivities and feature importances to unpack their "black boxes".



Quantum boosting can be applied to financial optimisation problems where the emphasis is on transparency, interpretability, and robustness.

Summary

In this chapter, we learned how to apply quantum annealing to build a strong classifier from several weak ones. We started with the general principles of quantum boosting and its corresponding QUBO formulation.

We then illustrated the application of the QBoost algorithm to solving a practical real-world financial problem, namely predicting credit card clients defaulting on their payments. The chosen dataset is reasonably large and complex enough to provide a meaningful challenge while remaining easy to understand and interpret the obtained results.

It is important to have an objective comparison with the corresponding classical counterparts. With this in mind, we introduced several

classical classifiers based on the concepts of a feedforward neural network and a decision tree. We benchmarked QBoost against the MLP and gradient boosting models using such metrics as accuracy, precision, and recall.

In the next chapter, we will learn how quantum annealing can assist in training powerful generative machine learning models.

Join our book's Discord space

Join our Discord community to meet like-minded people and learn alongside more than 2000 members at: https://packt.link/quantum



5
Quantum Boltzmann Machine

As we saw in Chapters 3 and 4, quantum annealing can be used to solve hard optimisation problems. However, the range of possible applications of quantum annealing is much wider than that. In this chapter, we will consider two distinct but related use cases that go beyond solving optimisation problems: sampling and training deep neural networks. Specifically, we will focus on the Quantum Boltzmann Machine (QBM) – a generative model that is a direct quantum annealing counterpart of the classical Restricted Boltzmann Machine (RBM), and the Deep Boltzmann Machine (DBM) – a class of deep neural networks composed of multiple layers of latent variables with connections between the layers but not between units within each layer.

We start by providing detailed descriptions of the classical RBM, including the corresponding training algorithm. Due to the fact that an RBM operates on stochastic binary activation units, one can establish the correspondence between the RBM graph and the QUBO graph embedded onto the quantum chip. This provides the main motivation for performing Boltzmann sampling (the key stage in training RBMs and DBMs) using quantum annealing.

DBMs can be trained as both generative and discriminative models. In both cases, since a DBM can be constructed by stacking together layers of RBMs, efficient Boltzmann sampling is the key element of the training process. Quantum annealing, which can be integrated into the hybrid quantum-classical training routine, has the potential to improve speed and accuracy. Quantum speedup is an especially appealing element of the envisaged quantum advantage since it can be achieved not only during the RBM training stage but also during the process of generating new samples.

5.1 From Graph Theory to Boltzmann Machines

We provide here a short self-contained review of graph theory in order to introduce Boltzmann machines (or energy-based models), which one can view as particular types of connected graphs or networks.

A *graph* is a set of vertices (points or nodes) and edges that connect the vertices. A *directed graph* is a type of graph that contains ordered pairs of vertices while an *undirected graph* is a type of graph that contains unordered pairs of vertices.

We consider a graph $\mathcal{G} = (\mathcal{V},\mathcal{E})$ characterised by a finite number of vertices $\mathcal{V}$ and undirected edges $\mathcal{E}$. For a given vertex $v \in \mathcal{V}$, its neighbourhood is defined as the set of all vertices connected to it by some edge, or

$$\mathcal{N}(v) := \{w \in \mathcal{V} : \{v, w\} \in \mathcal{E}\}.$$

Finally, a *clique* $\mathcal{C}$ is a subset of $\mathcal{V}$ such that all vertices in $\mathcal{C}$ are pairwise connected by some edge in $\mathcal{E}$.

To each vertex $v \in \mathcal{V}$, we associate a random variable X_v taking values in some space $\mathcal{X}$. The vector $X \in \mathcal{X}^{|\mathcal{V}|}$ is called a Markov random field if

$$\text{Law}\left(X_v | (X_w)_{\{w \in \mathcal{V}\setminus\{v\}}\right) = \text{Law}\left(X_v | (X_w)_{\{w \in \mathcal{N}(v)\}}\right).$$

The following theorem, originally proved by Hammersley and Clifford [125] (see also [167, Theorem 4.2]), provides a way to express the law of Markov random fields over graphs in a convenient form. The Markovian property is fundamental here, as dynamics (for example, the passing of a signal from a hidden layer to a visible layer of an RBM network) should only depend on the current state and not on the whole path followed by the system.

Theorem 8 (Hammersley-Clifford Theorem)**.** *A strictly positive distribution satisfies the Markov property with respect to an undirected graph if and only if it factorises over it.*

Phrased differently, the theorem says that X is Markovian over $\mathcal{G}$ if its distribution can be written as

$$\mathbb{P}_X(\mathrm{x}) := \mathbb{P}(X = \mathrm{x}) = \frac{1}{Z}\prod_{C\in\mathcal{C}} \psi_C(\mathrm{x}_C), \quad \text{for all } \mathrm{x} \in \mathcal{X}^{|\mathcal{V}|},$$

for a set $\{\psi_C\}_{C\in\mathcal{C}}$ of functions called the potential over all the cliques $C \in \mathcal{C}$ and where Z is a normalisation constant such that the probabilities integrate to unity. Here x_C naturally corresponds to the elements of the vector x over the clique C. The factorisation is often taken over the so-called *maximal cliques*, namely the cliques that are no longer cliques if any node is added. If the distribution of X is strictly positive, then so are the functions $\{\psi_C\}_{C\in\mathcal{C}}$ and therefore (5.1) can be written as

$$\mathbb{P}_X(\mathrm{x}) = \frac{1}{Z}\exp\left(\sum_{C\in\mathcal{C}} \log(\psi_C(\mathrm{x}_C))\right) =: \frac{1}{Z}\mathrm{e}^{-E(\mathrm{x})},$$

for all $\mathrm{x} \in \mathcal{X}^{|\mathcal{V}|}$. The function

$$E(\mathrm{x}) := -\sum_{C\in\mathcal{C}} \log(\psi_C(\mathrm{x}_C))$$

is called the *energy* function. Because of their uses in statistical physics, strictly positive distributions of Markov random fields, taking the form (5.1), are also called Boltzmann or Gibbs distributions.



Energy-based models are generative models that discover data dependencies by applying a measure of compatibility (scalar energy) to each configuration of the observed and latent variables. The inference consists of finding the values of latent variables that minimise the energy given the values of the observed variables. Energy-based models posses many useful properties (simplicity, stability, flexibility, compositionality) – this makes them models of choice for learning complex multivariate probability distributions.

5.2 Restricted Boltzmann Machine

5.2.1 The RBM as an energy-based model

The RBM corresponds to a special structure of such a graph, called bipartite, where the set $\mathcal{V}$ of vertices can be split into two groups of visible vertices $\mathcal{V}_{\mathrm{V}}$ and hidden vertices $\mathcal{V}_{\mathrm{H}}$ such that the set $\mathcal{E}$ of edges only consists of elements of the form $\{v,h\} \in \mathcal{V}_{\mathrm{V}} \times \mathcal{V}_{\mathrm{H}}$. Figure 5.1 provides a schematic representation of the RBM that implements the bipartite graph structure. This in particular implies that cliques can only be of size one (all the singleton nodes) or two (all the pairs (v,h) in $\mathcal{V}_{\mathrm{V}} \times \mathcal{V}_{\mathrm{H}}$). For simplicity, we shall denote v an element of $\mathcal{X}^{|\mathcal{V}_{\mathrm{V}}|}$ and h an element of $\mathcal{X}^{|\mathcal{V}_{\mathrm{H}}|}$, and identify the random variable X with the vertices. The following lemma gives us the general form of the energy function (5.1) for RBMs.

Lemma 6 (RBM Energy Lemma)**.** *In a Restricted Boltzmann Machine, the energy function takes the form*

$$E(\mathrm{v},\mathrm{h}) = \sum_{i=1}^{N} E_v(v_i) + \sum_{j=1}^{M} E_h(h_j) + \sum_{i=1}^{N}\sum_{j=1}^{M} E_{v,h}(v_i,h_j),$$

for any $\mathrm{v} := (v_1,\ldots,v_{\mathrm{N}}) \in \mathcal{X}^{|\mathcal{V}_{\mathrm{V}}|}$, $\mathrm{h} := (h_1,\ldots,h_{\mathrm{M}}) \in \mathcal{X}^{|\mathcal{V}_{\mathrm{H}}|}$. *Here, N is the number of visible vertices and M is the number of hidden vertices.*

Proof. By the Hammersley-Clifford theorem, for any $\mathrm{v} \in \mathcal{X}^{|\mathcal{V}_{\mathrm{V}}|}$, $\mathrm{h} \in \mathcal{X}^{|\mathcal{V}_{\mathrm{H}}|}$, we have the factorisation

$$\begin{aligned}
\mathbb{P}(\mathrm{v},\mathrm{h}) &= \frac{1}{Z}\prod_{C\in\mathcal{C}}\psi_C((\mathrm{v}_C,\mathrm{h}_C) \in C) \\
&= \frac{1}{Z}\prod_{\{\{\mathrm{v}\}:\mathrm{v}\in\mathcal{V}_{\mathrm{V}}\}}\psi_{\{\mathrm{v}\}}(v)\prod_{\{\{\mathrm{h}\}:\mathrm{h}\in\mathcal{V}_{\mathrm{H}}\}}\psi_{\{\mathrm{h}\}}(h)\prod_{\{\{\mathrm{v},\mathrm{h}\}\in\mathcal{V}_{\mathrm{V}}\times\mathcal{V}_{\mathrm{H}}\}}\psi_{\{\mathrm{v},\mathrm{h}\}}(v,h) \\
&= \frac{1}{Z}\exp\{-E(\mathrm{v},\mathrm{h})\},
\end{aligned}$$

over all singletons (cliques of size one) and couples (cliques of size two), where the term $-E(\mathrm{v},\mathrm{h})$ reads

$$-\qquad = \log$$

$E(\mathrm{v},\mathrm{h})$ $\left(\prod_{\{\{v\}:v\in\mathcal{V}_V\}} \psi_{\{v\}}(v) \prod_{\{\{h\}:h\in\mathcal{V}_H\}} \psi_{\{h\}}(h) \prod_{\{\{v,h\}\in\mathcal{V}_V\times\mathcal{V}_H\}} \psi_{\{v,h\}}(v,h) \right)$

$= \log \left(\prod_{\{\{v\}:v\in\mathcal{V}_V\}} \psi_{\{v\}}(v) \right)$

$+ \log \left(\prod_{\{\{h\}:h\in\mathcal{V}_H\}} \psi_{\{h\}}(h) \right)$

$+ \log \left(\prod_{\{\{v,h\}\in\mathcal{V}_V\times\mathcal{V}_H\}} \psi_{\{v,h\}}(v,h) \right)$

$= \sum_{\{\{v\}:v\in\mathcal{V}_V\}} \log\left(\psi_{\{v\}}(v)\right)$

$+ \sum_{\{\{h\}:h\in\mathcal{V}_H\}} \log\left(\psi_{\{h\}}(h)\right)$

$+ \sum_{\{\{v,h\}\in\mathcal{V}_V\times\mathcal{V}_H\}} \log\left(\psi_{\{v,h\}}(v,h)\right)$

$= -\sum_{i=1}^{N} E_v(v_i) - \sum_{j=1}^{M} E_h(h_j) - \sum_{i=1}^{N}\sum_{j=1}^{M} E_{v,h}(v_i,h_j),$

which concludes the proof of the lemma. □

The standard example of an RBM is when the random variables follow Bernoulli distribution, i.e., with $\mathcal{X} = \{0,1\}^{|\mathcal{V}|}$. In this case, their energies read

$$E_v(v_i) = -a_i v_i, \quad E_h(h_j) = -b_j h_j, \quad E_{v,h}(v_i,h_j) = -w_{ij} v_i h_j,$$

for some parameters a_i, b_j, w_{ij}, $i = 1,\ldots,N$, $j = 1,\ldots,M$. In particular, for a given v_i, we can write, using Bayes' formula,

$\mathbb{P}(v_i = 1 \mid \mathrm{v}_{v_i'}\mathrm{h})$	$= \dfrac{\mathbb{P}(v_i = 1, \mathrm{v}_{v_i}, \mathrm{h})}{\mathbb{P}(v_i = 1, \mathrm{v}_{v_i}, \mathrm{h}) + \mathbb{P}(v_i = 0, \mathrm{v}_{v_i}, \mathrm{h})}$		
	$= \dfrac{\exp\left(-E(v_i = 1, \mathrm{v}_{v_i}, \mathrm{h})\right)}{\exp\left(-E(v_i = 1, \mathrm{v}_{v_i}, \mathrm{h})\right) + \exp\left(-E(v_i = 0, \mathrm{v}_{v_i}, \mathrm{h})\right)}$.	(5.2.1)	

where we denote $\mathrm{v}_{\mathrm{v_i}}$ the states of all the nodes in $\mathcal{V}\backslash\{v_\mathrm{i}\}$. Now, using the RBM energy lemma, we can single out the energy arising from the particular node v using (5.2.1) as

$$E(v_i, \mathrm{v}_{v_i}, \mathrm{h}) = -\Phi_\mathrm{v}(v_i) - \Psi_\mathrm{v}(\mathrm{v}_{v_i}, \mathrm{h}),$$

where

$$\Phi_\mathrm{v}(v_i) := a_i v_i + \sum_{j=1}^{M} w_{ij} v_i h_j = \left[a_i + \sum_{j=1}^{M} w_{ij} h_j \right] v_i,$$

$$\Psi_\mathrm{v}(\mathrm{v}_{v_i}, \mathrm{h}) := \sum_{k=1(k\neq i)}^{N} a_k v_k + \sum_{j=1}^{M} b_j h_j + \sum_{k=1(k\neq i)}^{N} \sum_{j=1}^{M} w_{kj} v_k h_j.$$

Plugging this into (5.2.1) then yields

$$\begin{aligned}
\mathbb{P}(v_i = 1 | \mathrm{v}_{v_i}, \mathrm{h}) &= \frac{\exp\left(\Phi_\mathrm{v}(v_i = 1) + \Psi_\mathrm{v}\left(\mathrm{v}_{v_i}, \mathrm{h}\right)\right)}{\exp\left(\Phi_\mathrm{v}(v_i = 1) + \Psi_\mathrm{v}\left(\mathrm{v}_{v_i}, \mathrm{h}\right)\right) + \exp\left(\Phi_\mathrm{v}(v_i = 0) + \Psi_\mathrm{v}\left(\mathrm{v}_{v_i}, \mathrm{h}\right)\right)} \\
&= \frac{\exp\left(\Phi_\mathrm{v}(v_i = 1)\right)}{\exp\left(\Phi_\mathrm{v}(v_i = 1)\right) + 1} \\
&= \sigma\left(\Phi_\mathrm{v}(v_i = 1)\right),
\end{aligned}$$

since $\Phi_\mathrm{v}(v_\mathrm{i} = 0) = 0$, where

$$\sigma(x) := \frac{1}{1 + \mathrm{e}^{-x}}$$

is the sigmoid function.

Similarly, we can single out the contribution of the energy on a given hidden node h_j, using the RBM energy lemma:

$$E(\mathrm{v}, h_j, \mathrm{h}_{h_j}) = -\Phi_\mathrm{h}(h_j) - \Psi_\mathrm{h}(\mathrm{v}, \mathrm{h}_{h_j}),$$

where

$$\Phi_{\mathrm{h}}(h_j) := b_j h_j + \sum_{i=1}^{N} w_{ij} v_i h_j = \left[b_j + \sum_{i=1}^{N} w_{ij} v_i \right] h_j,$$

$$\Psi_{\mathrm{h}}(\mathrm{v}, \mathrm{h}_{h_j}) := \sum_{i=1}^{N} a_i v_i + \sum_{k=1(k\neq j)}^{M} b_k h_k + \sum_{i=1}^{N} \sum_{k=1(k\neq j)}^{M} w_{ik} v_i h_k.$$

Plugging this into (5.2.1) then yields

$$\begin{aligned}
\mathbb{P}(h_j = 1 | \mathrm{v}, \mathrm{h}_{h_j}) &= \frac{\exp\left(\Phi_{\mathrm{h}}(h_j = 1) + \Psi_{\mathrm{h}}\left(\mathrm{v}, \mathrm{h}_{h_j}\right)\right)}{\exp\left(\Phi_{\mathrm{h}}(h_j = 1) + \Psi_{\mathrm{h}}\left(\mathrm{v}, \mathrm{h}_{h_j}\right)\right) + \exp\left(\Phi_{\mathrm{h}}(h_j = 0) + \Psi_{\mathrm{h}}\left(\mathrm{v}, \mathrm{h}_{h_j}\right)\right)} \\
&= \frac{\exp\left(\Phi_{\mathrm{h}}(h_j = 1)\right)}{\exp\left(\Phi_{\mathrm{h}}(h_j = 1)\right) + 1} \\
&= \sigma\left(\Phi_{\mathrm{h}}(h_j = 1)\right),
\end{aligned}$$

since again $\Phi_{\mathrm{h}}(h_{\mathrm{j}} = 0) = 0$.

5.2.2 RBM network architecture

As shown above, an RBM is thus a shallow two-layer neural network that operates on stochastic binary activation units. The network forms a bipartite graph connecting stochastic binary inputs (visible units) to stochastic binary feature detectors (hidden units) with no connections between the units within the same layer, as shown in Figure 5.1 [102].

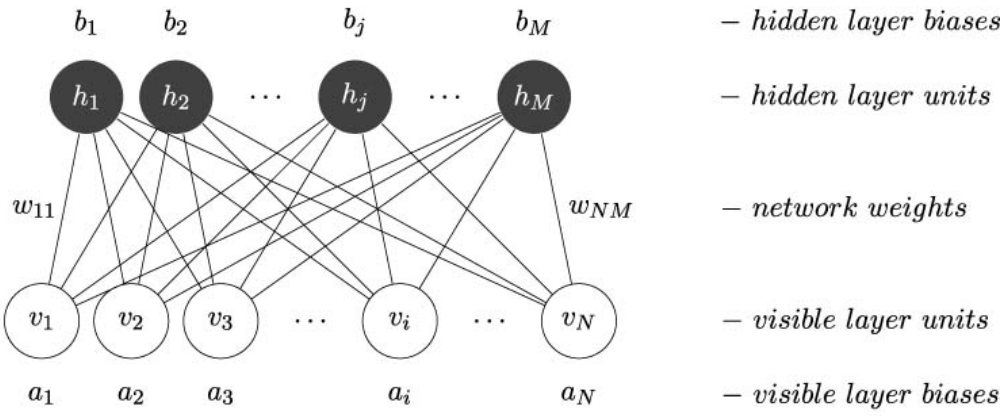


Figure 5.1: Schematic representation of an RBM with the visible layer units (white) and hidden layer units (dark) forming a bipartite graph.

Only the visible layer of the network is exposed to the training dataset and its inputs v := (v_1,…,v_N) flow through the network (forward pass) to the hidden layer, where they are aggregated and added to the hidden layer biases b := (b_1,…,b_M). The hidden layer sigmoid activation function (5.2.1) converts aggregated inputs into probabilities. Each hidden unit then "fires" randomly and outputs a {0,1} Bernoulli random variable with the associated probabilities:

$$\mathbb{P}(h_j = 1|\mathrm{v}) = \sigma\left(b_j + \sum_{i=1}^{N} w_{ij} v_i\right) \quad \text{and} \quad \mathbb{P}(h_j = 0|\mathrm{v}) = 1-\sigma\left(b_j + \sum_{i=1}^{N} w_{ij} v_i\right).$$

The outputs from the hidden layer h := (h_1,…,h_M) then flow back (backward pass) to the visible layer, where they are aggregated and added to the visible layer biases a := (a_1,…,a_N). Similar to the hidden layer, the visible layer sigmoid activation function first translates aggregated inputs into probabilities and then into Bernoulli random variables:

$$\mathbb{P}(v_i = 1|\mathrm{h}) = \sigma\left(a_i + \sum_{j=1}^{M} w_{ij} h_j\right) \quad \text{and} \quad \mathbb{P}(v_i = 0|\mathrm{h}) = 1-\sigma\left(a_i + \sum_{j=1}^{M} w_{ij} h_j\right).$$

Therefore, every unit communicates at most one bit of information. This is especially important for the hidden units since this feature implements the information bottleneck structure, which acts as a strong regulariser [134]. The hidden layer of the network can learn the low-dimensional probabilistic representation of the dataset if the network is organised and trained as an autoencoder.

5.2.3 Sample encoding

Figure 5.2 illustrates the binary representation of an input signal that enters the network through the visible layer. The number of activation units in the visible layer is determined by the number of features we have to encode and the desired precision of their binary representation. For example, if our sample consists of m continuous features and each feature is encoded as an n-digit binary number, the total number of activation units in the visible layer is $m \times n$.

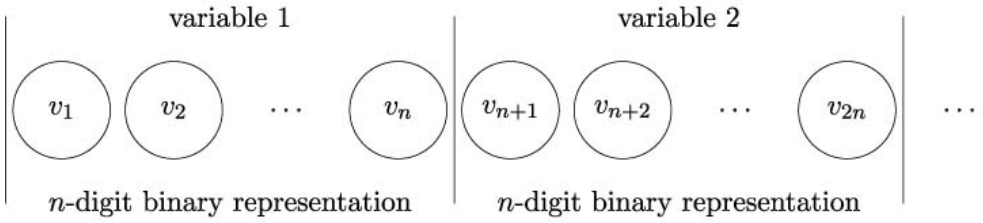


Figure 5.2: Schematic binary encoding of continuous variables.

5.2.4 Boltzmann distribution

The network learns the probability distribution $\mathbb{P}(\mathrm{v},\mathrm{h})$ of the configurations of visible and hidden activation units – the Boltzmann distribution – by trying to reconstruct the inputs from the training dataset (visible unit values) through finding an optimal set of the network weights and biases:

$$\mathbb{P}(\mathrm{v},\mathrm{h}) = \frac{1}{Z}\mathrm{e}^{-E(\mathrm{v},\mathrm{h})},$$

where the energy function reads

$$E(\mathrm{v},\mathrm{h}) = -\sum_{i=1}^{N} a_i v_i - \sum_{j=1}^{M} b_j h_j - \sum_{i=1}^{N}\sum_{j=1}^{M} w_{ij} v_i h_j.$$

Here, Z is the partition function:

$$Z = \sum_{\mathrm{v},\mathrm{h}} \mathrm{e}^{-E(\mathrm{v},\mathrm{h})}.$$

However, we are usually interested either in learning the probability distribution of the visible layer configurations if we want to generate new samples that would have the same statistical properties as the original training dataset, or in learning the probability distribution of the hidden layer configurations if we want to build a deep neural network where the RBM layer performs the feature extraction and dimensionality reduction function. The probabilities of the visible (hidden) states are given by summing over all possible hidden (visible) vectors:

$$\mathbb{P}(\mathrm{v}) = \frac{1}{Z}\sum_{\mathrm{h}} \mathrm{e}^{-E(\mathrm{v},\mathrm{h})} \quad \text{and} \quad \mathbb{P}(\mathrm{h}) = \frac{1}{Z}\sum_{\mathrm{v}} \mathrm{e}^{-E(\mathrm{v},\mathrm{h})}.$$

The most popular training algorithm for RBM, *k-step Contrastive Divergence*, was proposed by Hinton [134, 133]. The algorithm aims to maximise the log probability of a training vector, i.e., to find such network weights and biases that the "energy" function E is minimised for the samples from the training dataset (smaller value of energy corresponds to larger probability of a configuration). The k-step CD algorithm is fully specified in Section 5.3.2 and the interested reader can also find an excellent introduction to the training of RBMs in the work by Fischer and Igel [103].

5.2.5 Extensions of the Bernoulli RBM

The standard Bernoulli RBM setup we considered above restricts the visible layer v to a Bernoulli distribution. In fact, as long as the Hammersley-Clifford theorem holds, we can consider any distribution or any form of energy function. It was shown in [62, 178] for example, that a Bernoulli distribution for the hidden layer combined with a Gaussian distribution for the visible layer are compatible with an energy function of the form

$$E(\mathrm{v},\mathrm{h}) = \sum_{i=1}^{N} \frac{(v_i - a_i)^2}{2\sigma_i^2} - \sum_{j=1}^{M} b_j h_j - \sum_{i=1}^{N}\sum_{j=1}^{M} w_{ij}\frac{v_i h_j}{\sigma_i^2},$$

for some parameters a_i, σ_i, b_j, w_{ij}, $i = 1,\ldots,N$, $j = 1,\ldots,M$. In this case, for any h_j, the conditional probabilities $\mathbb{P}(h_j = 1 \mid \mathrm{v})$ remain of sigmoid form and the conditional distribution of the visible layer is Gaussian as

$$\mathrm{Law}(v_i|\mathrm{h}) = \mathcal{N}\left(a_i + \sum_{j=1}^{M} w_{ij}h_j, \sigma_i^2\right), \quad \text{for each } i = 1,\ldots,N.$$

The RBMs we have considered do not account for time series, i.e., probability structures with temporal dependence. By enlarging the corresponding graph, in particular adding a conditional layer with directed connections to the classical hidden and visible layers, Taylor [280] showed that such dependence can be accounted for.



An RBM is a neural network represented by a bipartite graph. Its power is derived from operating on stochastic binary activation units. It is a generative model that encodes learned probability distribution in its weights and biases and then generates new samples that are statistically indistinguishable from the samples in the original dataset.

If it is organised as an autoencoder with the bottleneck information structure, an RBM is able to learn the low-dimensional representation of the dataset. This property suggests that an RBM can be used as a feature extraction layer in a machine learning pipeline for certain supervised and unsupervised learning problems.

5.3 Training and Running RBM

To build a neural network means to specify the network architecture and training algorithm. Having described the RBM architecture in the previous section, we now outline the training routines.

5.3.1 Training RBM with Boltzmann sampling

The goal of RBM training is to estimate the optimal vector θ of model parameters (weights and biases) so that $\mathbb{P}_\theta(\mathrm{v}) = \mathbb{P}_{\mathrm{data}}(\mathrm{v})$. For a given training sample $\mathrm{v} := (v_1,\ldots,v_\mathrm{N})$, the RBM aims at maximising the log-likelihood function, namely

$$\max_{\theta} \sum_{i=1}^{n} \mathfrak{L}(\theta|v_i),$$

where, for any v,

$$\mathfrak{L}(\theta|\mathrm{v}) = \log(\mathbb{P}(\mathrm{v})) = \log\left(\frac{1}{Z}\sum_{\mathrm{h}} \mathrm{e}^{-E(\mathrm{v},\mathrm{h})}\right) = \log\left(\sum_{\mathrm{h}} \mathrm{e}^{-E(\mathrm{v},\mathrm{h})}\right) - \log\left(\sum_{\mathrm{v},\mathrm{h}} \mathrm{e}^{-E(\mathrm{v},\mathrm{h})}\right).$$

The standard optimisation method, as proposed in [133], is a standard gradient ascent method, i.e., starting from an initial guess θ^0, we update it as

$$\theta^{k+1} = \theta^k + \partial_\theta \sum_{i=1}^{N} \mathfrak{L}(\theta^k | v_i)$$

until we reach good enough convergence. In order to compute it, one first needs to compute the joint probabilities $\mathbb{P}(v_i, h_j)$, which is classically done via Boltzmann (Gibbs) sampling [3], which is possible since we know exactly the conditional distributions.

5.3.2 The Contrastive Divergence algorithm

While training RBMs can be performed with Boltzmann sampling, this is usually prohibitively expensive to run. A more efficient training algorithm, the k-step Contrastive Divergence (CD) algorithm, was proposed in [134].

Algorithm 2: k-step Contrastive Divergence

Result: Weights and biases updates.

Input:

Training minibatch S;

Model parameters a_i, b_j, w_{ij} for $i = 1, \ldots, N, j = 1, \ldots, M$ (before update).

Initialisation: for all $i, j : \Delta w_{ij} = \Delta a_i = \Delta b_j = 0$

for $\mathrm{v} \in S$ **do**

$\mathrm{v}^{(0)} \leftarrow \mathrm{v}$

for $t = 0, \ldots, k-1$ **do**

for $j = 1, \ldots, M$ **do**

sample Bernoulli random variable $h_j^{(t)} \sim \mathbb{P}(h_j | \mathrm{v}^{(t)})$

end

for $i = 1, \ldots, N$ **do**

sample Bernoulli random variable $v_i^{(t+1)} \sim \mathbb{P}(v_i | \mathrm{h}^{(t)})$

end

end

for $i = 1, \ldots, N,\ j = 1, \ldots, M$ **do**

$\Delta w_{ij} \leftarrow \Delta w_{ij} + \eta \left(\mathbb{P}(h_j = 1 | \mathrm{v}^{(0)}) v_i^{(0)} - \mathbb{P}(h_j = 1 | \mathrm{v}^{(k)}) v_i^{(k)} \right)$

end

for $i = 1, \ldots, N$ **do**

$\Delta a_i \leftarrow \Delta a_i + \eta \left(v_i^{(0)} - v_i^{(k)} \right)$

end

for $j = 1, \ldots, M$ **do**

$\Delta b_j \leftarrow \Delta b_j + \eta \left(\mathbb{P}(h_j = 1 | \mathrm{v}^{(0)}) - \mathbb{P}(h_j = 1 | \mathrm{v}^{(k)}) \right)$

end

end

The choice of *k* balances accuracy and speed. For many practical purposes *k* = 1 is an optimal choice, even though the expectations may be biased in this case. However, the bias tends to be small [53]. The network is trained through the updates of weights and biases, which increase the log probability of a training vector and are given by the following expressions:

$$\Delta w_{ij} = \eta \frac{\partial \mathbb{P}(\mathrm{v})}{\partial w_{ij}} = \eta \left(\langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{model}} \right),$$

$$\Delta a_i = \eta \frac{\partial \mathbb{P}(\mathrm{v})}{\partial a_i} = \eta \left(\langle v_i \rangle_{\text{data}} - \langle v_i \rangle_{\text{model}} \right),$$

$$\Delta b_j = \eta \frac{\partial \mathbb{P}(\mathrm{v})}{\partial b_i} = \eta \left(\langle h_j \rangle_{\text{data}} - \langle h_j \rangle_{\text{model}} \right),$$

where $\langle \cdot \rangle$ denote expectations under the distribution specified by the subscript and η is the chosen learning rate. Expectations $\langle \cdot \rangle_{\text{data}}$ can be calculated directly from the training dataset while getting unbiased samples of $\langle \cdot \rangle_{\text{model}}$ requires performing alternating sampling from the model Boltzmann distribution for a long time (this is needed to achieve the state of thermal equilibrium), starting from some randomly initialised state. However, the *k*-step CD method can be used to approximate $\langle \cdot \rangle_{\text{model}}$ with another, easier-to-calculate expectation, as shown in Algorithm 2.

5.3.3 Generation of synthetic samples

Once fully trained, the network can be used to generate new samples from the learned distribution. For example, the RBM can be used as a market generator that produces new market scenarios in the form of the new synthetic samples drawn from the multivariate distribution of the market risk factors encoded in the network weights and biases.

The first step is the generation of a random input: each visible unit is initialised with a randomly generated binary variable. The second step is performing a large number of forward and backward passes between the visible and the hidden layers, until the system reaches a state of *thermal equilibrium*: a state where the initial random vector is transformed into a sample from the learned distribution. The number of cycles needed to reach the state of thermal equilibrium is problem dependent and is a function of network architecture and network parameters (weights and biases). In some cases, the generation of independent samples requires $10^3 - 10^4$ forward and backward passes through the network [173]. The final step is the readout

from the visible layer, which gives us a bitstring, encoding the sample from the target distribution.

Figure 5.3 displays the QQ-plots of the samples drawn from the distributions of daily returns for two stock indices: German DAX and Brazilian BOVESPA. Recall that a quantile-quantile (or QQ) plot is a scatter plot created by plotting two sets of quantiles against one another. If both sets come from the same distribution, all points should lie close to the diagonal. The dataset consists of 536 samples – daily index returns observed between 5 January 2009 and 22 February 2011 (UCI Machine Learning Repository [10, 9]). The "Normal" distribution models daily returns as Normally distributed with a mean and variance that match those from the historical dataset. The "RBM" distribution is a dataset of RBM-generated samples that, ideally, should have exactly the same statistical properties as the original historical dataset. If the samples drawn from two distributions have identical quantiles, the QQ-plots will have all points placed on the diagonal and we can conclude that the two distributions are identical. Figure 5.3 shows that this is indeed the case (with reasonably good accuracy) for the samples from the "Data" and "RBM" distributions, while both demonstrate much heavier tails in comparison with the fitted Normal distribution.

The results shown in Figure 5.3 were obtained with an RBM trained on a dataset of daily returns. Each return from the training dataset was converted into a 12-digit binary number. Every digit of the binary number was treated as a separate binary feature (12 features per index; 24 features in total) – this required placing 24 activation units in the visible layer of the RBM network. The number of hidden units was set to 16. Thus, the network was trained as a strongly regularised autoencoder. The generated returns (in the binary format) were then converted back into their continuous representation. The model was Bernoulli RBM (sklearn.neural_network.BernoulliRBM) from the open source `scikit-learn` package [230] with the following set of parameters:

- n_components = 16 – number of hidden activation units
- learning_rate = 0.0005

- batch_size = 10 – size of the training minibatches
- n_iter = 40000 – number of iterations

The synthetic data generation approach can be formulated as Algorithm 3.

Algorithm 3: Synthetic Data Generation

1: The construction of the binary representation of the original dataset:

   a) A continuous feature can be converted into an equivalent binary representation with the required precision.

   b) An integer feature $x \in \{x_1, \ldots, x_n\}$ can be translated into an N-digit binary number through the standard procedure, where $2^{N-1} \leq \max\limits_{1 \leq j \leq n}(x_j) - \min\limits_{1 \leq j \leq n}(x_j) < 2^N$.

   c) A categorical feature can be binarised either through the one-hot encoding method or following the same procedure as for the integer numbers since categorical values can be enumerated.

   d) The same applies to class labels, both integer and categorical.

2: The training of an RBM on the binary representation of the original dataset with the help of a 1-step CD algorithm.

3: The generation of the required number of new synthetic samples in binary format.

4: For each synthetic data sample: the conversion of the generated binary features into the corresponding categorical, integer, and continuous representations.

5: The generated synthetic dataset is ready to be used for the training of various classifiers and regressors.

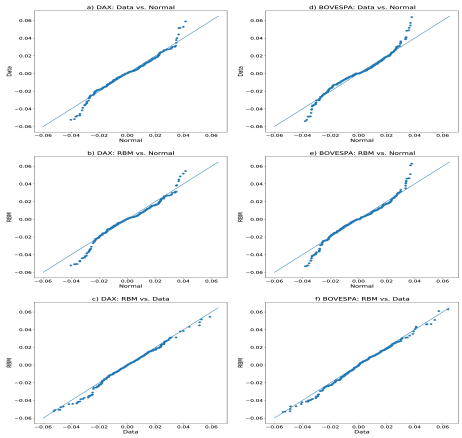


Figure 5.3: QQ-plots of the generated and historical returns. a)-c) DAX. d)-f) BOVESPA. The RBM learns the heavy-tailed empirical distribution of stock index returns.

Kondratyev and Schwarz [173] proposed an RBM-based market generator and investigated its properties on a dataset of daily spot FX log-returns. The time series of four currency pairs' log-returns covered a 20-year time interval (1999-2019), which allowed the RBM to learn the dependence structure of the multivariate distribution and successfully reconstruct linear and rank correlations as well as joint tail behaviour. Also, it was shown that an RBM can be used to perform conditional sampling (e.g., from low-volatility/high-volatility regimes) and achieve the desired degree of autocorrelation by varying the thermalisation parameter. Other productive applications of RBM-based synthetic data generators are data anonymisation, fighting overfitting, and the detection of outliers as demonstrated by Kondratyev, Schwarz, and Horvath [174].



In addition to operating on stochastic binary activation units, the RBM gains extra resistance to overfitting through the autoencoder architecture and being trained with stochastic gradient ascent. This allows RBMs to learn complex multivariate probability distributions from relatively small datasets while avoiding overfitting.

5.4 Quantum Annealing and Boltzmann Sampling

The application of quantum annealing to Boltzmann sampling is based on the direct correspondence between the RBM energy function given by (5.2.4) and the Hamiltonian in quantum annealing. Recall from Chapter 2 that quantum annealing is based on the principles of adiabatic evolution from the initial state at $t = 0$ given by a Hamiltonian $\mathcal{H}_0$ to a final state at $t = T$ given by a Hamiltonian $\mathcal{H}_F$, such that the system Hamiltonian at time $t \in [0,T]$ is given by

$$\mathcal{H}(t) = r(t)\mathcal{H}_0 + (1 - r(t))\mathcal{H}_F,$$

where $r(t)$ decreases from 1 to 0 as t goes from 0 to T. An ideal adiabatic evolution scenario envisages the system always staying in the ground state of $\mathcal{H}(t)$: if the system starts in the ground state of $\mathcal{H}_0$ and the evolution proceeds slowly enough to satisfy the conditions of the quantum adiabatic theorem (Chapter 2), then the system will end up in the ground state of $\mathcal{H}_F$.

In practice, existing quantum annealing hardware does not strictly satisfy the conditions of the quantum adiabatic theorem. Quantum annealers operate at very low temperatures of about 15mK [90], but some residual thermal noise is still present. There is also some amount of cross-talk between the qubits and the chains of physical qubits that represent logical qubits can be broken. Cross-talk is the effect of a desired action on one or more qubits unintentionally affecting one or more other qubits. In some cases, cross-talk is the major source of computational errors. This poses serious issues for quantum annealers solving optimisation problems where the main objective is to find an exact ground state. But some residual amount of thermal and electromagnetic noise is desirable if we want to use a quantum annealer as a sampler.

5.4.1 Boltzmann sampling

The quantum annealer as a sampling engine is based on the central proposal [4] that the distribution of excited states can be modelled as a Boltzmann

distribution:

$$\mathbb{P}(\mathrm{x}) = \frac{1}{Z} \exp\left(-\beta \mathcal{H}_F(\mathrm{x})\right),$$

where β is some parameter (which can be seen as an effective inverse temperature) and Z is the partition function:

$$Z = \sum_{\mathrm{x}} \exp\left(-\beta \mathcal{H}_F(\mathrm{x})\right).$$

If we define the binary vector x to be the concatenation of the visible node vector v and the hidden node vector h:

$$\mathrm{x} := (v_1, v_2, \ldots, v_N, h_1, h_2, \ldots, h_M),$$

then, by comparing (5.2.4) and (5.4.1), we can establish a direct correspondence between the energy function E and the Hamiltonian $\mathcal{H}_{\mathrm{F}}$. Therefore, we can suggest an alternative way of calculating the expectations $\langle\cdot\rangle_{\text{model}}$ formulated as in the following algorithm [4]:

Algorithm 4: Boltzmann Sampling

1: Use the RBM energy function E as the final Hamiltonian $\mathcal{H}_F$.

2: Run quantum annealing K times and collect the readout statistics for $v_i(k)$ and $h_j(k)$, $i = 1, \ldots, N$, $j = 1, \ldots, M$, $k = 1, \ldots, K$.

3: Calculate the unbiased expectations:

$$\langle v_i h_j \rangle_{\text{model}} := \frac{1}{K} \sum_{k=1}^{K} v_i(k) h_j(k),$$

$$\langle v_i \rangle_{\text{model}} := \frac{1}{K} \sum_{k=1}^{K} v_i(k),$$

$$\langle h_j \rangle_{\text{model}} := \frac{1}{K} \sum_{k=1}^{K} h_j(k).$$

There are two main motivations for using quantum annealing to perform Boltzmann sampling as described in Algorithm 4. First, it bypasses the need for running the Contrastive Divergence algorithm (Algorithm 2), which only provides approximations to the expectations $\langle \cdot \rangle_{\text{model}}$ (even though these approximations can be sufficiently accurate). Second, the anneal time needed to generate a new sample from the Boltzmann distribution is of the order of ~1 microsecond regardless of the graph size. This is not the case with the classical RBM, where it is often necessary to perform thousands of forward and backward passes through the network before a new independent sample from the Boltzmann distribution encoded in the network weights and biases can be read out [173]. For large RBM graphs, it can easily take tens of milliseconds on standard hardware. Thus, we have two avenues of exploring the potential quantum advantage offered by quantum annealing for Boltzmann sampling: accuracy and speedup.

5.4.2 Mapping

The first step in performing Boltzmann sampling on a quantum annealer is the mapping of the RBM onto the quantum annealing hardware graph. We start with writing an expression for the RBM energy function E in the following form:

$$E(\mathrm{v}, \mathrm{h}) = E(\mathrm{x}) = \beta \mathrm{x}^T Q \mathrm{x}.$$

Here, Q is the $(N + M) \times (N + M)$ matrix whose elements are RBM weights and biases:

$$Q = \frac{1}{\beta} \left[\begin{array}{cccc|cccc} a_1 & 0 & \dots & 0 & w_{11} & w_{12} & \dots & w_{1M} \\ 0 & a_2 & \dots & 0 & w_{21} & w_{22} & \dots & w_{2M} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_N & w_{N1} & w_{N2} & \dots & w_{NM} \\ \hline 0 & 0 & \dots & 0 & b_1 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & b_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & b_M \end{array}\right].$$

Quantum annealers operate on spin variables {−1,+1} instead of binary variables {0, 1}. The vector of binary variables x can be transformed into the vector of spin variables s using

$$\mathrm{x} \longrightarrow \mathrm{s} = 2\mathrm{x} - \mathbf{1},$$

and we obtain the following expression for the RBM energy:

$$E = -\sum_{i=1}^{N} g_i s_i - \sum_{j=N+1}^{N+M} g_j s_j - \sum_{i=1}^{N} \sum_{j=N+1}^{N+M} J_{ij} s_i s_j - \text{const} = E_{\text{Ising}} - \text{const},$$

where, for $i = 1,\dots,N$ and $j = N + 1,\dots,N + M$,

$$g_i := \frac{a_i}{2} + \frac{1}{4} \sum_{j=N+1}^{N+M} w_{ij}, \quad g_j := \frac{b_j}{2} + \frac{1}{4} \sum_{i=1}^{N} w_{ij}, \quad J_{ij} := \frac{1}{4} w_{ij},$$

and $(s_i)_{i=1,\dots,N}$ are spin variables corresponding to the visible nodes and $(s_j)_{j=N+1,\dots,N+M}$ are spin variables corresponding to the hidden nodes.

We can ignore the constant term in the RBM energy expression (5.4.2) since the same factor will appear in the numerator and denominator of $\mathbb{P}(v,h)$. Thus, we have

$$\langle v_i h_j \rangle_{\text{model}}^{E_{\text{Ising}}} = \langle v_i h_j \rangle_{\text{model}}^{E}.$$

To express the Ising Hamiltonian using a quantum mechanical description of spins, we replace the spin variables with their respective Pauli operators:

$$\mathcal{H}_{\text{Ising}} = -\sum_{i=1}^{N} g_i \sigma_z^i - \sum_{j=N+1}^{N+M} g_j \sigma_z^j - \sum_{i=1}^{N} \sum_{j=N+1}^{N+M} J_{ij} \sigma_z^i \sigma_z^j,$$

with σ_z^{i} being the usual Pauli matrix representation for an Ising quantum spin. With the initial Hamiltonian given by

$$\mathcal{H}_0 = \sum_{i=1}^{N+M} \sigma_x^i,$$

the time-dependent Hamiltonian (5.4) takes the form

$$\mathcal{H}(t) = r(t)\mathcal{H}_0 + (1 - r(t))\mathcal{H}_{\text{Ising}}.$$

5.4.3 Hardware embedding and parameters optimisation

In the standard programming practices of existing quantum annealers, each spin variable s_{i} should ideally be assigned to a specific chip element, a superconducting flux qubit, modelled by a quantum two-level system that could represent the quantum Hamiltonian

$$\mathcal{H}_{\text{local}} = \sum_i g_i \sigma_z^i.$$

While each qubit supports the programming of the g_{i} terms, the J_{ij} parameters can then be implemented energetically through inductive elements, meant to represent

$$\mathcal{H}_{\text{couplers}} = \sum_{ij} J_{ij} \sigma_z^i \sigma_z^j,$$

if and only if the required circuitry exists between qubits *i* and *j*, which cannot be manufactured too far apart in the spatial layout of the processor

due to engineering considerations [296]. In other words, $J_{ij} = 0$ unless $(i,j) \in G$, where G is a particular quantum annealing graph (e.g., *Chimera* or *Pegasus* graphs in the case of D-Wave quantum annealers).

It would be straightforward to embed the final Hamiltonian (5.4.2) on the quantum chip had all the physical qubits been connected to each other. Unfortunately, this is not the case. The existing quantum annealers have rather limited qubit connectivity. For example, in the case of the *Chimera* (*Pegasus*) graph, a physical qubit is connected with a maximum of six (fifteen) other physical qubits.

To get around this restriction, the standard procedure is to employ the minor-embedding compilation technique for fully connected graphs. By means of this procedure, we obtain another Ising form, where qubits are arranged in ordered 1D chains (forming the *logical* qubits that represent the spin variables) interlaced on the quantum annealer graph:

$$\mathcal{H}_{\text{Ising}} = -\sum_{i=1}^{N} |J_F| \left[\sum_{c=1}^{N_c-1} \sigma_z^{ic} \sigma_z^{i(c+1)} \right] - \sum_{j=N+1}^{N+M} |J_F| \left[\sum_{c=1}^{N_c-1} \sigma_z^{jc} \sigma_z^{j(c+1)} \right]$$

$$- \sum_{i=1}^{N} \frac{g_i}{N_c} \left[\sum_{c=1}^{N_c} \sigma_z^{ic} \right] - \sum_{j=N+1}^{N+M} \frac{g_j}{N_c} \left[\sum_{c=1}^{N_c} \sigma_z^{jc} \right]$$

$$- \sum_{i=1}^{N} \sum_{j=N+1}^{N+M} J_{ij} \left[\sum_{c_i,c_j=1}^{N_c} \delta_{ij}^{G}(c_i, c_j) \sigma_z^{ic_i} \sigma_z^{jc_j} \right] .$$

In (5.4.3), we explicitly isolate the encoding of the *logical* quantum variable: the classical binary variable s_i is associated with N_c Ising spins σ_z^{ic}, ferromagnetically coupled directly by strength J_F, forming an ordered 1D chain subgraph of G. The value of J_F should be strong enough to correlate the value of the magnetisation of each individual spin if measured in the computational basis ($\langle \sigma_z^{ic} \rangle = \langle \sigma_z^{i(c+1)} \rangle$).

In (5.4.3) and (5.4.3), we encode the Ising Hamiltonian (5.4.2) through our extended set of variables: the local field g_i is evenly distributed across all

qubits belonging to the logical chain i, and each coupler J_{ij} is active only between one specific pair of qubits $(\sigma_z^{ic_i^\star}, \sigma_z^{jc_j^\star})$, which is specified by the adjacency check function $\delta_{ij}^{G}(c_i, c_j)$, which assumes a unit value only if $(c_i = c_i^\star)$ and $(c_j = c_j^\star)$, and is zero otherwise.

Given this particular embedding scheme, we can turn our attention to finding an optimal value for the parameter β in (5.4.2), which can only be done experimentally. Since the final Hamiltonian is programmed on the quantum annealer using dimensionless coefficients, the parameter β cannot be expressed in the usual form $1/kT$, where k is the Boltzmann constant and T the effective temperature. Instead, it should be viewed as an empirical parameter that depends on the network architecture, the embedding scheme, and the physical characteristics of the quantum annealer (such as the operating temperature, the anneal time, the energy scale of the superconducting flux qubit system, etc.).

The experimental approach of estimating β consists of the following five steps [4]:

1. Construct an RBM.
2. Map the RBM to a final Hamiltonian assuming a particular value of β (Alg. 4-Step 1).
3. Run quantum annealing (Alg. 4-Step 2).
4. Compute the model expectations using the quantum samples (Alg. 4-Step 3).
5. Compare the resulting expectations with the "correct" benchmark values (e.g., obtained with the classical CD algorithm).

This process is repeated for different choices of β. The value of β that gives the best fit can then be used for the given RBM architecture. As noted in [4], even with the optimal settings for β, the estimates of the model expectations will still have some error. However, in comparison to the noise associated with the Boltzmann sampling in the Contrastive Divergence algorithm, this may be sufficient to estimate the gradients in (5.3.2), (5.3.2), and (5.3.2).

5.4.4 Generative models

The main application of the Boltzmann sampling we've considered so far is in providing an unbiased estimate of the model expectations as specified in Algorithm 4. Once fully trained with the help of quantum annealing, an RBM can be used in a conventional classical way to generate new synthetic samples from the learned probability distribution. In this case, quantum annealing is only used as a subroutine in the hybrid quantum-classical training protocol.

However, it is possible to use a quantum annealer as a generator in its own right. Rather than assisting in training the classical RBM, a quantum annealer can output the binary representation of the continuous samples as per the distribution encoded in the final Hamiltonian (5.4.2). The Quantum Variational Autoencoder [162] is another example of a QBM that can be trained end to end by maximising a well-defined cost function: a quantum lower bound to a variational approximation of the log-likelihood.



Botzmann sampling is the key element of RBM training and the generation of new samples. Quantum annealing can provide orders of magnitude speedup by replacing classical Boltzmann sampling with quantum sampling.

5.5 Deep Boltzmann Machine

Deep Boltzmann Machines (DBMs) can be constructed from several RBMs where the hidden layer of the first RBM becomes the visible layer of the second, and so on, as shown in Figure 5.4.

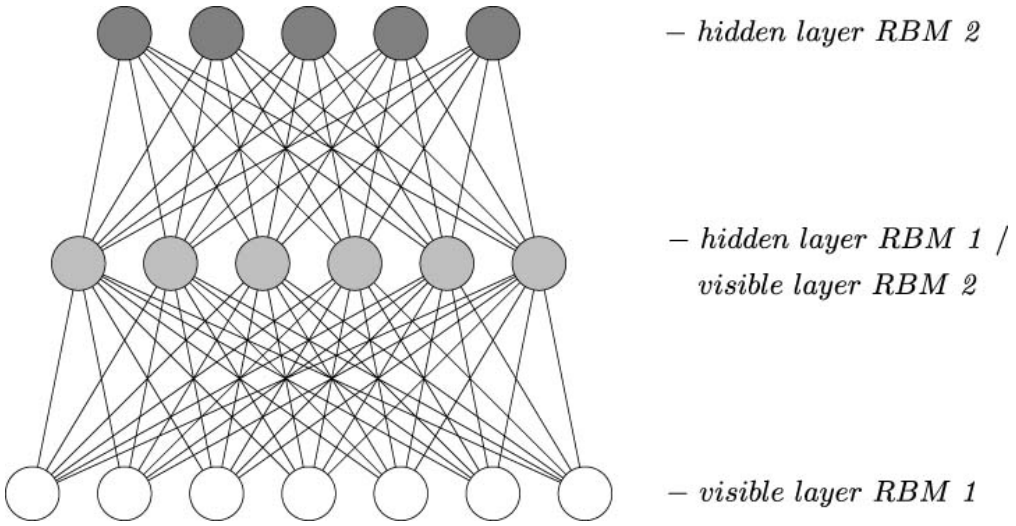


Figure 5.4: Schematic representation of a DBM.

A DBM can be trained layer by layer, one RBM at a time. This will result in a powerful generative model capable of learning complex multivariate distributions and dependence structures. However, the generative training of the DBM can be used as the first step towards building a discriminative model if the training dataset samples are labelled. In this case all DBM weights and biases found with the help of either CD or quantum Boltzmann sampling algorithms are seen as initial values of the weights and biases of the corresponding feedforward neural network. The discriminative model will consist of all the layers of the original DBM with an extra output layer performing assignment of the class labels. The discriminative model can be fine tuned through the standard backpropagation of the error algorithm.

5.5.1 Training DBMs with quantum annealing

The generative training of DBMs can be seen as a pre-training of the discriminative model. Figure 5.5 provides a schematic illustration of the hybrid quantum-classical training process.

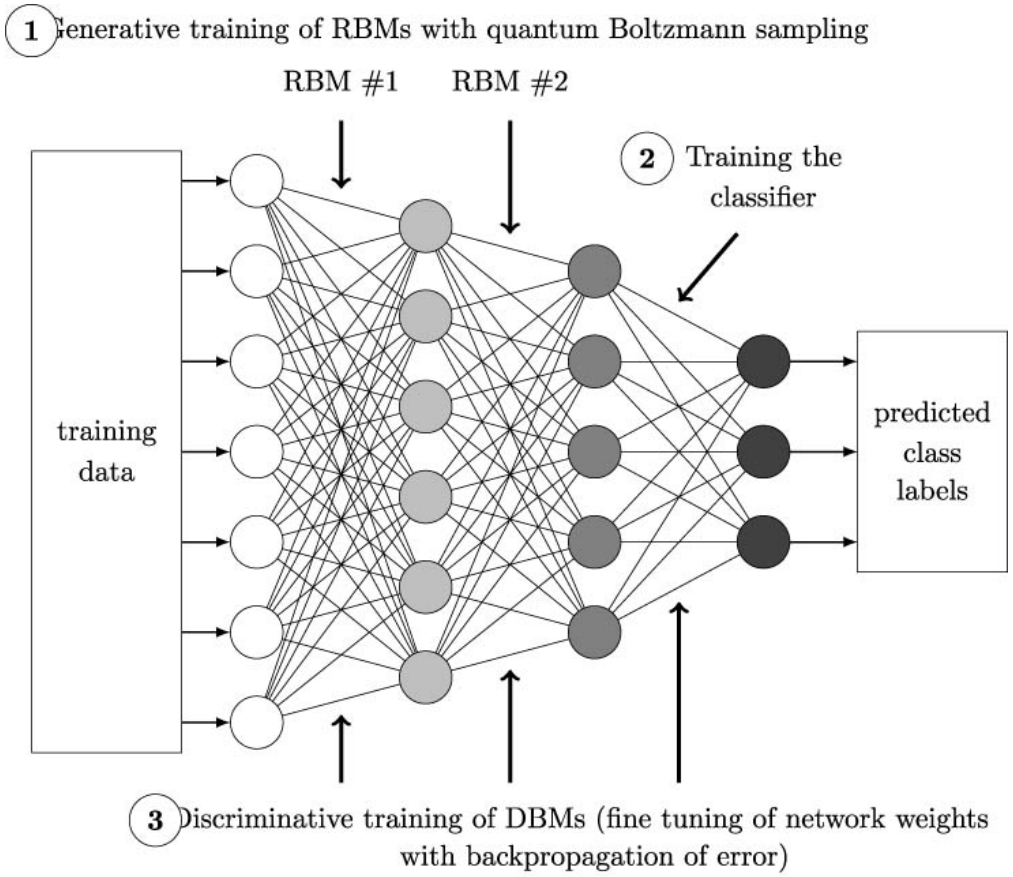


Figure 5.5: Generative and discriminative training of a DBM.

In the DBM training scheme shown in Figure 5.5, only Step 1 relies on quantum annealing. Steps 2 and 3 are completely classical. Step 3 is optional: without it we have a standard machine learning "pipeline" where one or several RBMs (Step 1) perform "feature extraction" by building a low-dimensional representation of the samples in the dataset, thus helping the discriminative model (Step 2) to achieve better classification results.

5.5.2 A DBM pipeline example

The pipeline approach can be illustrated using the popular "King+Rook vs. King+Pawn" dataset from the UCI Machine Learning Repository [262, 263]. The task is to classify the end game positions with the black pawn one move from queening and the white side (King+Rook) to move. The possible outcomes are "white can win" (Class 1) and "white cannot win" (Class 0). The board is described by 36 categorical attributes that can be encoded as 38

binary variables. The dataset consists of 3,196 samples (white can win in 52% of all cases in the dataset).

The `scikit-learn` package provides all the necessary components for building the classical part of a DBM pipeline. The pipeline itself can be constructed with the help of `sklearn.pipeline.make_pipeline`. The DBM is constructed from two RBMs implemented with the help of `sklearn.neural_network.BernoulliRBM`. RBM #1 has 38 nodes in the visible layer and 30 nodes in the hidden layer; RBM #2 has 30 nodes in the visible layer and 20 nodes in the hidden layer. The exact pipeline configuration is as follows (all other parameters were set at their default values):

RBM #1	RBM #2	MLP Classifier
n_components = 30	n_components = 20	hidden_layer_sizes = (20)
learning_rate = 0.00025	learning_rate = 0.00025	activation = 'tanh'
batch_size = 10	batch_size = 10	solver = 'adam'
n_iter = 100000	n_iter = 100000	alpha = 0.1
		max_iter = 5000

Table 5.1: Configuration of the DBM pipeline for the "King+Rook vs. King+Pawn" classification problem.

Thus, both RBMs are trained as autoencoders: the DBM translates each 38-feature sample into its 20-feature low-dimensional representation. These new "extracted" features, ideally, should have higher predicting power in comparison with the original features, assuming that both RBMs learned the main characteristics and dependence structure of the dataset and stripped away the noise or the less important characteristics. The discriminator is `sklearn.neural_network.MLPClassifier` with 20 tanh activation units in its single hidden layer.

With this setting, the DBM achieves the following out-of-sample classification results (with the dataset split 70:30 into the training and testing datasets using `sklearn.model_selection.train_test_split`):

- Classification accuracy: 95.2%

This compares favourably with, for example, an ensemble learning classifier such as random forest (`sklearn.ensemble.RandomForestClassifier`). The random forest classifier with the number of estimators set at 1,000 and the maximum depth set equal to 5 has the following out-of-sample classification results:

- Classification accuracy: 94.9%



The architecture of DBMs allows them to be trained as either generative or discriminative models. In both cases, Boltzmann sampling can play an important role in improving their performance.

Summary

In this chapter, we learned about energy-based models – a special class of powerful generative models. We learned how to build, train, and run RBMs in order to generate synthetic samples that are statistically indistinguishable from the original training dataset.

We familiarised ourselves with the Boltzmann sampling and Contrastive Divergence algorithms. Boltzmann sampling can be efficiently performed on NISQ-era quantum annealers that may improve the quality of the model and achieve orders of magnitude of speedup in generating new samples.

We learned how to combine individual RBMs together to construct a DBM. Quantum annealing can be productively applied to the pre-training of a DBM before it is fine tuned as a deep feedforward neural network classifier.

Finally, we explored the possibility of using RBMs and DBMs as the first model in the machine learning pipeline for denoising and feature extraction.

In the next chapter, we will shift our attention to gate model quantum computing. We will start with the concept of a classical binary digit (bit) and classical logic gates before introducing their quantum counterparts: the

quantum binary digit (qubit) and one-qubit/multi-qubit quantum logic gates and quantum circuits.

Join our book's Discord space

Join our Discord community to meet like-minded people and learn alongside more than 2000 members at: https://packt.link/quantum



Part II
Gate Model Quantum Computing

6
Qubits and Quantum Logic Gates

A computation can be broadly defined as a transformation of one memory state into another. Put slightly differently, a computation is a function that transforms information [281]. In the case of classical digital computing, the fundamental memory unit is a *binary digit* (bit) of information. Functions that operate on bits of information are called *logic gates*. Logic gates are Boolean functions that can be combined into *circuits* capable of performing addition and multiplication, as well as more complex operations. In logic gates, the number of output bits does not have to be the same as the number of input bits.

A computation may seem to be an abstract mathematical concept but it always requires some physical system in order to be executed. It does not matter what this physical system is: billiard balls, electric switches, transistors, or anything else – the computation is substrate independent. However, it is always some physical process that changes the state of the system in a controlled way.

Classical digital computing requires some physical implementation of two distinct deterministic states (usually denoted as 0 and 1) and a set of gates that perform controlled transitions between them. In the following sections, we will see how classical digital computation can be implemented, what set of basis operations is required, and how the logic of classical computation can be extended to more general logic of quantum computation, of which classical computing is just a special case.

6.1 Binary Digit (Bit) and Logic Gates

In this section, we rapidly review classical logic gates and their universality, in order to draw a parallel later to quantum gates.

6.1.1 Logic gates

A logic gate is an implementation of a Boolean function, a logical operation performed on one or more binary inputs that produces a single binary output. Logic gates are represented by their *truth tables*. A truth table has one column for each input variable, and one final column showing all of the possible results of the

logical operation that the table represents. Each row of the truth table contains one possible configuration (a single bit or a bitstring) of the input variables, and the result of the operation for those values.

Figures and Tables 6.1, 6.2, 6.3, and 6.4 are schematic circuit representations of the AND, OR, NAND (not AND), and XOR (exclusive OR) logic gates as well as their corresponding truth tables.

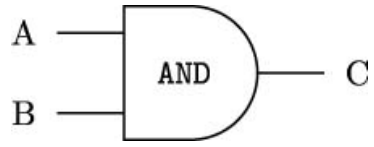


Figure 6.1: AND gate diagram and truth table.

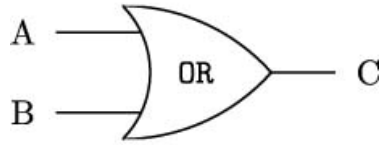


Figure 6.2: OR gate diagram and truth table.

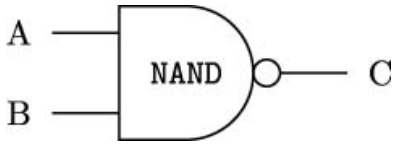


Figure 6.3: NAND gate diagram and truth table.

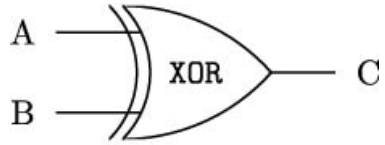


Figure 6.4: XOR gate diagram and truth table.

6.1.2 NAND as a universal logic gate

Logic gates can be combined into circuits where the output of one is the input of another. This allows us i) to implement more complex operators than basic Boolean functions and ii) to implement all necessary Boolean functions using only a small number of easy-to-build logic gates. For example, all Boolean functions can be constructed using only a NAND gate and a fan-out operation. This makes NAND a *universal* gate in classical computing. Figure 6.5 illustrates this by

presenting the decomposition of four basic logic gates (NOT, AND, OR, XOR) into circuits consisting only of NAND gates.

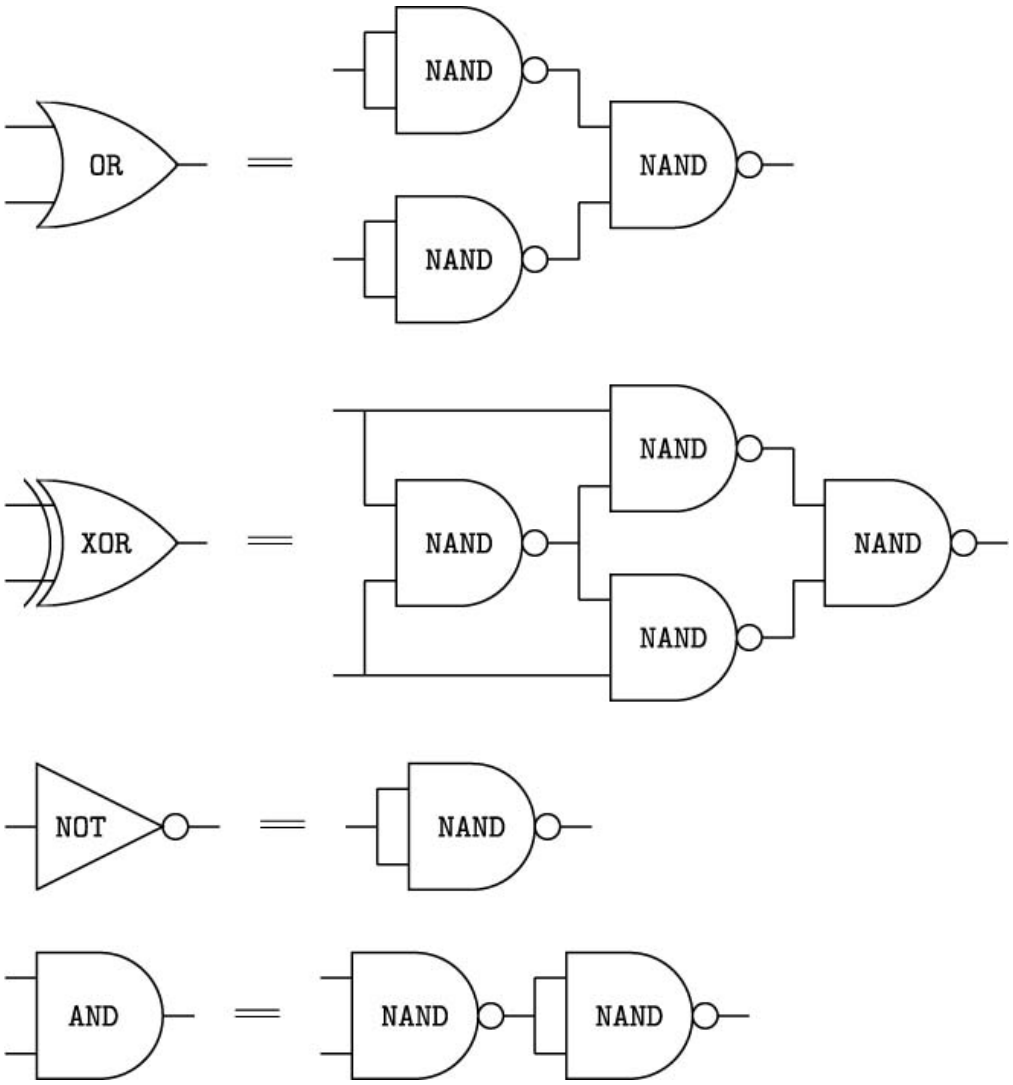


Figure 6.5: Examples of logic gates decomposition into NAND gates and fan-out operations.

6.1.3 Building an addition operator from the NAND gates

Figure 6.6 shows how NAND and XOR gates can be combined into a circuit that implements the basic addition operator. As we know, the XOR gate itself can be constructed from the combination of NAND gates. The addition operator takes three 1-bit binary numbers as inputs and outputs two 1-bit binary numbers that can be read as a 2-bit bitstring (a 2-bit binary number). This 2-bit binary number can be translated into its integer number representation – an integer number between 0 and 3 as shown in the truth table (Table 6.1).

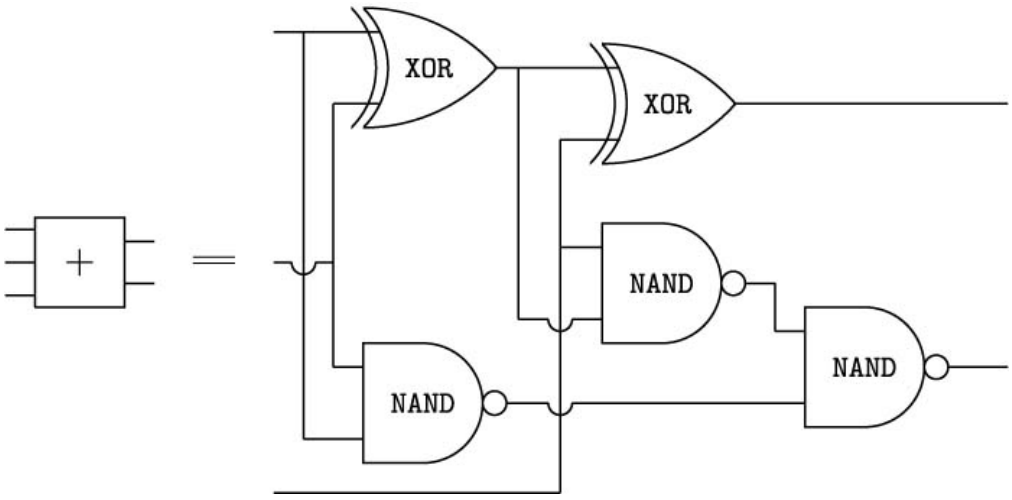


Figure 6.6: Addition operator: the input is three 1-bit binary numbers and the output is a single 2-bit binary number.

input 1	input 2	input 3	output 1	output 2	binary	integer
0	0	0	0	0	00	0
0	0	1	0	1	01	1
0	1	0	0	1	01	1
1	0	0	0	1	01	1
0	1	1	1	0	10	2
1	0	1	1	0	10	2
1	1	0	1	0	10	2
1	1	1	1	1	11	3

Table 6.1: Addition operator truth table.



Computation is a transformation of one memory state into another. Functions that perform such transformations are called logic gates. Logic gates are fully specified by their truth tables. A universal logic gate is one from which all other Boolean functions can be constructed. We only need to find an efficient physical realisation of a universal gate in order to perform computations of arbitrary complexity.

6.2 Physical Realisations of Classical Bits and Logic Gates

We have so far defined bits and classical logic gates from a theoretical computer science point of view. We now provide an overview of the most efficient hardware techniques used to effectively implement such operations.

6.2.1 Implementation of the NAND gate

The NAND gate (together with the fan-out operator) is a universal gate in classical digital computing. Therefore, it should be sufficient to find a practical physical implementation of the NAND Boolean function in order to build a universal computer. Figure 6.7 displays several possible realisations of the NAND gate using different technologies, from electrical switches to semiconductors.

Relay Logic: Switches are interpreted as bits with 0 = open and 1 = closed. When switches A and B are both closed, an electromagnet opens switch C. If either or both of switches A and B are open, the circuit is broken and an electromagnet cannot open switch C.

Resistor-Transistor (RT) Logic: Voltages are interpreted as bits with 0 = zero volts and 1 = 3 volts. When wires A and B are both at +3 volts, the two transistors conduct electricity and wire C drops to zero volts. If either or both of inputs A and B are zero volts, the corresponding transistors do not conduct and output C stays at +3 volts.

Complementary Metal-Oxide-Semiconductor (CMOS) Logic: Similar to RT logic, voltages are interpreted as bits with 0 = zero volts and 1 = 3 volts. The PMOS transistor is open when the input is 1 (+3 volts) and closed when the input is 0 (zero volts). NMOS is the logical opposite of PMOS. The PMOS circuit is placed between the voltage and the output. The NMOS circuit is placed between the output and the ground.

If both of the A and B inputs are high, both the NMOS transistors will conduct, neither of the PMOS transistors will conduct, and a conductive path will be established between the output, C, and the ground thus bringing the output low. If both of the A and B inputs are low, then neither of the NMOS transistors will conduct, while both of the PMOS transistors will conduct, establishing a conductive path between the output and the voltage source, bringing the output

high. If either of the A or B inputs is low, one of the NMOS transistors will not conduct, one of the PMOS transistors will, and a conductive path will be established between the output and the voltage source, bringing the output high. Therefore, the circuit implements the NAND gate as the only configuration of the two inputs that results in a low output is when both are high.

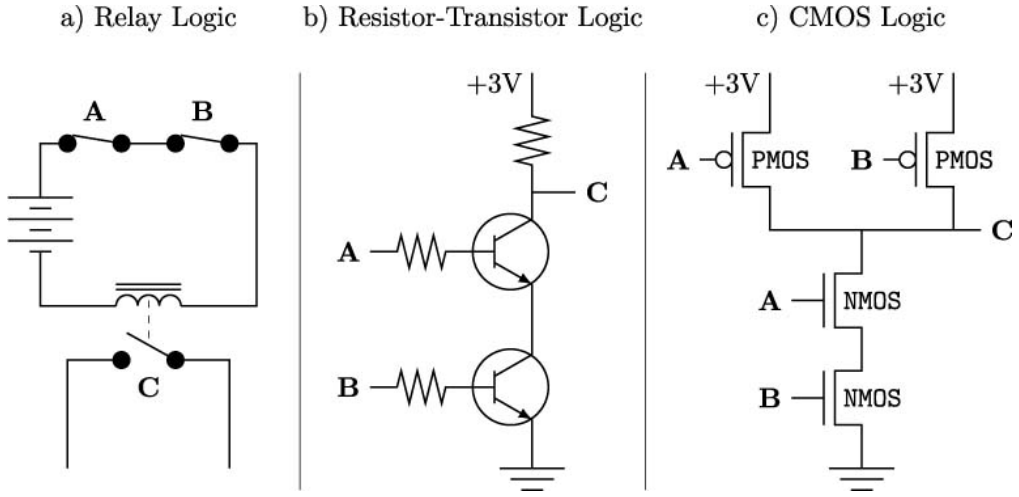


Figure 6.7: Physical realisations of the NAND gate.

6.2.2 Implementation of the RAM memory cell

Random Access Memory (RAM) is used to store instructions and data currently used by the CPU. It is called *volatile memory* in the sense that it is wiped out when the computer is switched off. RAM may consist of many billions of elementary *memory cells*, with each memory cell being able to store one bit of information.

Now that we know how to build a universal logic gate, we can try to design a circuit that would implement the elementary memory unit we need to build RAM. For example, Figure 6.8 shows how the memory cell can be built from four NAND logic gates.

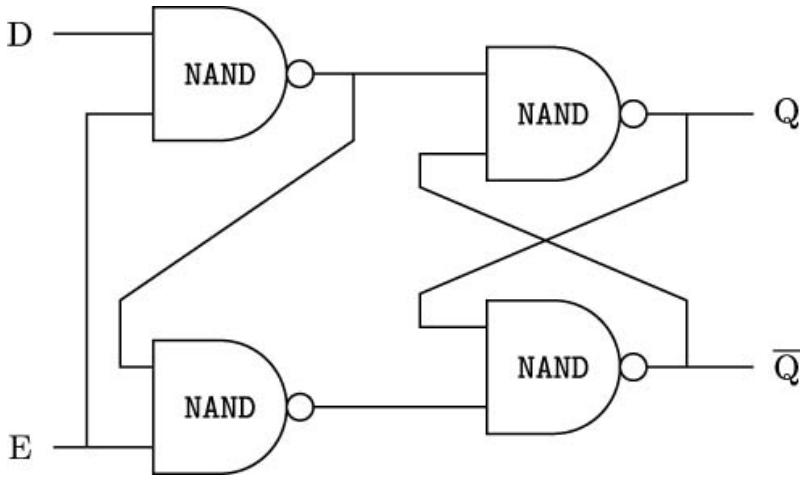


Figure 6.8: Construction of the elementary memory cell from NAND gates.

D	E	Q	Q
0	1	0	1
1	1	1	0
0	0	Q	Q
1	0	Q	Q

Table 6.2: Memory cell truth table.

The circuit in Figure 6.8 has two input pins, D (Data) and E (Enabler), and two output pins, Q and Q (NOT Q). The truth table (Table 6.2) of the memory cell circuit explains how it works:

- When the enabler input E is set to 1, the output Q can be set to the data input D.
- When the enabler input E is set to 0, the output Q cannot be changed – it retains its value.

These are the key features that allow the circuit to serve as a memory cell.



Computation is substrate independent. Any physical system that can exist in two discrete, stable states with controlled transitions between them can be used to implement gate model digital computing. At the same time, some of the implementations are more efficient (faster, cheaper, more reliable) than others.

These classical logic gates provide a natural framework to understand their quantum formulations, which we'll investigate now.

6.3 Quantum Binary Digit (Qubit) and Quantum Logic Gates

Quantum bits and quantum logic gates are the quantum computing counterparts of classical bits and logic gates. While they share common features, the quantum aspects yield a multitude of specific properties, which are the subject of this section.

6.3.1 Computation according to the laws of quantum mechanics

Classical logic gates operating on bits implement Boolean functions, forming the basis of digital classical computing. As we have seen, there are many possible physical implementations of a classical bit – a system that has two distinct, stable states with controlled transitions between them. What can we say about such a system from the quantum mechanical point of view?

As we know from Chapter 1, any such system may exist in a superposition of states, and the state of a qubit $|\psi\rangle$ is described by the expression

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,,$$

where α and β are complex numbers satisfying

$$|\alpha|^2 + |\beta|^2 = 1.$$

The coefficients α and β are *probability amplitudes*. Any attempt to *measure* the state $|\psi\rangle$ results in getting $|0\rangle$ with probability $|\alpha|^2$, and $|1\rangle$ with probability $|\beta|^2$. The measurement consists in coupling the quantum system to the environment, which collapses the superposition. After the measurement, the system is in the measured state and further measurements on the same basis will always yield the same result.

Since the qubit state $|\psi\rangle$ is described by two complex probability amplitudes satisfying (6.3.1), we can say that the state of a qubit is a unit vector in the two-

dimensional complex vector space. In other words, the state $|\psi\rangle$ can be written as the vector

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \alpha \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \beta \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

This means that the basis states $|0\rangle$ and $|1\rangle$ are represented by the standard orthonormal basis vectors

$$|0\rangle := \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad |1\rangle := \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

The standard orthonormal basis $|0\rangle$ and $|1\rangle$ is not the only possible choice of the basis vectors. Any pair of *linearly independent* unit vectors $|u\rangle$ and $|v\rangle$ from the complex two-dimensional vector space can serve as a basis:

$$\alpha\,|0\rangle + \beta\,|1\rangle = \alpha'\,|u\rangle + \beta'\,|v\rangle\,.$$

For example, we can use the Hadamard basis $\{|+\rangle,|-\rangle\}$ defined by

$$|+\rangle := \frac{1}{\sqrt{2}}\,|0\rangle + \frac{1}{\sqrt{2}}\,|1\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \quad \text{and} \quad |-\rangle := \frac{1}{\sqrt{2}}\,|0\rangle - \frac{1}{\sqrt{2}}\,|1\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix}.$$

The basis is determined by the measurement process or the physical realisation of the quantum computer [80].

It is important to specify the choice of the basis. For example, the vector

$$\begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

measured in the standard orthonormal basis (the *computational basis*) gives outcomes $|0\rangle$ and $|1\rangle$ with equal probability 1/2. Measured in the Hadamard basis, it gives the outcome $|+\rangle$ with probability 1.

The state of a two-qubit system can be represented by a unit vector in the four-dimensional complex vector space. In this case, the standard orthonormal basis consists of four orthonormal unit vectors

$$|00\rangle := \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad |01\rangle := \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad |10\rangle := \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad |11\rangle := \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix},$$

and the system state is described by four probability amplitudes:

$$|\psi\rangle = \alpha\,|00\rangle + \beta\,|01\rangle + \gamma\,|10\rangle + \delta\,|11\rangle,$$

with $\alpha,\beta,\gamma,\delta \in \mathbb{C}$ such that $|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1$. The basis vectors (6.3.1) of the two-qubit states are constructed as *tensor products* of the individual qubit basis vectors:

$$|00\rangle = |0\rangle \otimes |0\rangle = \begin{bmatrix} 1 \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ 0 \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad |01\rangle = |0\rangle \otimes |1\rangle = \begin{bmatrix} 1 \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ 0 \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix},$$

$$|10\rangle = |1\rangle \otimes |0\rangle = \begin{bmatrix} 0 \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ 1 \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad |11\rangle = |1\rangle \otimes |1\rangle = \begin{bmatrix} 0 \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ 1 \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

Generally, the n-qubit system can exist in any superposition of the 2^n basis states and requires 2^n probability amplitudes to be fully specified.



Computation is a transformation of the memory state. The qubit states are transformed by an application of quantum logic gates. Quantum logic gates are unitary linear operators that are represented by unitary matrices. The action of a quantum logic gate on a specific quantum state is found by multiplying the unitary

matrix representing the gate by the vector representing the state. The result is the new quantum state.

6.3.2 Qubit

It is convenient to visualise the state of a qubit as a point on the unit sphere, named the Bloch sphere after physicist Felix Bloch. Every point on the Bloch sphere is uniquely specified by two angles, $\theta \in [0,\pi]$ and $\varphi \in [0,2\pi]$, as shown in Figure 6.3.2.

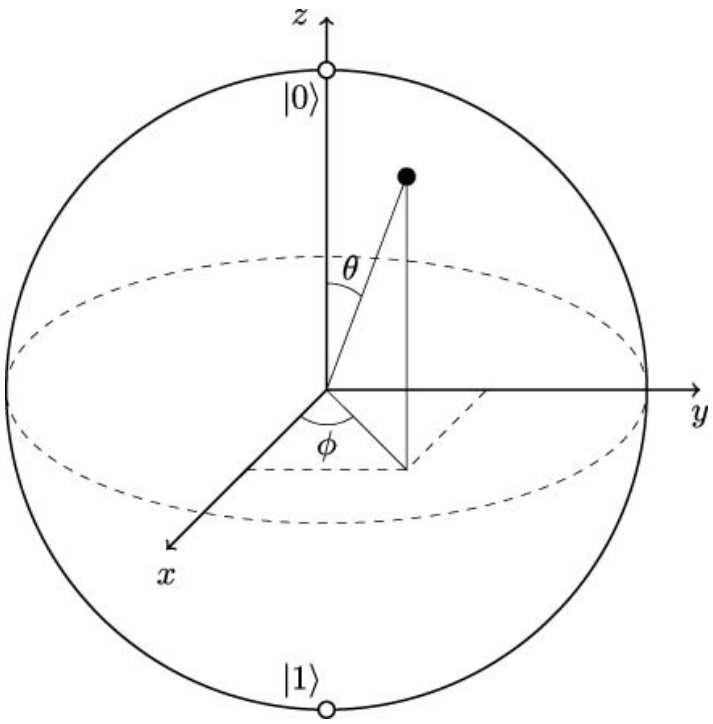


Figure 6.9: Quantum state $|\psi\rangle$ on the Bloch sphere.

With the mapping

$$\alpha = \cos\left(\frac{\theta}{2}\right), \quad \beta = e^{i\phi}\sin\left(\frac{\theta}{2}\right),$$

we obtain the canonical representation of the qubit state:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) \\ e^{i\phi}\sin\left(\frac{\theta}{2}\right) \end{bmatrix}.$$

A transformation of the qubit state can be visualised as a transition from one point on the Bloch sphere to another. Therefore, the unitary matrix (quantum logic gate) that performs this transformation can be seen as a rotation operator and we can speak about *rotation* as a synonym of gate operation and *rotation angles* as gate parameters.

6.3.3 One-qubit quantum logic gates

Unlike classical computing where we can only define two logic gates operating on a single bit (the identity gate and the NOT gate), quantum computing has infinitely many single qubit logic gates: any unitary 2 × 2 matrix (rotation) is a quantum logic gate. Some of these gates are more important (or easier to implement) than others. Below we provide detailed descriptions of some of them, starting with the identity I and the Pauli matrices X, Y, and Z. The action of the I gate is obvious – it leaves the state of the qubit unchanged; Pauli matrices perform rotation of the qubit state by π radians around, respectively, the x, y, and z axis:

$$\mathtt{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathtt{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \mathtt{Y} = \begin{bmatrix} 0 & -\mathrm{i} \\ \mathrm{i} & 0 \end{bmatrix}, \quad \mathtt{Z} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

We can easily verify by performing simple algebraic operations that the X gate flips the bit and that the Z gate flips the phase:

$$\mathtt{X}\text{ gate:} \quad \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

$$\mathtt{Z}\text{ gate:} \quad \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = -\begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

These operations can be visualised with the help of the following graphical representation of the quantum gates:

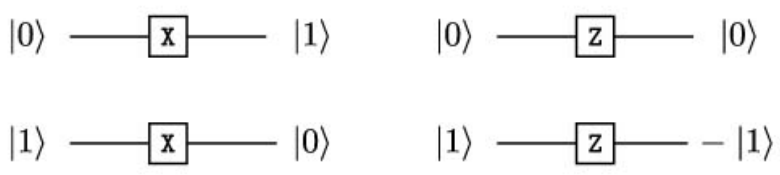


Figure 6.10: Graphical representation of the X and Z gates.

Here, horizontal lines represent *quantum registers* and boxes represent quantum gates. Together, quantum registers and quantum gates form the graphical representation of *quantum circuits* – the sequence of quantum gates that transform the quantum state, thus implementing quantum computation. The quantum circuits are read from left to right: the initial quantum state is shown at the left end of the quantum circuit and the final state is shown at the right end. Often, the last operator on the quantum register is the *measurement operator*. Following measurement (in the computational basis), a qubit is transformed into a classical bit and its value is a known binary number.

Since the X gate flips the state of the qubit, it is also called the NOT gate. Similarly, the Z gate that flips the phase of the qubit state is known as the PHASE gate.

We can draw a direct analogy between the NOT gate in classical computing and the NOT gate in quantum computing but there are quantum gates that perform operations that do not exist in classical computing. One such example is the $\sqrt{\texttt{NOT}}$ gate (represented by the matrix M introduced in Section 1.2.1). In classical computing, we do not have a function that, when applied twice, would flip the bit. But such a function exists in quantum computing:

$$\sqrt{\texttt{NOT}} \equiv \sqrt{\texttt{X}} = \frac{1}{2}\begin{bmatrix} 1+\mathrm{i} & 1-\mathrm{i} \\ 1-\mathrm{i} & 1+\mathrm{i} \end{bmatrix}.$$

We know that one of the main sources of power of quantum computing is the ability of a qubit to exist in a superposition of basis states. But how can we put a qubit that was initialised as $|0\rangle$ (or $|1\rangle$) in a superposition of states $|0\rangle$ and $|1\rangle$? The answer is the Hadamard gate, H, which creates an equal superposition of states $|0\rangle$ and $|1\rangle$ when applied to either state $|0\rangle$ or state $|1\rangle$:

$$\texttt{H} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix},$$

$$\texttt{H}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 0 \end{bmatrix} + \frac{1}{\sqrt{2}}\begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \text{and} \quad \texttt{H}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 0 \end{bmatrix} - \frac{1}{\sqrt{2}}\begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$



Figure 6.11: Graphical representation of the Hadamard H gate.

Interestingly, the Hadamard H gate is its own inverse, so that the second application of the Hadamard gate reverses the action of the first (mathematically, $H^2 = I$, or $H = H^{-1}$):



Figure 6.12: Hadamard H gate applied twice.

Some other useful one-qubit gates are the *phase shift* gates where the phase is shifted by $\pi/2$ and $\pi/4$ rather than by π as is the case for the Z (PHASE) gate:

$$\mathtt{S} = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/2} \end{bmatrix} \quad \text{and} \quad \mathtt{T} = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}.$$

Finally, it is necessary to mention the *adjustable* one-qubit gates that perform rotation of the qubit state around a specific axis by an arbitrary angle θ. For any given gate G, define

$$\mathtt{R_G}(\theta) := \exp\left(-\frac{1}{2} i\theta \mathtt{G}\right).$$

Using Lemma 1 in Chapter 1, we can then immediately compute R_X, R_Y and R_Z as

$$\mathtt{R_X}(\theta) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix},$$

$$\mathtt{R_Y}(\theta) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix},$$

$$\mathtt{R_Z}(\theta) = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}.$$

The adjustable gates play a very important role in Parameterised Quantum Circuits (PQC), which we consider in the following chapters of this book.

6.3.4 Two-qubit quantum logic gates

Similar to one-qubit gates specified by unitary 2 × 2 matrices, we can construct any number of multi-qubit gates. The n-qubit gates would be represented by $2^n \times 2^n$ unitary matrices. Since multi-qubit gates act on several qubits at the same time, they can be used to *entangle* them – i.e., make their states depend on each other. We also have a possibility to create conditional operators, where an operator is applied to a target qubit only if a control qubit is in state $|1\rangle$. Such gates are called *controlled* gates and we consider below some of them.

Controlled gates are shown in the quantum circuit as a straight line connecting two quantum registers. One quantum register represents the control qubit, and is indicated by the dot placed at the end of the line connecting the quantum registers. Another quantum register represents the target qubit: the desired conditional operator is put on this register. Figure 6.13 illustrates this by displaying a Controlled Y (CY) gate. Here, q_1 is the quantum register representing the control qubit, q_2 is the quantum register representing the target qubit, and the operator applied to the target qubit is Y.

$$\mathrm{CY} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -\mathrm{i} \\ 0 & 0 & \mathrm{i} & 0 \end{bmatrix}.$$

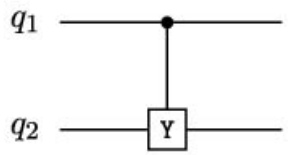


Figure 6.13: CY gate.

The Controlled NOT gate, usually denoted as CNOT or CX, is another example of a two-qubit controlled gate. It consists of applying the Pauli X gate to the target qubit if the control qubit is in state $|1\rangle$ and is given by the following unitary matrix:

$$\texttt{CNOT} \equiv \texttt{CX} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

This gate is often represented in the quantum circuit by an XOR logical symbol (circled plus) placed on the target qubit quantum register since its truth table (for the target qubit) coincides with the truth table of the XOR logic gate.

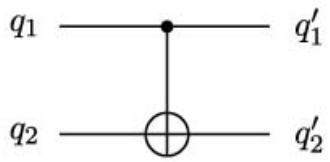


Figure 6.14: CX (CNOT) gate.

q_1	q_2	q'_1	q'_2
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Table 6.3: Truth table for CX (CNOT) gate.

Seen differently, note that we in fact have the equality

$$\texttt{CX}\,|q_1 q_2\rangle = |q_1\rangle\,|q_1 \oplus q_2\rangle\,,$$

for any q_1,$q_2 \in \{0,1\}$, where $\oplus$ denotes addition modulo 2.

The CZ gate is a Pauli Z (phase flip) applied to the target qubit conditional on the control qubit being in state $|1\rangle$ and is given by the following unitary matrix:

$$\texttt{CPHASE} \equiv \texttt{CZ} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}.$$

Interestingly, for CZ it does not really matter which qubit is the target qubit and which is the control qubit – the result is the same:

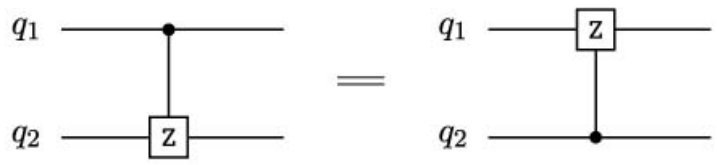


Figure 6.15: CZ (CPHASE) gates.

The SWAP gate swaps the states of two qubits. The $\sqrt{\texttt{SWAP}}$ gate is universal in the sense that any multi-qubit gate can be constructed from only $\sqrt{\texttt{SWAP}}$ and single qubit gates.

$$\texttt{SWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \sqrt{\texttt{SWAP}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1+i}{2} & \frac{1-i}{2} & 0 \\ 0 & \frac{1-i}{2} & \frac{1+i}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Very often, the choice of the set of universal gates from which all other gates can be constructed is dictated by the characteristics of the physical system used to perform quantum computation. The $\sqrt{\texttt{SWAP}}$ is a *native gate* in the systems that exploit exchange interactions [221]. Related gates such as iSWAP and $\sqrt{\texttt{iSWAP}}$ are natural gates in the systems with Ising-like interactions [244]:

$$\texttt{iSWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & i & 0 \\ 0 & i & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \sqrt{\texttt{iSWAP}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{i}{\sqrt{2}} & 0 \\ 0 & \frac{i}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

An example of an adjustable two-qubit gate is the XY gate, which is a rotation by some angle θ between the $|01\rangle$ and $|10\rangle$ states:

$$\texttt{XY}(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\left(\frac{\theta}{2}\right) & i\sin\left(\frac{\theta}{2}\right) & 0 \\ 0 & i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Note that XY(π) = iSWAP and XY($\pi/2$) = $\sqrt{\text{iSWAP}}$. Together with CZ, the iSWAP gate plays an important role in the construction of quantum circuits since any two-qubit gate can be expressed with at most three CZ or three iSWAP gates [2].

6.3.5 The Toffoli gate

The classical Toffoli gate, invented by Tommaso Toffoli [284], is a three-bit logic gate, which is universal in classical computing. In quantum computing, it is a three-qubit Controlled Controlled NOT (CCNOT) gate that is represented by the following quantum circuit, where the qubit C is the target qubit and the qubits A and B are the control qubits:

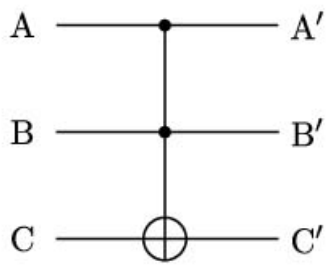


Figure 6.16: Toffoli (CCNOT) gate.

The classical Toffoli gate is given by the following truth table:

A	B	C	A′	B′	C′
0	0	0	0	0	0
1	0	0	1	0	0
0	1	0	0	1	0
1	1	0	1	1	1
0	0	1	0	0	1
1	0	1	1	0	1
0	1	1	0	1	1
1	1	1	1	1	0

Table 6.4: Truth table for Toffoli gate.

The quantum Toffoli gate is represented by the unitary matrix:

$$\texttt{CCNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

It is clear from the Toffoli gate truth table that it also implements the AND and NAND gates. With C = 0 it can be viewed as the AND gate:

$$\text{if } \mathrm{C} = 0: \quad \mathrm{C}' = \mathrm{A}\ \texttt{AND}\ \mathrm{B}.$$

And with C = 1 it can be viewed as the NAND gate:

$$\text{if } \mathrm{C} = 1: \quad \mathrm{C}' = \mathrm{A}\ \texttt{NAND}\ \mathrm{B}.$$

The Toffoli gate can be decomposed into a quantum circuit consisting of CNOT and one-qubit gates:

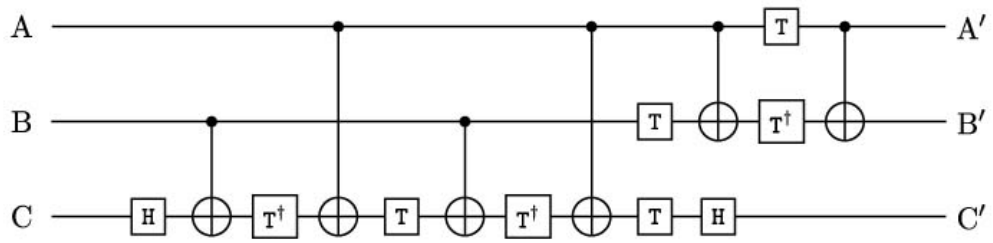


Figure 6.17: Decomposition of the Toffoli (CCNOT) gate. The "dagger" superscript after the gate symbol such as $T^\dagger$ indicates the adjoint operator (see Section 1.1.1).

The fact that the Toffoli circuit allows us to implement the NAND gate, which is universal in classical computing, demonstrates the fact that quantum computing can perform all operations which are possible on classical computers. In other words, quantum computers can simulate classical computers. At the same time, we have seen examples of quantum operations that do not have their analogues in classical computing. In the most general case, classical simulation of an n-qubit quantum system would require the ability to store 2^n probability amplitudes – an impossible task for n larger than several hundred as there would not be enough

matter in the visible universe to implement such classical memory. Consequently, quantum computation is more general than classical computation. Computation, as a concept, is really quantum computation. Classical computation is just a special case of quantum one [32].



Quantum computing offers a wider range of logic gates than classical computing.

The Toffoli gate demonstrates that quantum computers can perform all operations implementable on classical computers. At the same time, an attempt to simulate quantum computing classically will immediately run into memory issues.

As discussed, quantum gates correspond to unitary matrices, which have the property of being invertible. Since quantum circuits are fundamentally classical and tensor products of such matrices, they can easily be inverted, yielding the concept of *reversible computing*, which we focus on in the next section.

6.4 Reversible Computing

The importance of the Toffoli gate goes beyond mere universality. It is a universal *reversible* logic gate, meaning that it can serve as a basis for reversible computing. Here, we should note that *all* quantum logic gates that are represented by unitary matrices are reversible. So, what does reversible computing mean?

Reversible computing is a model of computation where the computational process is time-reversible. It also means that no information is lost through the computation process and we can always reconstruct the initial state. The ability to physically realise reversible computing is hugely important due to the deep physical link between loss of information and generation of heat.

According to the principle formulated by Landauer [183], in order for a computational process to be physically reversible, it must also be logically reversible. Fundamentally, this is due to the fact that the act of computation can only be performed by some physical system and is subject to the physical laws of thermodynamics.

The loss of information leads to the increase in information entropy. Similarly, the increase in thermodynamic entropy leads to the generation of heat. In both cases, we are moving from the more ordered state to the less ordered state, which is an irreversible process.

This can be illustrated by the definitions of entropy (as a measure of disorder) in both statistical mechanics and information theory. Entropy in statistical mechanics is given by

$$S = -k_B \sum_i p_i \log(p_i),$$

where k_B is the Boltzmann constant and p_i is the probability of the microstate i taken from the equilibrium ensemble (macroscopic thermodynamic state), while entropy in information theory is given by

$$\mathrm{H} = -\sum_i p_i \log_2(p_i),$$

where p_i is the probability of the message i taken from the message space.

A very high probability of a particular microstate and a very high probability of a specific message indicate highly ordered systems with low entropy. The entropy is maximised (and information is minimised) when the microstates/messages are uniformly distributed.

Any probability distribution can be approximated arbitrarily closely by some thermodynamic system [219]. If h is information (bits) per particle, then for N particles the entropy measured in the *natural unit of information* (1 bit = log(2) nat) is given by the expression

$$S = -k_B \log(2) N h.$$

In energy units, $k_B T$ log(2) of heat is generated for each bit of information lost. Here, T is the temperature of the heat sink (in Kelvins). For example, if we take T = 300K (approx. 27C), then the minimum possible amount of energy required to erase one bit of information is 2.87 zJ (zeptojoule).

In practical terms, it means that every *logically irreversible* operation (e.g., NAND or XOR gates) must be accompanied by the corresponding entropy increase and generation of heat. As the energy efficiency of the computation process becomes

progressively more important, efforts to develop reversible computing are increasing since it may prove difficult for traditional technology based on the laws of classical physics to progress very far beyond current levels of energy efficiency if reversible computing principles are not used [106].



In contrast with classical computing, all quantum computing operations are reversible (except measurement). This means that the quantum advantage is likely to be demonstrated not only in quantum speedup and the expressive power of quantum circuits but also in achieving superior energy efficiency.

We have so far seen forward and backward (reversible) quantum operations. However, what fundamentally distinguishes quantum computing from classical computing is the concept of entanglement.

6.5 Entanglement

The key aspect of quantum computing is entanglement, which allows for quantum states to encode more information than the sum of their individual components. We explain this in detail here and provide examples for two-qubit systems.

6.5.1 Quantum entanglement and why it matters

An n-qubit system can exist in any superposition of the 2^n basis states:

$$\sum_{i=0}^{2^n-1} c_i \left|i\right\rangle = c_0 \left|00\ldots00\right\rangle + c_1 \left|00\ldots01\right\rangle + \ldots + c_{2^n-1} \left|11\ldots11\right\rangle,$$

with

$$\sum_{i=0}^{2^n-1} |c_i|^2 = 1.$$

If such a state can be represented as a tensor product of individual qubit states then the qubit states are *not entangled*. For example, it is easy to check that

	$\frac{1}{4\sqrt{2}}$

$$\left(\sqrt{3}\,|000\rangle + |001\rangle + 3\,|010\rangle + \sqrt{3}\,|011\rangle + \sqrt{3}\,|100\rangle + |101\rangle + 3\,|110\rangle + \sqrt{3}\,|111\rangle\right)$$

$$= \left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) \otimes \left(\frac{1}{2}|0\rangle + \frac{\sqrt{3}}{2}|1\rangle\right) \otimes \left(\frac{\sqrt{3}}{2}|0\rangle + \frac{1}{2}|1\rangle\right), \quad (6.5.1)$$

so that the quantum state is not entangled (only in superposition). An *entangled* state cannot be represented as a tensor product of individual qubit states.

For example, the two-qubit state

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

does not allow a tensor product decomposition. Namely, for any $a,b,c,d \in \mathbb{C}$ such that $|a|^2 + |b|^2 = |c|^2 + |d|^2 = 1$, we have

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle \neq (a\,|0\rangle + b\,|1\rangle) \otimes (c\,|0\rangle + d\,|1\rangle).$$

We notice that we need 2^n probability amplitudes to describe the state on the left side of (6.5.1) while we only need $2n$ probability amplitudes to describe the state on the right side of (6.5.1). The number of probability amplitudes needed to fully describe the state of a system is directly related to the amount of information the system can store. Entanglement allows us to encode a significantly larger amount of information than is possible with individual independent qubits. One can say that most of the information encoded in the state of a quantum mechanical system is stored non-locally in the correlations between the qubit states. This non-locality of information is one of the major distinguishing features of quantum computing over classical computing and is essential for a number of applications.

What happens if we measure the entangled qubits? In (6.5.1) both qubits are in the state of equal superposition, i.e., if we measure the first qubit we will get both 0 and 1 with probability 1/2. If instead we measure the second qubit we will also get 0 and 1 with equal probability. However, the situation is completely different if we measure the second qubit after the first has already been measured. In this case, the state of the second qubit is fully determined by the act of measuring the first qubit and there is no longer any uncertainty about its value: if the first qubit was measured as 0, the second qubit is also in state 0 and if the first qubit was measured as 1, the second qubit is also in state 1. In other words, measuring one qubit collapses the superposition and has an immediate effect on the other.

6.5.2 Entangling qubit states with two-qubit gates

Qubit states can be entangled with the help of two-qubit gates. The two-qubit state given by (6.5.1) is known as one of the four maximally entangled Bell states. It can be constructed from the unentangled state $|00\rangle$:

$$|00\rangle = (1 \cdot |0\rangle + 0 \cdot |1\rangle) \otimes (1 \cdot |0\rangle + 0 \cdot |1\rangle)$$

by applying the *Bell circuit* consisting of H and CNOT gates:

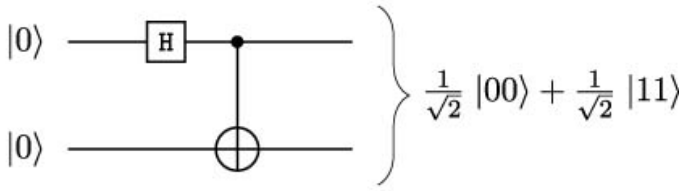


Figure 6.18: Bell circuit.

Running this circuit on the unentagled states $|01\rangle$, $|10\rangle$, and $|11\rangle$ will result in the construction of the other three Bell states:

$$|01\rangle \rightarrow \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle),$$

$$|10\rangle \rightarrow \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle),$$

$$|11\rangle \rightarrow \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle).$$

Entanglement can be achieved with other two-qubit gates as well. Depending on the hardware implementation, it may be a SWAP, a CPHASE, or some other fixed two-qubit gate, or it can be an adjustable two-qubit gate such as XY(θ).



Entanglement allows us to store most of the information in the correlations between the states rather than in the states of individual qubits.

Entanglement is one of the main sources of the expressive power of quantum circuits that underpins our search for the quantum advantage.

We saw that entanglement is a distinctive feature of quantum computing. We now see how it comes into play when analysing the quantum equivalents of classical logic gate decompositions studied in Section 6.1.

6.6 Quantum Gate Decompositions

The most widely used NISQ computing technologies are trapped ions and superconducting qubits. In both cases, one-qubit gates are much faster than two-qubit gates (by an order of magnitude). Additionally, one-qubit gates have much higher fidelity [46, 164]. This means that we can treat one-qubit gates as computationally inexpensive and should not worry too much about their quantities. At the same time, we have to be economical with two-qubit gates: out of two equivalent circuits, the one with the smaller number of two-qubit gates would generally perform better. Therefore, we should be aware of the two-qubit gates that are native to any particular system – gates that can be implemented naturally using standard hardware control techniques. More complex gates can be decomposed into a sub-circuit of the native gates but an even better solution would be to specify the algorithm that takes advantage of the native gates and bypasses the need of having non-native two-qubit gates. For example, Rigetti's Aspen system [275] is based on the superconducting qubits with two native two-qubit gates CZ and XY – constructing a circuit based on these gates rather than, e.g., SWAP gates would achieve better performance.

However, it is not always practical or desirable to make an algorithm hardware dependent. And since the choice of the native gates is inevitably limited, it is useful to keep in mind several basic decompositions. The following relationships can be verified by direct calculations and play an important role in quantum circuit construction:

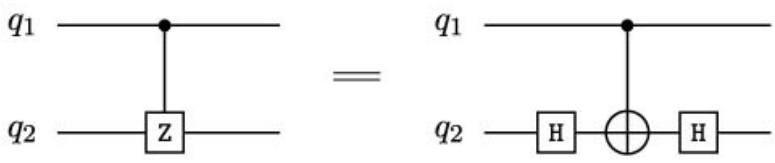


Figure 6.19: CZ gate decomposition into CX and Hadamard gates.

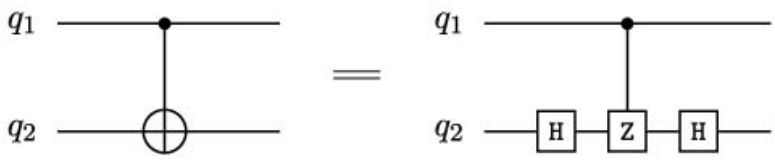


Figure 6.20: CX gate decomposition into CZ and Hadamard gates.

Given the limited connectivity of NISQ devices (nearest neighbours for most qubits), the SWAP gate that swaps the states of the qubits is especially useful and its efficient implementation using available native gates is very important. The SWAP gate can be represented by a sub-circuit consisting of three CX gates:

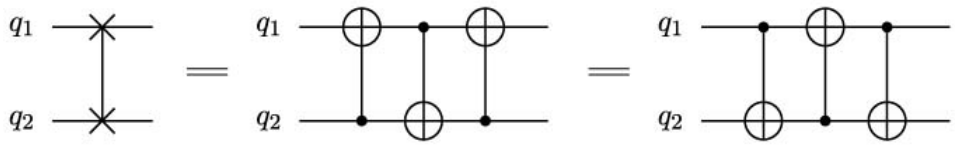


Figure 6.21: SWAP gate decomposition into three CX gates.

Taking into account the relationship between CX and CZ gates in Figure 6.20, the SWAP gate can also be decomposed into a sub-circuit of three CZ and a handful of one-qubit Hadamard gates:

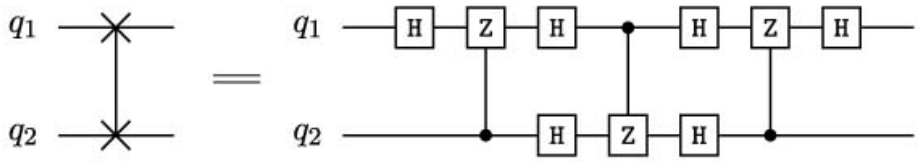


Figure 6.22: SWAP gate decomposition into three CZ and six Hadamard gates.

Alternatively, the SWAP gate can be implemented with the help of three iSWAP gates:

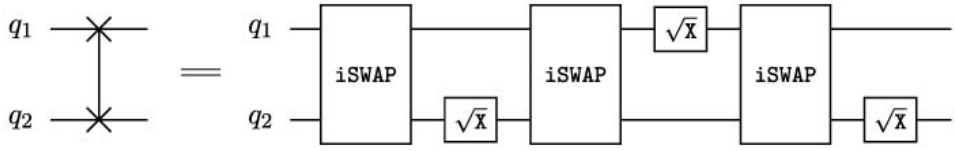


Figure 6.23: SWAP gate decomposition into three iSWAP gates (iSWAP = XY(π)) and three $\sqrt{X}$ gates.

It is easy to verify by direct calculations that the circuit on the right side of Figure 6.23 performs the following transformations:

$$|0\rangle \otimes |1\rangle \equiv \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \longrightarrow \begin{bmatrix} 0 \\ 0 \\ i \\ 0 \end{bmatrix} = \exp\left(i\frac{\pi}{2}\right) \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \equiv \exp\left(i\frac{\pi}{2}\right) |1\rangle \otimes |0\rangle ,$$

$$|1\rangle \otimes |0\rangle \equiv \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \longrightarrow \begin{bmatrix} 0 \\ \mathrm{i} \\ 0 \\ 0 \end{bmatrix} = \exp\left(\mathrm{i}\frac{\pi}{2}\right) \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \equiv \exp\left(\mathrm{i}\frac{\pi}{2}\right) |0\rangle \otimes |1\rangle \,.$$

The coefficient $\exp(\mathrm{i}\pi/2)$ is a *global phase* and can be ignored. We can do this because a global phase is not *observable*: measuring the states $|\psi\rangle$ and $\exp(\mathrm{i}\varphi)|\psi\rangle$ will yield the same result (i.e., the same states with the same probabilities) for any $\varphi \in \mathbb{R}$. Said differently, two states differing only by a global phase represent the same physical system.

Finally, we mention the iSWAP representation of the CNOT gate. To do this, we need two iSWAP and several one-qubit gates as shown in Figure 6.24.

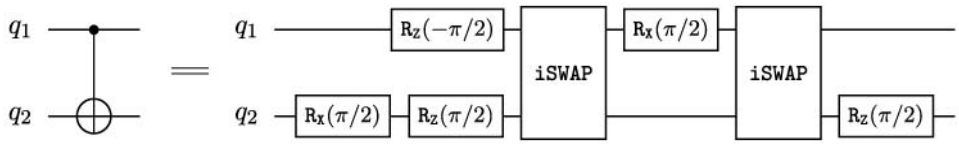


Figure 6.24: CNOT gate decomposition into two iSWAP gates (iSWAP = XY(π)) and several one-qubit rotation gates.

The CNOT gate applies the NOT gate to the target qubit if the control qubit is in state $|1\rangle$ while leaving the control qubit state unchanged. This is exactly what we see when we apply the circuit shown on the right side of Figure 6.24 to the states $|10\rangle$ and $|11\rangle$:

$$|1\rangle \otimes |0\rangle \equiv \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \longrightarrow \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1-\mathrm{i}}{\sqrt{2}} \end{bmatrix} = \exp\left(-\mathrm{i}\frac{\pi}{4}\right) \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \equiv \exp\left(-\mathrm{i}\frac{\pi}{4}\right) |1\rangle \otimes |1\rangle \,,$$

$$|1\rangle \otimes |1\rangle \equiv \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 0 \\ 0 \\ \frac{1-\mathrm{i}}{\sqrt{2}} \\ 0 \end{bmatrix} = \exp\left(-\mathrm{i}\frac{\pi}{4}\right) \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \equiv \exp\left(-\mathrm{i}\frac{\pi}{4}\right) |1\rangle \otimes |0\rangle \,.$$

Here, the accumulated unobservable global phase is $\exp(-\mathrm{i}\pi/4)$.



Decomposition of non-native two-qubit gates into the subcircuits consisting of the native two-qubit gates and high fidelity one-qubit gates allows us to build hardware-independent quantum algorithms.

Mimicking the setup for classical logic gates above, we now investigate how qubits and quantum logic gates can be effectively (physically) realised.

6.7 Physical Realisations of Qubits and Quantum Gates

Now that the theoretical framework for quantum bits and quantum gates has been set, it is important to understand how these can actually be realised from a hardware point of view.

6.7.1 The DiVincenzo criteria

The modern approach to building quantum computing hardware was marked by the set of requirements for the physical implementation of quantum computation proposed in 2000 by DiVincenzo [87]. These requirements, now known as the "DiVincenzo criteria", are as follows:

1. A scalable physical system with well characterised qubits. A qubit being "well characterised" means the following:

- its physical parameters should be accurately known, including the internal Hamiltonian of the qubit, which determines the qubit energy eigenstates. Typically, the ground state is taken as $|0\rangle$ and the first excited state is taken as $|1\rangle$;
- the presence of and couplings to other states of the qubit;
- the couplings to external fields, needed to manipulate the state of the qubit;
- the interactions with other qubits, needed to implement multi-qubit gates.

2. The ability to initialise the state of the system to a simple fiducial state, such as an all-zero state. This requirement arises from the clear need to initialise quantum registers to a known value before the start of the computation. Another motivation for this requirement is the fact that quantum error correction requires a continuous, fresh supply of qubits in a low-entropy state ($|0\rangle$ state). The need for a

continuous supply of 0s, rather than just an initial supply, is a non-trivial problem that may rule out some of the potentially promising qubit implementations.

3. Long relevant decoherence times, much longer than the gate operation time. Decoherence can be viewed as the loss of information from a quantum system into the environment. Coupling with the environment leads to entanglement between the system and environment and the transfer of quantum information to the surroundings. As a result, the system dynamics is no longer unitary and the computation becomes irreversible (though the combined system plus environment evolves in a unitary fashion). This means that the quantum computer behaves as a classical machine. Therefore, it is important to preserve coherence for sufficiently long time to ensure that the uniquely quantum features of this style of computation have a chance to come into play. The term "relevant" emphasises that a physical system that realises a qubit can have many decoherence times pertaining to different degrees of freedom but many of these can be irrelevant to the functioning of this system as a qubit.

4. A universal set of quantum gates. In all the physical implementations, only particular sorts of Hamiltonians can be turned on and off. In most cases, we are limited by only two-body (two-qubit) interactions. This poses a problem for a quantum computation specified with multi-qubit unitary transformations. Fortunately, these can always be re-expressed in terms of sequences of one- and two-qubit gates, and the two-qubit gates can be of just one type, which is "native" to a particular implementation (e.g., CNOT, CPHASE, or XY).

5. A qubit-specific measurement capability. This is a straightforward requirement for the efficient quantum computing process: the result of a computation must be read out, and this requires the ability to measure specific qubits.

There are many possible realisations of quantum computers satisfying the DiVincenzo criteria. The fundamental building blocks of quantum computers – qubits – can be constructed from electrons, photons, trapped ions, neutral atoms, superconducting circuits, to name just a few possibilities. Essentially, any quantum mechanical system that can exist in a superposition of two distinct states with controlled transitions between them can serve as a physical realisation of a qubit. This can be the spin of an electron ("up", "down") or the polarisation of a photon ("vertical", "horizontal"). In this section, we start with considering how the DiVincenzo criteria can be satisfied by superconducting qubits.

6.7.2 Superconducting qubits

Qubits constructed from tiny superconducting circuits are strong candidates for the scalable physical realisation of the principles of digital quantum computing. In a normal conductor, the charge carriers are individual electrons. Electrons are spin-$\frac{1}{2}$ elementary particles (fermions) satisfying the Pauli principle: no two fermions can simultaneously occupy the same state. In a superconducting circuit, the basic charge carriers are pairs of electrons (known as Cooper pairs), which are bosons (the total spin of a Cooper pair is an integer number) and can occupy the same quantum energy level. This effect is known as the Bose-Einstein condensate. The condensate wave function allows designing and measuring macroscopic quantum effects. The parameters of the superconducting circuits may be designed by setting the classical values of the electrical elements that compose them, e.g., adjusting the capacitance and inductance.

This gives us a concrete idea of how to build macroscopic qubits with desired quantum properties. Let us first have a look at the system known as Quantum Harmonic Oscillator (QHO) shown schematically in Figure 6.25. Before explaining the physical aspects, let us have a look at its mathematical justification.

From classical to quantum harmonic oscillator

Consider a simple harmonic oscillator, namely a spring on a flat frictionless surface, attached on one side to an unmovable object and on the other side to a movable one (say, a weight). In the equilibrium state, in the resting position, nothing moves. After applying some force, say by pulling (or pushing) the movable object, the spring starts oscillating due to its restoring force F_{R}. Hooke's law states that this force is proportional to the extension, namely

$$F_{\mathrm{R}}(x(t)) = -kx(t),$$

starting from $x(0) = x_0 \in \mathbb{R}$, where $x(t)$ denotes the position of the spring at time t and k is the spring constant. Newton's second law of motion also states that

$$F_{\mathrm{R}}(x(t)) = ma(t),$$

for $t \geq 0$, where $a(t)$ denotes the acceleration at time t, and m is the mass of the spring. Since $a(t) = \ddot{x}(t)$, combining the two equations yields, for each $t \geq 0$,

$$\ddot{x}(t) = -\frac{k}{m}x(t),$$

starting from $x(0) = x_0$, which is the equation of motion for the simple oscillator. It is a simple one-dimensional second order linear ordinary differential equation, which can be solved simply as

$$x(t) = x_0 \cos(\omega t) + \frac{v_0}{\omega} \sin(\omega t), \quad \text{for all } t \geq 0,$$

where

$$\omega := \sqrt{\frac{k}{m}} \quad \text{and} \quad v_0 := \dot{x}(0)$$

are, respectively, the natural frequency of the oscillator and the speed. Trigonometric manipulations show that (6.7.2) can equivalently be written as

$$x(t) = \alpha \cos(\omega t - \varphi), \quad \text{for all } t \geq 0,$$

where

$$\alpha := \sqrt{x_0^2 + \frac{v_0^2}{\omega^2}} \quad \text{and} \quad \tan(\varphi) := \frac{v_0}{\omega x_0}.$$

Recall now that the potential energy $\mathfrak{V}$ is the energy stored in the oscillator when it is extended or compressed, i.e. (considering $x = 0$ as the equilibrium state),

$$\mathfrak{V}(x,t) = -\int_0^{x(t)} F_{\mathrm{R}}(z)\mathrm{d}z = \frac{k}{2}x(t)^2.$$

The total energy of the system is then the sum of the kinetic and the potential energies:

$$\begin{aligned} \mathfrak{E}_{\text{total}}(t) &= \mathfrak{E}_{\text{kinetic}}(t) + \mathfrak{V}(x(t)) \\ &= \frac{m}{2} v(t)^2 + \frac{k}{2} x(t)^2. \end{aligned}$$

Using the explicit solution (6.7.2), with $\omega := \sqrt{\frac{k}{m}}$ we obtain:

$$\begin{aligned} \mathfrak{E}_{\text{total}}(t) &= \frac{m}{2}\left(\frac{\mathrm{d}}{\mathrm{d}t}\left(\alpha \cos(\omega t - \varphi)\right)\right)_2 + \frac{k}{2}\left(\alpha \cos(\omega t - \phi)\right)^2 \\ &= \frac{m\alpha^2\omega^2}{2} \sin(\omega t - \phi)^2 + \frac{m\alpha^2\omega^2}{2} \cos(\omega t - \phi)^2 \end{aligned}$$

$$= \frac{m\alpha^2\omega^2}{2} = \frac{k\alpha^2}{2}.$$

In this classical setting, we see that the total energy of the system can take a continuum of values. The quantum counterpart is fundamentally different and we shall see below that it is in fact quantised (giving rise to the "quantum" theory), as originally proposed by Bohr in 1913, and later detailed by Schrödinger and Heisenberg in 1926. Recall now the general form of the time-dependent Schrödinger equation describing the evolution of a system over time:

$$\mathrm{i}\hbar\frac{\mathrm{d}\Psi(x,t)}{\mathrm{d}t} = \mathcal{H}\Psi(x,t),$$

where $\mathcal{H}$ represents the Hamiltonian of the system. Since the latter (representing the energy of the system) is the sum of the kinetic energy and the potential energy, we have

$$\mathcal{H} = \mathfrak{E}_{\text{kinetic}} + \mathfrak{E}_{\text{potential}} = \frac{p^2}{2m} + \mathfrak{V} = -\frac{\hbar}{2m}\frac{\mathrm{d}^2}{\mathrm{d}x^2} + \mathfrak{V},$$

where m is the mass of the particle, $\hbar$ is the usual Planck constant, $\mathfrak{V}$ is the potential representing the environment, and p is the momentum operator:

$$p = -\mathrm{i}\hbar\frac{\mathrm{d}}{\mathrm{d}x}.$$

Plugging this Hamiltonian into (6.7.2) yields

$$\mathrm{i}\hbar\frac{\mathrm{d}\Psi(x,t)}{\mathrm{d}t} = \left(-\frac{\hbar^2}{2m}\frac{\mathrm{d}^2}{\mathrm{d}x^2} + \mathfrak{V}(x)\right)\Psi(x,t).$$

Since the potential $\mathfrak{V}$ does not depend on time, separation of variables, with $\Psi(x,t) = \psi(x)u(t)$, gives

$$\mathrm{i}\hbar\psi(x)u'(t) = \left(-\frac{\hbar^2}{2m}\psi''(x) + \mathfrak{V}(x)\psi(x)\right)u(t),$$

or else

$$\mathrm{i}\hbar\frac{u'(t)}{u(t)} = \frac{-\frac{\hbar^2}{2m}\psi''(x) + \mathfrak{V}(x)\psi(x)}{\psi(x)}.$$

Since both sides depend on a different variable, they must be equal to a constant, say E, and we thus obtain the ordinary differential equation

$$\mathrm{i}\hbar\frac{u'(t)}{u(t)} = E,$$

as well as the eigenvalue equation

$$\mathcal{H}\psi(x) = -\frac{\hbar^2}{2m}\psi''(x) + \mathfrak{V}(x)\psi(x) = E\psi(x).$$

The first one immediately admits the solution, with normalisation $u(0) = 1$,

$$u(t) = \exp\left(-\frac{\mathrm{i}Et}{\hbar}\right).$$

The eigenvalue equation (6.7.2) can be solved, for example, by spectral method. In fact, it can be proved (and we refer the interested reader to [258, Section 3.1] for details) that the operator $\mathcal{H}$ here admits a finite spectrum, with the set of (eigenvalues, eigenfunctions) $\{(E_n, \psi_n)\}_{n\geq 0}$ given by

$$\begin{cases} E_n &= \left(n + \frac{1}{2}\right)\hbar\omega, \\ \psi_n(x) &= \frac{1}{\sqrt{2^n n!}}\left(\frac{m\omega}{\pi\hbar}\right)^{1/4}\exp\left(-\frac{m\omega x^2}{2\hbar}\right)H_n\left(\sqrt{\frac{m\omega}{\hbar}}x\right), \end{cases}$$

for each $n \geq 0, x \in \mathbb{R}$, where H_n denotes the n-th physicists' Hermite polynomial

$$H_n(z) := (-1)^n \mathrm{e}^{z^2}\frac{\mathrm{d}^n}{\mathrm{d}z^n}\left(\mathrm{e}^{-z^2}\right).$$

Physical representation of the QHO

A schematic representation of the QHO is shown in Figure 6.25. In this system, energy oscillates between electrical energy in the capacitor C and magnetic energy in the inductor L.

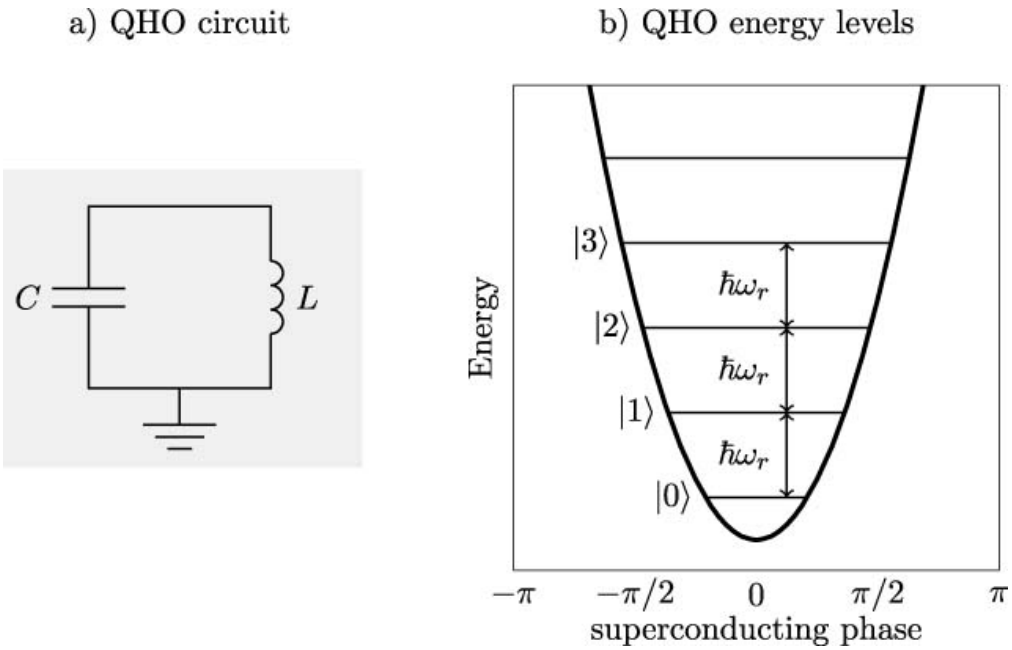


Figure 6.25: Quantum Harmonic Oscillator (QHO).

The Hamiltonian of this system is identical to the one describing a particle in a one-dimensional quadratic potential [177]. The solution to the eigenvalue problem above (see (6.7.2)) gives an infinite series of eigenstates $(|n\rangle)_{n\geq 0}$, whose corresponding eigenenergies, $(E_n)_{n\geq 0}$, are all equidistantly spaced as can be seem from (6.7.2):

$$E_{n+1} - E_n = \hbar\omega_r, \quad \text{for all } n \geq 0,$$

where the resonant frequency, ω_r, is given by the Thompson formula [283]

$$\omega_r = \frac{1}{\sqrt{LC}}.$$

Our first task is to define a computational subspace consisting of only two energy states, $|0\rangle$ and $|1\rangle$, usually the lowest two energy eigenstates in between which transitions can be driven without also exciting other levels in the system. However, we cannot use the lowest two energy eigenstates of the QHO for this purpose since the quantum logic gate operations depend on frequency selectivity. The equidistant level-spacing of the QHO makes this impossible.

Therefore, we need to add anharmonicity (or non-linearity) into the system. We require the transition frequencies ω_{01} between eigenstates $|0\rangle$ and $|1\rangle$ and ω_{12} between eigenstates $|1\rangle$ and $|2\rangle$ to be sufficiently different in order to be

individually addressable. The required non-linearity can be introduced by replacing the inductor *L* with the Josephson junction module *J*, as shown schematically in Figure 6.7.2.

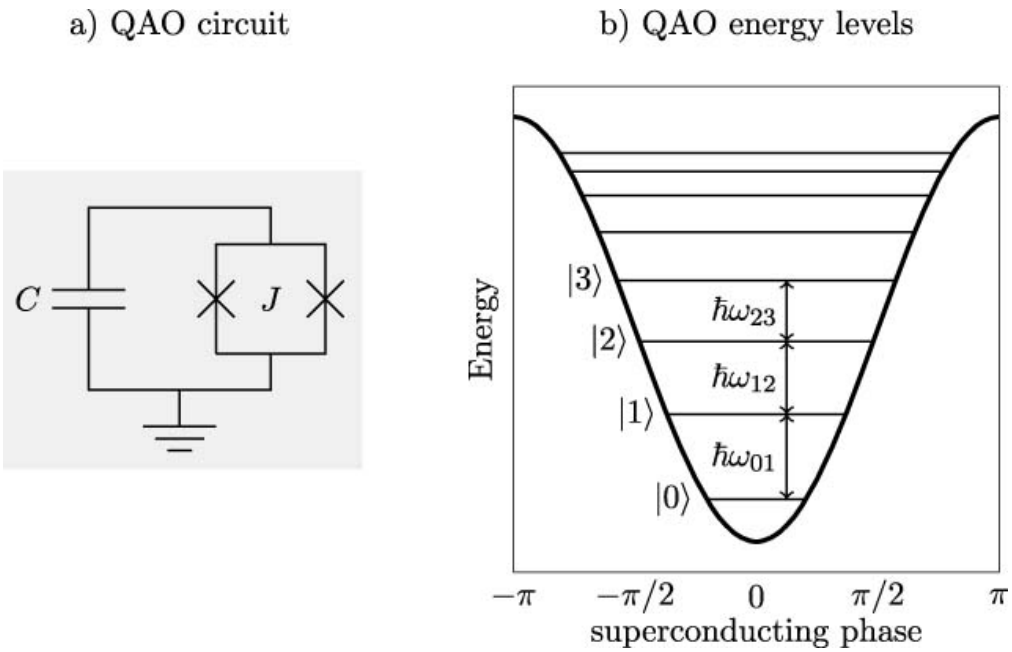


Figure 6.26: Qubit implemented as Quantum Anharmonic Oscillator (QAO). The two lowest energy eigenstates $|0\rangle$ *and* $|1\rangle$ *form the qubit's computational space.*

The Josephson junction is the key element that transforms a superconducting circuit into a qubit. The description of the Josephson effect (the quantum tunnelling of the Cooper pairs) is outside the scope of this book but interested readers are encouraged to learn more about it from the excellent Feynman's Lectures on Physics [101].

After introducing the Josephson module to the circuit (the electric circuit symbol for the Josephson junction is an "X"), the potential energy no longer has the parabolic form (as a function of the superconducting phase), but rather takes a cosinusoidal form, which makes the energy spectrum non-equidistant. Now we can identify the two lowest energy eigenstates as a qubit computational subspace [177, 204].

A superconducting loop with two Josephson junctions in either arm is very sensitive to the magnetic flux enclosed [119]. In the following, we shall use a more compact symbol for the Josephson junction subcircuit:

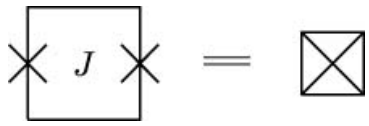


Figure 6.27: Josephson junction module subcircuit.

Remark: It may also be possible to form a computational subspace using the three lowest energy eigenstates: $|0\rangle$, $|1\rangle$ and $|2\rangle$. In this case, we would have a physical realisation of a *qutrit*, whose superposition state vector, $|\psi\rangle$, can be represented as a linear combination of the three orthonormal basis states:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle + \gamma\,|2\rangle\,,$$

where $\alpha,\beta,\gamma \in \mathbb{C}$ are probability amplitudes such that $|\alpha|^2 + |\beta|^2 + |\gamma|^2 = 1$. Qutrits increase the amount of information encoded in a single element, enable techniques that decrease readout errors [200] and reduce the cost of decomposing three-qubit gates into basic two-qubit components [132].

Controlling and measuring superconducting qubits

Having satisfied the first DiVincenzo requirement (well defined qubit), we have to demonstrate how superconducting qubits can be controlled, coupled together to build scalable systems, and measured. We start with the control and measurement of the superconducting qubit states.

The capacitive coupling between a resonator (or a feedline) and the superconducting qubit allows for microwave control to implement single-qubit rotations as well as certain two-qubit gates [177]. Figure 6.28 provides a schematic representation of the superconducting qubit coupled to a microwave source (also referred to as a *qubit drive*). The qubit is controlled by the pulses of microwave radiation. The control parameters are the frequency, the phase and the duration of the pulses.

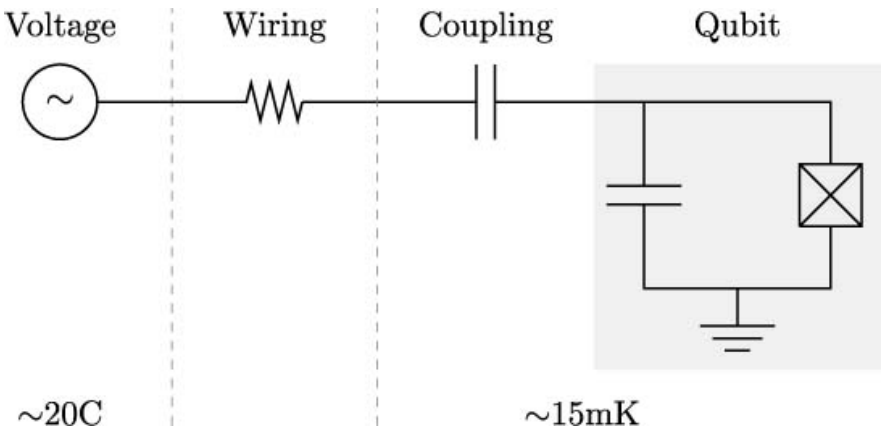


Figure 6.28: Qubit capacitively coupled to the feedline.

Figure 6.29 shows the qubit that is capacitively coupled to a microwave resonator (non-linear) whose frequency is shifted by the qubit state [254]. This frequency shift is exploited for reading the qubit state using the dispersive readout method. When sending a microwave pulse to the resonator, the phase of the reflected (or transmitted) signal conveys information on the qubit state.

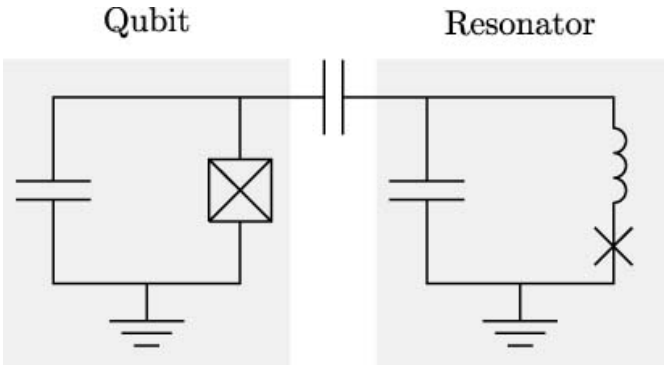


Figure 6.29: Qubit readout circuit that features a non-linear resonator with Josephson junction.

Entanglement with superconducting qubits

For the implementation of multi-qubit gates (and, therefore, entanglement), qubits must be connected. The connectivity of superconducting qubits is realised via capacitive coupling – either directly or with the help of a coupler as shown schematically in Figure 6.30, where capacitive coupling is achieved via a coupler in the form of a linear resonator [177].

The fixed-frequency superconducting qubits typically feature longer coherence times and are less sensitive to flux noise. The two-qubit gate developed for these qubits is the cross-resonance gate CR. In the schematic circuit diagram of two fixed-frequency superconducting qubits coupled through a linear resonator

(Figure 6.30), the CR gate is realised when qubit 1 is driven at the frequency of qubit 2.

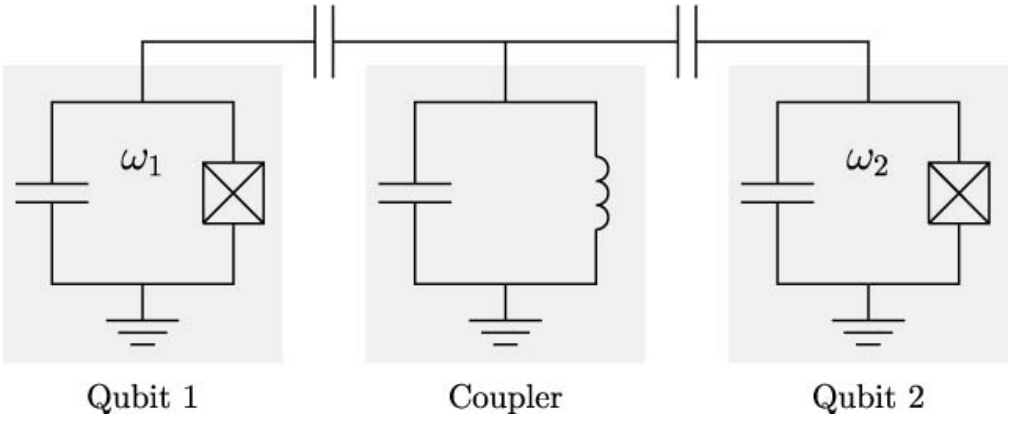


Figure 6.30: Capacitive coupling via coupler (linear resonator). Qubit 1 and Qubit 2 are fixed-frequency qubits with frequencies ω_1 and ω_2 respectively.

The unitary matrix representation of the CR(θ) gate is given by the following [240]:

$$\begin{aligned}\mathrm{CR}(\theta) = \mathrm{R}_{\mathrm{Z}\otimes\mathrm{X}}(\theta) &= \exp\left(-\frac{1}{2}\mathrm{i}\theta\sigma_z\otimes\sigma_x\right) \\ &= \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -\mathrm{i}\sin\left(\frac{\theta}{2}\right) & 0 & 0 \\ -\mathrm{i}\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) & 0 & 0 \\ 0 & 0 & \cos\left(\frac{\theta}{2}\right) & \mathrm{i}\sin\left(\frac{\theta}{2}\right) \\ 0 & 0 & \mathrm{i}\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix},\end{aligned}$$

where the effective rotation angle θ is a function of the physical characteristics of the qubits, the coupler, and the driving microwave pulse.

Due to the form of (6.7.2), the CR(θ) gate can also be denoted as the ZX(θ) gate. This also tells us how we can use the cross-resonance gate to generate a CNOT gate (up to a global phase $\exp(-\mathrm{i}\pi/4)$) in combination with only one-qubit gates:

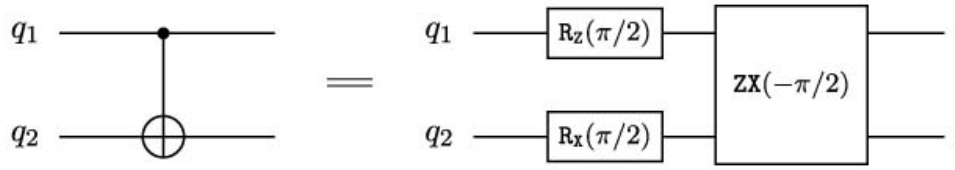


Figure 6.31: CNOT gate decomposition into ZX, R_X and R_Z gates.

6.7.3 Photonic qubits

At the time of writing, it is not clear which qubit construction technology will become the industry standard, if any. There are a lot of exciting experiments and technological breakthroughs ahead of us. The superconducting qubits clearly satisfy the DiVincenzo criteria but there are many other interesting solutions at various stages of development.

One of the possibilities is to encode qubits in photons. Single photons are largely free of noise and can be easily manipulated to realise one-qubit gates. A qubit can be encoded in any of multiple photon's degrees of freedom: temporal, path, and polarisation. One-qubit gates can be implemented using *birefringent waveplates* and conversion between polarisation and path encoding can be achieved using a *polarising beam splitter* [225], where $|0\rangle$ or $|1\rangle$ represents a photon in the upper or lower path, respectively (see Figure 6.33).

As we know, in order to entangle qubits, we need to find a suitable physical implementation of two-qubit gates. Let us have a look at the possible realisation of the CNOT gate using photonic qubits. First of all, we notice that the CNOT gate can be expressed in terms of the CPHASE gate, which can be naturally implemented on the photonics hardware:

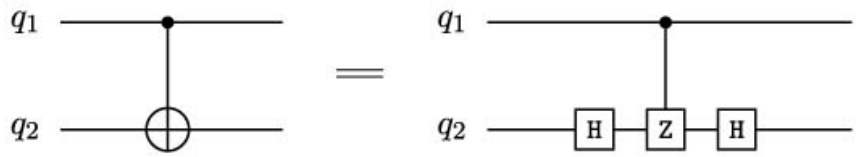


Figure 6.32: CNOT gate decomposition into CPHASE and H gates.

When the control qubit is in state $|0\rangle$ the two H gates cancel each other, and when it is in state $|1\rangle$ the combination of the gates acts as a NOT gate.

Figure 6.33 displays schematic representation of the possible photonic implementation of the CNOT gate [225]. The two paths used to encode the target qubit are mixed at a 50% reflecting beam splitter (BS) that performs the Hadamard operation. If the phase shift is not applied, the second beam splitter (second Hadamard gate) undoes the first by returning the target qubit to the same state it started in. This is an example of classical interference. If a π phase shift is applied, the target qubit is flipped.

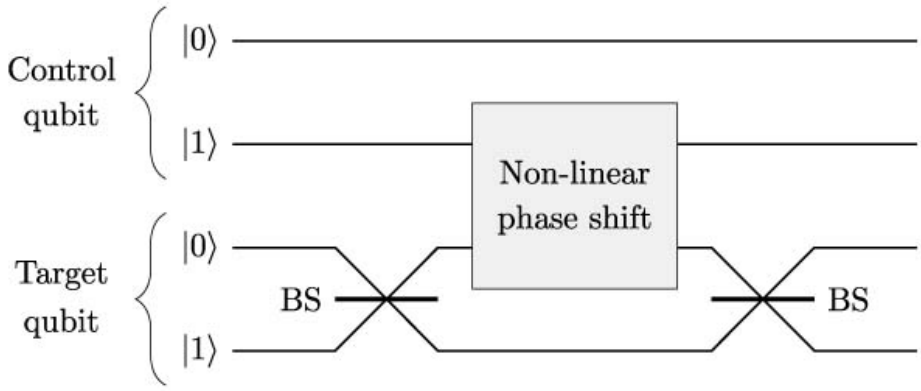


Figure 6.33: Photonic implementation of the CNOT gate.

When the control qubit is in state $|0\rangle$, the π-phase shift is not applied, while it is applied when the control qubit is in state $|1\rangle$. A CNOT gate must implement this phase shift when the control qubit is in the $|1\rangle$ path, otherwise not.

Although the proposed realisation of the CNOT gate is simple in principle, it is a hard practical problem to find a material with optical non-linearity strong enough to implement the conditional phase shift [225]. However, it is possible to achieve a CNOT gate with the help of single photon sources, single photon detectors, and linear optical circuits consisting of beam splitters as was proposed by Knill, Laflamme, and Milburn [165]. The first integrated photonic CNOT gate for polarisation-encoded qubits was demonstrated in [74].

6.7.4 Trapped ion qubits

Another promising approach to building large-scale quantum computers is based on the trapped ion technology [68]. Ions (positively charged atoms that lost an electron) are *trapped* in the electromagnetic field potential, which fixes their positions in space. The quantum chip is cooled and placed in the vacuum chamber. The ions themselves are cooled and made almost motionless by the laser beams that drain their energy through the rapid absorption-emission of photons (ions emit photons of slightly higher frequency than absorbed photons, thus losing their kinetic energy).

The most widely used technique is a *linear trap* shown in Figure 6.34. The two states of the i-th qubit can be identified with the internal states of the corresponding ion: a ground state $|g\rangle_i$ and an excited state $|e\rangle_i$. The trapped ions do not sit perfectly still but can oscillate around their equilibrium positions.

Figure 6.34 depicts a situation where N ions are confined in a linear trap and interact with different laser beams in standing wave configurations [67].

The confinement of the motion along the x-, y-, and z-axes can be described by a harmonic potential of frequencies $\omega_x \ll \omega_y, \omega_z$ respectively. Additionally, the Coulomb repulsion between the positively charged ions provides the coupling of the motion of the ions along the x-axis. The collective motion (excitation) along the x axis, if present, behaves as a quasiparticle called a *phonon*. We denote the state of the Centre-of-Mass (CM) mode of N ions moving in the x-direction as $|0\rangle$ (no phonon) or $|1\rangle$ (one phonon).

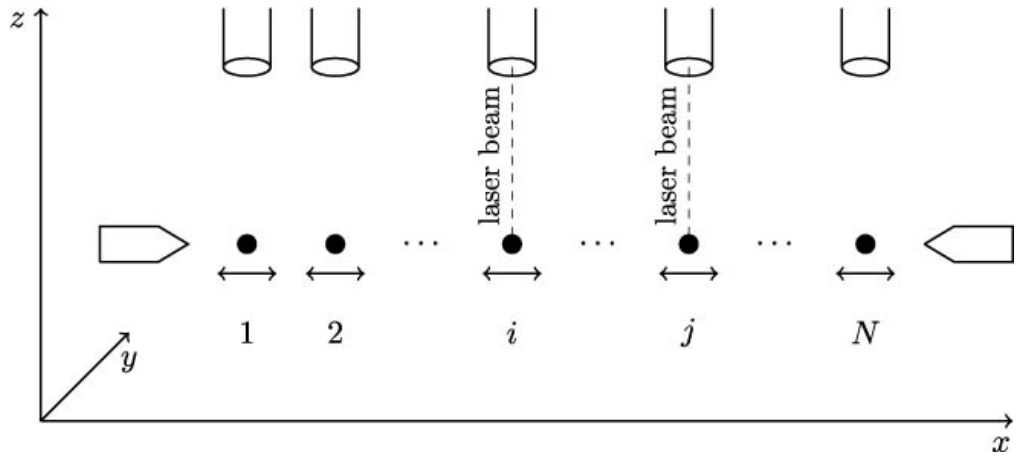


Figure 6.34: Schematic representation of the linear ion trap.

Applying the laser beam at the right frequency, it is possible to exclusively excite either a single ion or the CM mode. Addressing a single ion (and, thus, implementing a one-qubit gate) is straightforward. Let us see how we can implement a multi-qubit gate needed to create entanglement.

The following protocol that implements a two-qubit gate was proposed by Cirac and Zoller [68]. First, we note that the excited state $|e\rangle_i$ is not unique and depends on the polarisation of the laser beam applied to ion i. If we have two possible polarisations, which we denote as $q = 0$ and $q = 1$, then the corresponding excited states are denoted as $|e_0\rangle_i$ and $|e_1\rangle_i$. The computational basis is $\{|g\rangle_i, |e_0\rangle_i\}$. The protocol reads as follows:

1. Apply a π laser pulse with polarisation $q = 0$ to excite the i-th ion. The π laser pulse has the meaning of a laser pulse applied for time π/ω, where ω is a characteristic frequency of the trapped ion system.
2. Direct the laser to the j-th ion and turn it on for a time of a 2π pulse with polarisation $q = 1$.

3. Direct the laser back to the i-th ion and turn it on for a time of a π pulse with polarisation $q = 0$.

The effect of this procedure is to change the sign of the state only when both ions are initially excited as shown in Table 6.5.

$$|g\rangle$$

Table 6.5: Two-qubit gate (CPHASE) with the trapped ion qubits.

We note that the state of the CM mode is restored to the initial state $|0\rangle$ (no phonon) after the process. The protocol realises the CPHASE two-qubit gate.

Trapped ion-based qubits are characterised by longer coherence time (how long the quantum state survives) and higher fidelity (accuracy of the gate operations) in comparison with the superconducting qubits. On the flip side, superconducting qubits enjoy orders of magnitude shorter gate times.



Similar to many possible physical realisations of a classical bit, there exist many competing quantum computing technologies. It is too early to say which one will become the ultimate winner in the long run, or whether multiple technologies will co-exist by occupying their respective niches.

We now have all the tools, theoretical and physical, to actually build quantum circuits. In the next section, we discuss the quantum hardware and quantum simulators that would allow us to do it.

6.8 Quantum Hardware and Simulators

The current state-of-the-art quantum computing technology demonstrates impressive qubit fidelity and coherence time:

- Qubits made of superconducting circuits (coherence time: ~ 10μs) [164]

one-qubit gate		two-qubit gate	
Gate time:	~ 10^{-2}μs	Gate time:	~ 10^{-2}-10^{-1}μs
Fidelity:	99.9%	Fidelity:	99.7%

Table 6.6: Superconducting qubits.

- Qubits made of trapped ions (coherence time: > 10^7μs) [46]

one-qubit gate		two-qubit gate	
Gate time:	~ 1-10μs	Gate time:	~ 10μs
Fidelity:	99.9999%	Fidelity:	99.9%

Table 6.7: Trapped ion qubits.

Even more importantly, the pace of technological improvements remains very fast. Recent experiments conducted by MIT researchers on superconducting qubits [277] demonstrated a possibility to sharply reduce errors in two-qubit gates, bringing two-qubit gate fidelity for CZ and iSWAP gates to near 99.9%. Interestingly, this improvement in two-qubit gate fidelity has been achieved through the introduction of tunable couplers (Figure 6.30 schematically shows a two-qubit interaction via a coupler). To eliminate the error generating qubit-qubit interactions, the higher energy levels of the coupler were used to cancel out the problematic interactions. As was schematically shown in Figure 6.7.2, the higher energy levels are usually ignored, although they have a non-negligible contribution. Better control and design of the coupler is key to tailoring the qubit-qubit interaction as needed.

This is a big step towards implementing *error correction*: additional qubits can be added to improve the robustness of quantum computation. Qubit errors can be actively addressed by adding redundancy. However, in order for the hardware redundancy to be practical, higher qubit fidelity is required. Different error correction protocols require different fidelity thresholds, and 99.9% two-qubit gate fidelity is not a bad place to start.

However, we are still some years away from sufficiently fault-tolerant quantum computers. This is why it is useful (and even necessary) to experiment with quantum simulators – classical computers operating according to the logic of quantum computing. There is nothing preventing classical digital computers from operating according to the laws of quantum computing except for prohibitive memory requirements. The state of an *n*-qubit quantum system can be stored in classical memory as 2^n probability amplitudes. This prevents most classical computers from performing quantum computing operations on more than 35-40

qubits. But it is perfectly feasible to run quantum computing programs on up to 25-30 quantum registers.

Although quantum simulators can only operate on a relatively small number of qubits, they are *ideal* quantum computers that do not suffer from any type of quantum hardware imperfections. This makes them invaluable in testing principles and small-scale versions of quantum algorithms. They can be used for the proof-of-concept and to help develop new ideas in a situation where actual quantum hardware is still too noisy and not readily available.

There are many open-source quantum simulators (and, even, specialised quantum computing programming languages). In this book, we investigate the performance of various quantum algorithms using `Qiskit` [238] – an open-source Python package that implements the logic of quantum computing in an intuitive and user-friendly way. Qiskit also owes its popularity to a well-written and highly educational textbook with many well-thought-out examples that make it a pleasure to learn the principles of quantum computing. The `Qiskit` package and the textbook are available from

`https://qiskit.org/`

and

`https://qiskit.org/textbook/preface.html`

The field of quantum programming is growing fast and several languages or software development kits are now competing. Quantum instruction sets such as `Quil`, `cQASM`, `OpenQASM`, and `Blackbird` allow us to translate high-level algorithms into physical instructions run on quantum computers. They are used in quantum software development kits (QSDKs) to represent quantum circuits. The most important QSDKs at the time of writing are

- `Ocean` (D-Wave)
- `Forest` (Rigetti)
- `Qiskit` (IBM)
- `Cirq` (Google)
- `Quantum Development Kit` (Microsoft)
- `Braket SDK` (Amazon)
- `ProjectQ` (ETH Zurich)
- `Strawberry Fields` (Xanadu)

Parallel to these QSDKs, quantum programming languages have been developed, both of the imperative type (step-by-step instructions) such as `QCL`, `QMASM`, and `Silq` and of the functional type such as `QML`, `Quantum Lambda Calculus`, `QFC`, `QPL`, and `Q#`.



We observed an exceptionally fast pace of quantum computing hardware development over the last several years, with multiple technological breakthroughs. Additionally, the progress on the quantum software development side assists in relaxing requirements for the physical qubits needed to build fault-tolerant quantum computers.

Summary

In this chapter, we covered and contrasted the basic elements of classical and quantum computing. We started with the concept of a fundamental memory unit (bit) and functions that transform the memory states (logic gates). We also provided examples of the possible physical realisations of the logic gates and a memory cell – this highlights the fundamental dualism of both classical and quantum computing: computation is substrate independent but its practical realisation requires the existence of a suitable physical system.

Then, we introduced the concept of the qubit and its canonical mathematical representation. Visualisation of a qubit with the help of the Bloch sphere allows for the natural representation of single qubit quantum gates as rotation operators.

Next, we studied two-qubit gates and their matrix representation. We learned how to assemble one-qubit and multi-qubit gates into quantum circuits – a good example is the Bell circuit, which creates a maximally entangled state of two qubits from the completely unentangled initial states. We also touched on the important topics of reversible computing and cutting edge quantum hardware.

In the next chapter, we will introduce a particular type of quantum circuit, so-called Parameterised Quantum Circuits, that provide great flexibility for applications. We will also explore various data encoding schemes – the mapping of samples from the classical datasets into the corresponding quantum states.

Join our book's Discord space

Join our Discord community to meet like-minded people and learn alongside more than 2000 members at: https://packt.link/quantum



7
Parameterised Quantum Circuits and Data Encoding

Having built the quantum hardware, how can we use it to the maximum effect given its scale, connectivity, and fidelity rate? This question can be best answered if we split it into two parts. First, what problems are in principle solvable on NISQ computers? Second, how do we encode classical data into quantum states?

The rest of this book focuses on the first part: problems and models that can be formulated in a way that doesn't require a massive number of qubits and that are, at least to some extent, noise tolerant. The first step in this direction is the concept of the Parameterised Quantum Circuit (PQC) as a generic quantum machine learning model.

The second part – data encoding – is equally important and relies on several practical methods described in this chapter. This is an active area of research where we can expect most of the progress to come from the quantum software side.

7.1 Parameterised Quantum Circuits

We have seen how to combine quantum gates to form arbitrarily wide and deep quantum circuits. A quantum circuit transforms an initial quantum state $|\psi\rangle$ into a final quantum state $|\psi'\rangle$ by applying a sequence of unitary operators:

$$|\psi'\rangle = \mathtt{U}_m(\theta_m)\ldots\mathtt{U}_2(\theta_2)\mathtt{U}_1(\theta_1)\,|\psi\rangle\,.$$

Here, $(\mathrm{U}_i)_{i=1,\ldots,m}$ and $(\theta_i)_{i=1,\ldots,m}$ denote respectively the individual gates and the associated vectors of gate parameters. Some gates may be *fixed* (e.g., a two-qubit CNOT gate viewed as a controlled rotation of the target

qubit state around the x-axis by a fixed angle $\theta = \pi$) while some gates may be *adjustable* (e.g., a one-qubit $R_X(\theta)$ gate that rotates the qubit state around x-axis by an arbitrary angle $\theta \in [-\pi, \pi]$).

Once the final quantum state $|\psi'\rangle$ is constructed, the individual qubits can be measured. After measurement the qubit states stay the same in the basis in which they were measured, which we always assume to be the standard computational basis unless explicitly specified otherwise. Therefore, the final output of running the quantum circuit and then measuring the qubits (not necessarily all qubits have to be measured) is a classical bitstring.

What we just described is a parameterised quantum circuit schematically shown in Figure 7.1. The PQC can be used in many different ways. First of all, let us note that the PQC can be *trained*. Training the PQC has the meaning of finding an optimal set of adjustable parameters (the vectors $\theta_1, \ldots, \theta_m$ above, for example) given the overall PQC *ansatz* (architecture). The meaning of "optimal" is problem dependent but generally means a configuration of adjustable parameters that ensures maximum closeness of the final quantum state $|\psi'\rangle$ to some desired target quantum state that corresponds to a particular probability distribution we aim to encode.

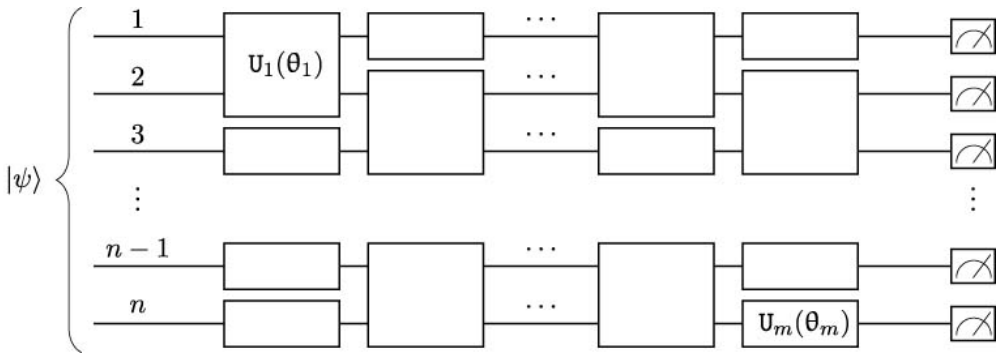


Figure 7.1: Schematic representation of a Parameterised Quantum Circuit.

In the following chapters, we will see how PQCs can be used as discriminative ML models (Chapter 8), as generative ML models (Chapter 9), and as optimisers (Chapters 10 and 11):

- In the case of the Variational Quantum Eigensolver, a PQC is used to construct the final quantum state $|\psi'\rangle$ that is close to the eigenstate of the problem Hamiltonian that corresponds to the smallest eigenvalue (the ground state energy that is linked to the minimum of the cost function).
- When we build a quantum discriminative model – a Quantum Neural Network trained as a classifier – we are interested in measuring only a handful of qubits (or even just a single qubit). This should give us the binary representation of the "class label" for the given sample. The input (initial quantum state $|\psi\rangle$) encodes the sample we want to classify.
- When our objective is to build a quantum generative model – the Quantum Circuit Born Machine – we measure all qubits. This gives us a bitstring that is a generated sample from the probability distribution encoded in the final quantum state $|\psi'\rangle$ constructed by the PQC. The initial state is initialised as $|0\rangle^{\otimes n}$.

PQCs are invariably trained using hybrid quantum-classical protocols. The hybrid approach is shown schematically in Figure 7.2 and consists of three components: the user, the classical computer, and the quantum computer [29].

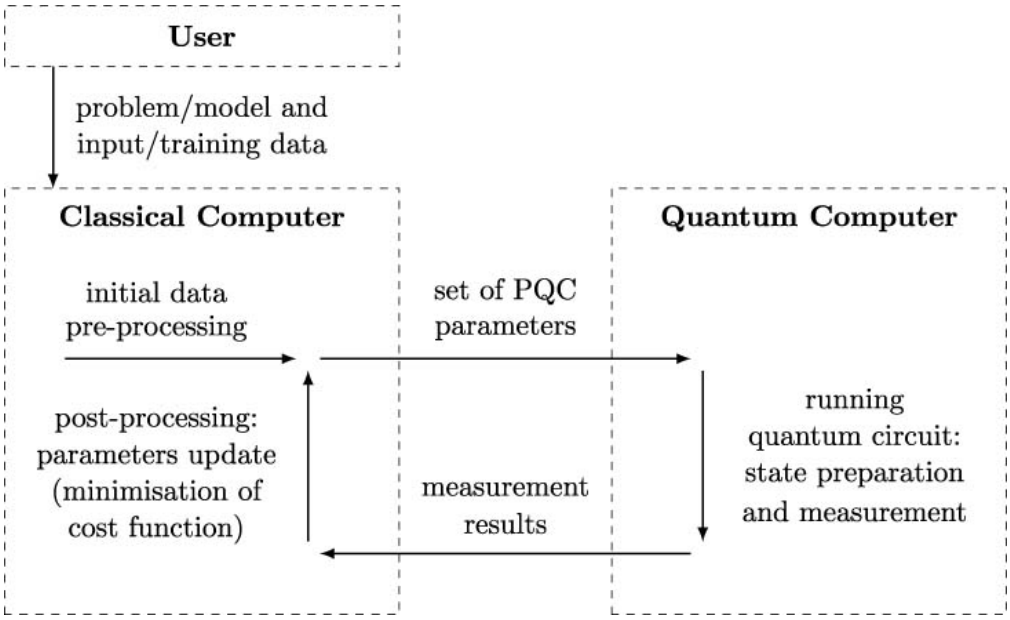


Figure 7.2: Training PQCs – schematic process.

The user provides the model for the problem; the classical computer pre-processes the data and produces the initial set of parameters for the PQC; the quantum computer runs the PQC by preparing the quantum state as prescribed by the PQC and by performing measurements. Measurement outcomes are then post-processed by the classical computer, which updates the model parameters as per the chosen training algorithm (backpropagation of error with gradient descent, non-differentiable learning method, etc.) The overall algorithm is run in a closed loop between the classical and quantum computers.



PQC is one of the most successful frameworks for applying NISQ computers to solving non-trivial real-world problems. It follows the paradigm of a hybrid quantum-classical computational protocol and can be used for experimenting with the wide range of quantum machine learning models.

7.2 Angle Encoding

Let us go back to the Bloch sphere (Figure 7.2) that visualises the canonical representation of the qubit state – a unit vector in the 2-dimensional complex vector space:

$$|\psi\rangle = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) \\ e^{i\phi}\sin\left(\frac{\theta}{2}\right) \end{bmatrix}.$$

The angles $\theta \in [0,\pi]$ and $\varphi \in [0,2\pi]$ uniquely determine the position of the qubit on the unit sphere. Since we need two continuous variables to specify the qubit state, a single qubit can encode two real-valued features.

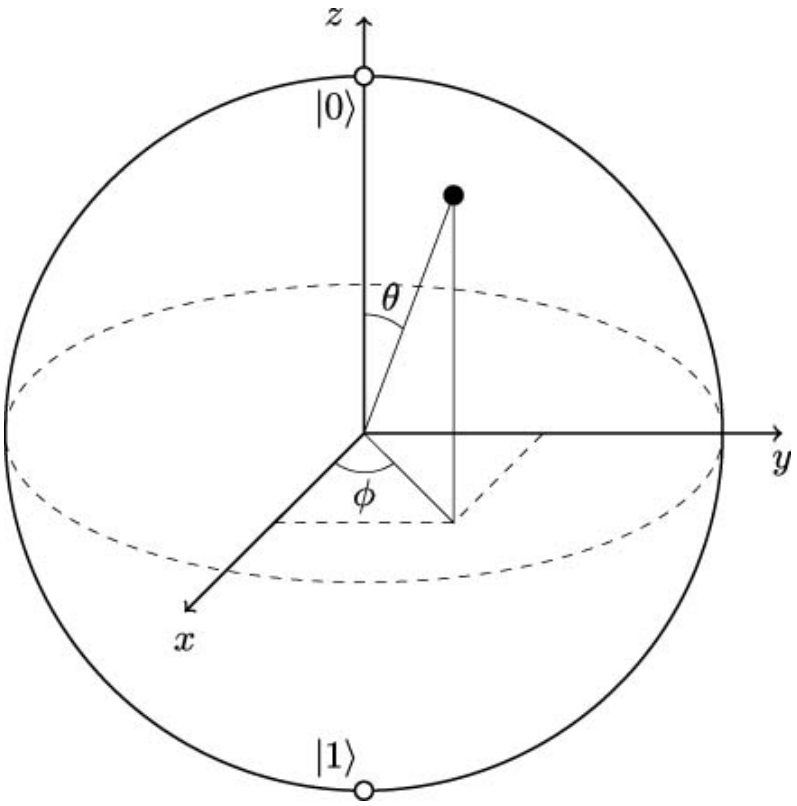


Figure 7.3: Quantum state $|\psi\rangle$ on the Bloch sphere.

7.2.1 The basic encoding scheme

We can illustrate this with the following schematic example: let us assume that we have an 8-feature dataset consisting of N samples and that all features $X_1,\ldots,X_8$ are real-valued and such that their extremal values, $X_i^{\min}$ and $X_i^{\max}$, $i = 1,\ldots,8$, can be computed. Then, for every sample $j = 1, \ldots,N$ from the dataset, we can establish a one-to-one mapping between the values of the features X_i^j and the corresponding rotation angles θ_i^j:

$$\theta_i^j = \frac{X_i^j - X_i^{\min}}{X_i^{\max} - X_i^{\min}}\pi,$$

where $X_i^{\min} := \min_j X_i^j$ and $X_i^{\max} := \max_j X_i^j$. Rotation angles θ_i^j generalise angles θ and φ in Figure 7.2.

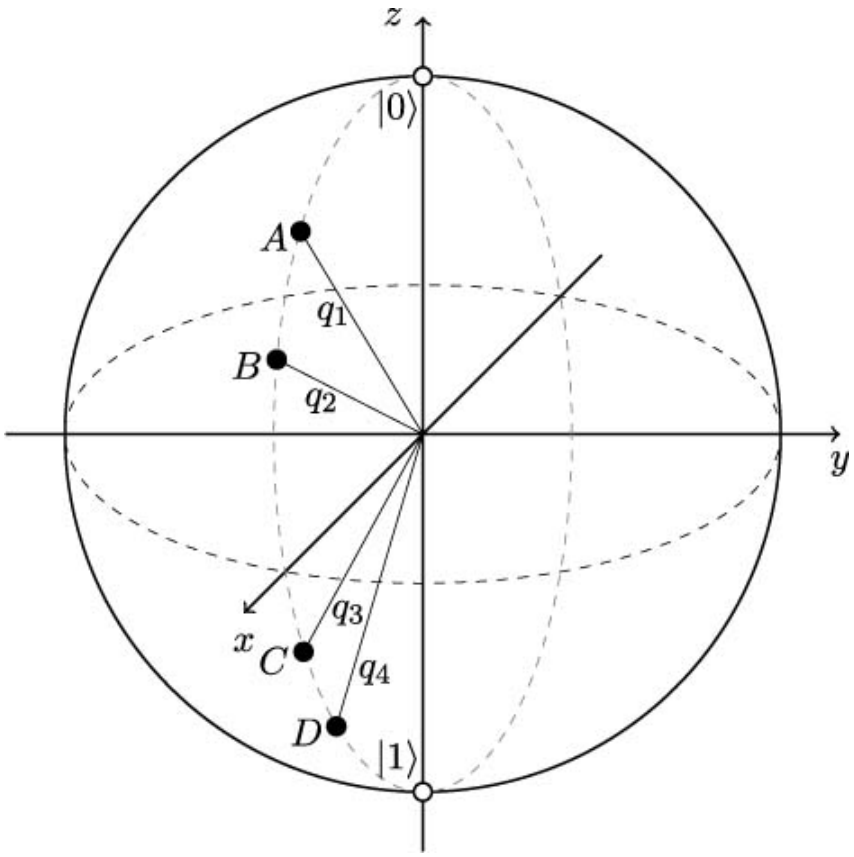


Figure 7.4: Feature encoding – rotations around y axis. States of qubits $q_1,\ldots,q_4$ are shown on the same Bloch sphere. The initial state of all qubits is $|0\rangle$ and the end states after rotations around the y-axis by angles $\theta_1,\ldots,\theta_4$ are denoted as A, B, C, and D.

The 8-feature sample can be encoded in a 4-qubit state (unentangled). For example, starting with four quantum registers initialised as $|0\rangle$ in the computational basis, we can first perform rotations around the y-axis: rotation by θ_1 for qubit 1, rotation by θ_2 for qubit 2, and so on. This is shown schematically in Figure 7.2.1 where qubits move from their initial state $|0\rangle$ to states A, B, C, and D.

7.2.2 Encoding two features per quantum register

After that, we encode the remaining features by performing rotations around the z-axis: rotation by θ_5 for qubit 1, rotation by θ_6 for qubit 2, and so on as shown in Figure 7.5.

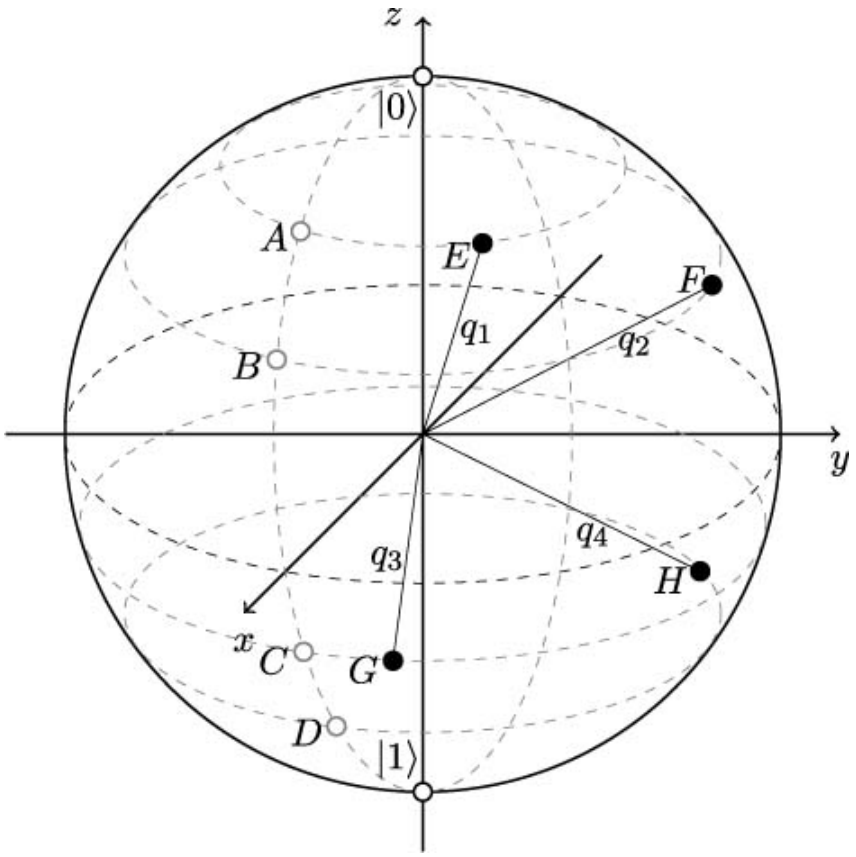


Figure 7.5: Feature encoding – rotations around the z-axis. The initial states of qubits q_1,…,q_4 are A,B,C,D. After rotation around the z-axis by angles θ_5,…,θ_8, the final qubit states are E,F,G,H.

The qubit states move from *A* to *E*, from *B* to *F*, from *C* to *G*, and from *D* to *H*. The corresponding quantum circuit looks as follows:

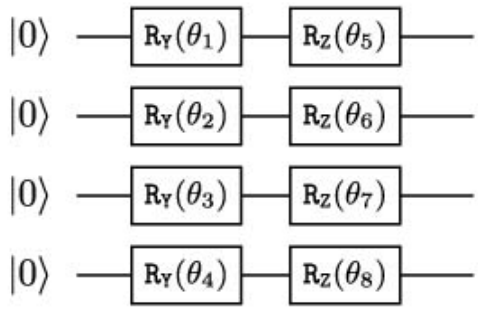


Figure 7.6: 4-qubit quantum circuit for 8-feature sample encoding.

7.2.3 Mapping a classical data sample into a quantum state

We can summarise the angle encoding scheme as follows. For the classical data sample $X^j := (X_1^j,\ldots,X_K^j) \in \mathbb{R}^K$, $j = 1,\ldots,N$, angle encoding works by constructing the map

$$X^j \longmapsto \bigotimes_{i=1}^{K} \left(\cos\left(\frac{\theta_i^j}{2}\right) |0\rangle + \sin\left(\frac{\theta_i^j}{2}\right) |1\rangle \right),$$

where angles $(\theta_i^j)_{i=1,\ldots,K;j=1,\ldots,N}$ are given by the expression (7.2.1). This scheme only requires one rotation gate for each qubit, hence encodes as many features as the number of qubits. However, we know that a single quantum register can encode two real variables. The following scheme maps the classical sample into the quantum state with the help of an extra phase gate:

$$X^j \longmapsto \bigotimes_{i=1}^{K} \left(\cos\left(\frac{\theta_{2i-1}^j}{2}\right) |0\rangle + \exp\left(i\theta_{2i}\right) \sin\left(\frac{\theta_{2i-1}^j}{2}\right) |1\rangle \right).$$

This scheme allows us to encode $2n$ features with n qubits.



n quantum registers have capacity to encode 2*n* continuous features with just two layers of one-qubit gates.

7.3 Amplitude Encoding

So far we have not utilised the information encoding possibilities provided by entanglement although, in principle, most of the information in large quantum systems can be stored in correlations. In the case of our 8-feature dataset example considered in the previous section, we can reduce the number of necessary qubits to just three if we use entanglement. The first six rotation angles $\theta_1,\ldots,\theta_6$ can still be used for the single-qubit rotations R_Y,R_Z. The last two, θ_7 and θ_8, can be used for controlled rotations that entangle qubits 1 and 2 and qubits 2 and 3 as shown in Figure 7.7:

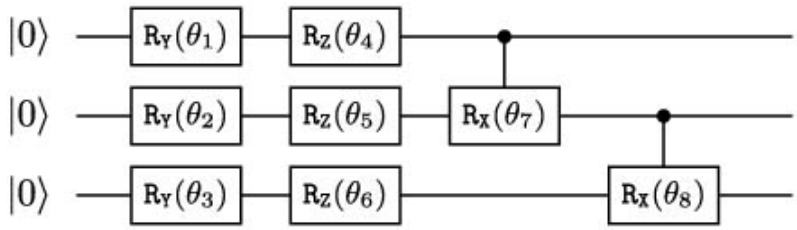


Figure 7.7: 3-qubit quantum circuit for 8-feature sample encoding.

In principle, since the n-qubit state can be uniquely described by specifying 2^n probability amplitudes, we only need n qubits to encode 2^n features. However, this superdense encoding is not always practical or desirable. The *amplitude encoding* was used in the seminal work by Harrow, Hassidim, and Lloyd [126], but obtaining the amplitude encoding is a non-trivial task for problems of realistic size and this, usually, is the main bottleneck for many quantum algorithms [184].

The amplitude encoding can be formalised in the following way. Consider an N-dimensional (with $N = 2^n$) data point $\mathrm{x} := (x_1,\ldots,x_N) \in \mathbb{C}^N$. We can associate quantum amplitudes to the coordinates as

$$|\psi_{\mathrm{x}}\rangle = \frac{1}{\|\mathrm{x}\|}\sum_{i=1}^{N} x_i\,|i\rangle\,,$$

where $\|\mathrm{x}\| := \sum_{i=1}^{N} |x_i|^2$ is the normalisation factor. We can therefore encode the dataset $\mathrm{D} := (\mathrm{x}^1,\ldots,\mathrm{x}^M)$ consisting of M points in $\mathbb{R}^N$ as

$$|\mathrm{D}\rangle = \frac{1}{C_{\mathrm{D}}}\sum_{i=1}^{2^p} \overline{x}_i\,|i\rangle\,,$$

for some integer p, where

$$\overline{\mathrm{x}} = (\overline{x}_i)_{i=1,\ldots,2^p} = (x_1^1,\ldots,x_N^1,x_1^2,\ldots,x_N^2,\ldots,x_1^M,\ldots,x_N^M) \in \mathbb{R}^{MN}$$

is the concatenation of all the data points and C_{D} is a normalisation constant. Here the constraint is hence that $2^p \geq MN$, namely $p \geq \log_2(MN)$. Note that there may again be some sparsity in the case where $2^p > MN$.

The clear advantage is that it can store 2^n features with only n qubits, but unfortunately has a depth $\mathcal{O}(2n)$ and is hence hard to build.



With the amplitude encoding, n quantum registers have capacity to encode 2^n continuous features. However, this requires construction of the deep quantum circuits with the circuit depth growing as $\mathcal{O}(2n)$. This solution may not be practical for NISQ computers when dealing with a large number of features.

7.4 Binary Inputs into Basis States

Consider a real number $x \in \mathbb{R}$ approximated with the binary representation

$$x \approx \widehat{x} = (x_{\mathfrak{i}}, x_{\mathfrak{i}-1}, \cdots, x_{-\mathfrak{d}}) := (-1)^{x_{\mathfrak{i}}} \left(\sum_{j=0}^{\mathfrak{i}-1} x_j 2^j + \sum_{j=1}^{\mathfrak{d}} x_{-j} 2^{-j} \right)$$
$$\longmapsto |x_{\mathfrak{i}} x_{\mathfrak{i}-1} \cdots x_{-\mathfrak{d}}\rangle =: |x\rangle ,$$

for some non-negative integers $\mathfrak{i},\mathfrak{d}$, where $x_{\mathfrak{i}} \in \{0,1\}$ accounts for the sign of x and $(x_j)_{j=0,\ldots,\mathfrak{i}-1} \in \{0,1\}^{\mathfrak{i}}$ and $(x_j)_{j=-1,\ldots,-\mathfrak{d}} \in \{0,1\}^{\mathfrak{d}}$ represent respectively the integer and decimal parts of x. Consider now a vector $\mathrm{x} := (x^1,\ldots,x^N) \in \mathbb{R}^N$. We can concatenate all the binary approximations $x^1,\ldots,x^N$ into one vector

$$\left(x_{\mathfrak{i}}^1, x_{\mathfrak{i}-1}^1, \cdots, x_{-\mathfrak{d}}^1, \cdots, x_{\mathfrak{i}}^N, x_{\mathfrak{i}-1}^N, \cdots, x_{-\mathfrak{d}}^N\right) \in \{0,1\}^{(1+\mathfrak{i}+\mathfrak{d})N}$$

to obtain a quantum state representation with $(1 + \mathfrak{i} + \mathfrak{d})N$ qubits of the form

$$|x_{\mathfrak{i}}^1 x_{\mathfrak{i}-1}^1 \cdots x_{-\mathfrak{d}}^1 \cdots x_{\mathfrak{i}}^N, x_{\mathfrak{i}-1}^N \cdots x_{-\mathfrak{d}}^N\rangle .$$

Since the vector thus obtained only contains 0 and 1, starting from the quantum state $|0\rangle^{\otimes(1+i+\mathfrak{d})N}$, we only need to apply the NOT gate X whenever the corresponding value is equal to one, so that the encoding circuit simply reads

$$|0\rangle^{\otimes(1+i+\mathfrak{d})N} \longmapsto \bigotimes_{l=1}^{N} \bigotimes_{k=-\mathfrak{d}}^{i} \mathrm{X}^{x_k^l} \, |0\rangle^{\otimes(1+i+\mathfrak{d})N} .$$

While the algorithm is straightforward and only requires the use of the single-qubit quantum gate X, it requires a large number of qubits and is in general not efficient in practice. Indeed, for a given dimension N, there are 2^N possible basis states. If a dataset contains only M points with M being much smaller than N, the quantum representation will therefore be sparse.

Example: Consider a dataset D = (x^1,x^2) with $x^1,x^2 \in [-2,2]$, each approximated with four qubits:

$$x^1 \approx (-1)^{x_1^1}(2^0 x_0^1 + 2^{-1}x_{-1}^1 + 2^{-2}x_{-2}^1) = (-1)^{x_1^1}\left(x_0^1 + \frac{1}{2}x_{-1}^1 + \frac{1}{4}x_{-2}^1\right)$$

and

$$x^2 \approx (-1)^{x_1^2}(2^0 x_0^2 + 2^{-1}x_{-1}^2 + 2^{-2}x_{-2}^2) = (-1)^{x_1^2}\left(x_0^2 + \frac{1}{2}x_{-1}^2 + \frac{1}{4}x_{-2}^2\right),$$

with $x_k^i \in \{0,1\}$ for each $i = 1,2$ and $k = -2,-1,0,1$. Their quantum embeddings therefore read $|x_1^1 x_0^1 x_{-1}^1 x_{-2}^1\rangle$ and $|x_1^2 x_0^2 x_{-1}^2 x_{-2}^2\rangle$, and the quantum circuit to encode the dataset therefore takes the form

$$|0\rangle^{\otimes 8} \longmapsto \left[\left(\mathrm{X}^{x_1^1} \otimes \mathrm{X}^{x_0^1} \otimes \mathrm{X}^{x_{-1}^1} \otimes \mathrm{X}^{x_{-2}^1}\right) \otimes \left(\mathrm{X}^{x_1^2} \otimes \mathrm{X}^{x_0^2} \otimes \mathrm{X}^{x_{-1}^2} \otimes \mathrm{X}^{x_{-2}^2}\right)\right] |0\rangle^{\otimes 8} .$$

7.5 Superposition Encoding

As developed in [287, 294], it is possible to build such a superposition of data in time linear in the number of points and features. We consider

again a dataset D := (x^1,…,x^M), with x^k := (x_1^k,…,x_n^k) ∈{0,1}^{n} for each k = 1, …,M. We use a quantum system of the form

$$|\psi_0\rangle := |0\rangle^{\otimes n} |00\rangle |0\rangle^{\otimes n},$$

where the left-most part with n qubits is called the loading register while the right-most one (also with n qubits) is the storage register. The middle one is an ancilla register that will be used to control manipulations between the loading and storage registers. The encoding algorithm works recursively. We first apply an Hadamard gate to the second ancilla qubit and store the first data point x^1 into the storage register. Since

$$\left(\bigotimes_{i=1}^{n} \mathtt{X}^{x_i^1}\right) |0\rangle^{\otimes n} = |x_1^1 \cdots x_n^1\rangle = |\mathrm{x}^1\rangle,$$

this can be achieved (after the Hadamard operation) by applying the unitary operator

$$\mathtt{I}^{\otimes n} \otimes \mathtt{I} \otimes \mathtt{H} \otimes \left(\bigotimes_{i=1}^{n} \mathtt{X}^{x_i^1}\right)$$

controlled with the second ancilla qubit, and the resulting quantum state reads

$$|\psi_1\rangle := \frac{|0\rangle^{\otimes n} |00\rangle |0\rangle^{\otimes n}}{\sqrt{2}} + \frac{|0\rangle^{\otimes n} |01\rangle |\mathrm{x}^1\rangle}{\sqrt{2}}.$$

This can easily be turned (see the proof of Lemma 7 below) into

$$|\psi_1\rangle = \frac{|0\rangle^{\otimes n} |00\rangle |\mathrm{x}^1\rangle}{\sqrt{2}} + \frac{|0\rangle^{\otimes n} |01\rangle |0\rangle^{\otimes n}}{\sqrt{2}}.$$

After m steps, we arrive at a quantum state of the form

$$|\psi_m\rangle := \frac{1}{\sqrt{M}} \sum_{k=1}^{m} |0\rangle^{\otimes n} |00\rangle |\mathrm{x}^k\rangle + \sqrt{\frac{M-m}{M}} |0\rangle^{\otimes n} |01\rangle |0\rangle^{\otimes n}.$$

The following lemma guarantees the validity of the algorithm:

Lemma 7. *There exists a unitary operator* U *such that*

$$\mathtt{U}\,|\psi_m\rangle = \frac{1}{\sqrt{M}} \sum_{k=1}^{m+1} |0\rangle^{\otimes n}\,|00\rangle\,\left|\mathrm{x}^k\right\rangle + \sqrt{\frac{M-(m+1)}{M}}\,|0\rangle^{\otimes n}\,|01\rangle\,|0\rangle^{\otimes n} =: |\psi_{m+1}\rangle$$

Proof. The proof is constructive and shows precisely what the operator U looks like.

1. Construct the successive maps

$$\begin{aligned}
|\psi_m\rangle = \quad & \frac{1}{\sqrt{M}} \sum_{k=1}^{m} |0\rangle^{\otimes n}\,|00\rangle\,\left|\mathrm{x}^k\right\rangle + \sqrt{\frac{M-m}{M}}\,|0\rangle^{\otimes n}\,|01\rangle\,|0\rangle^{\otimes n}\,, \\
\longmapsto & \frac{1}{\sqrt{M}} \sum_{k=1}^{m} |\mathrm{x}^{m+1}\rangle\,|00\rangle\,\left|\mathrm{x}^k\right\rangle + \sqrt{\frac{M-m}{M}}\,|\mathrm{x}^{m+1}\rangle\,|01\rangle\,|0\rangle^{\otimes n}\,, \\
\longmapsto & \frac{1}{\sqrt{M}} \sum_{k=1}^{m} |\mathrm{x}^{m+1}\rangle\,|00\rangle\,\left|\mathrm{x}^k\right\rangle + \sqrt{\frac{M-m}{M}}\,|\mathrm{x}^{m+1}\rangle\,|01\rangle\,|\mathrm{x}^{m+1}\rangle\,, \\
\longmapsto & \frac{1}{\sqrt{M}} \sum_{k=1}^{m} |\mathrm{x}^{m+1}\rangle\,|00\rangle\,\left|\mathrm{x}^k\right\rangle + \sqrt{\frac{M-m}{M}}\,|\mathrm{x}^{m+1}\rangle\,|11\rangle\,|\mathrm{x}^{m+1}\rangle =: \left|\widetilde{\psi}_m\right\rangle.
\end{aligned}$$

   The first one is easily achieved by applying the operator $\left(\bigotimes_{i=1}^{n} \mathtt{X}^{x_i^{m+1}}\right)\otimes \mathrm{I}^{\otimes 2} \otimes \mathrm{I}^{\otimes \mathrm{n}}$ to $|\psi_m\rangle$. The second step is realised with controlled gates using the second qubit of the ancilla register as control. The last one is trivial with a CNOT gate on the first ancilla qubit using the second ancilla qubit as control.

2. Now define the unitary gate

$$\widetilde{\mathtt{U}} := \frac{1}{\sqrt{M-m}} \begin{pmatrix} \sqrt{M-m-1} & 1 \\ -1 & \sqrt{M-m-1} \end{pmatrix},$$

   and note that its controlled (by the first ancilla qubit a_1) version ${}_{a_1}\mathrm{U}$ acts as

$$\begin{aligned}
{}_{a_1}\mathrm{U}|00\rangle \quad &= |00\rangle, \\
{}_{a_1}\mathrm{U}|11\rangle \quad &= \frac{1}{\sqrt{M-m}}|1\rangle\otimes\left(|0\rangle + \sqrt{M-m-1}\,|1\rangle\right) =
\end{aligned}$$

$$\frac{|10\rangle + \sqrt{M-m-1}\,|11\rangle}{\sqrt{M-m}}.$$

Applying it to the ancilla register of $|\widetilde{\psi}_m\rangle$ in Step 1 (and leaving all other qubits unchanged) yields

$$= \mathrm{I}^{\otimes n} \otimes {}_{a_1}\mathrm{U} \otimes \mathrm{I}^{\otimes n} |\widetilde{\psi}_m\rangle$$

$$= \frac{1}{\sqrt{M}} \sum_{k=1}^{m} |x^{m+1}\rangle |00\rangle |x^k\rangle$$

$$+ \sqrt{\frac{M-m}{M}} |x^{m+1}\rangle \left\{ \frac{|10\rangle + \sqrt{M-m-1}\,|11\rangle}{\sqrt{M-m}} \right\} |x^{m+1}\rangle$$

$$= \frac{1}{\sqrt{M}} \sum_{k=1}^{m} |x^{m+1}\rangle |00\rangle |x^k\rangle$$

$$+ \frac{1}{\sqrt{M}} |x^{m+1}\rangle \left\{ |10\rangle + \sqrt{M-m-1}\,|11\rangle \right\} |x^{m+1}\rangle$$

We then flip the first ancilla qubit to 0 in the $|10\rangle$ case (easily achievable with SWAP and CNOT gates) and, regrouping the same ancilla terms together we obtain

$$\frac{1}{\sqrt{M}} \sum_{k=1}^{m+1} |x^{m+1}\rangle\, |00\rangle\, |x^k\rangle + \sqrt{\frac{M-(m+1)}{M}}\, |x^{m+1}\rangle\, |11\rangle\, |x^{m+1}\rangle.$$

Resetting the registers as in (7.5) to obtain

$$\frac{1}{\sqrt{M}} \sum_{k=1}^{m+1} |0\rangle^{\otimes n}\, |00\rangle\, |x^k\rangle + \sqrt{\frac{M-(m+1)}{M}}\, |0\rangle^{\otimes n}\, |01\rangle\, |0\rangle^{\otimes n}$$

finishes the proof of the lemma.

□

7.6 Hamiltonian Simulation

Hamiltonian encoding, popular in quantum machine learning, is inspired by the Schrödinger equation (1.2.2), which reads

$$i\hbar\frac{d|\psi(t)\rangle}{dt} = \mathcal{H}|\psi(t)\rangle,$$

for some Hamiltonian $\mathcal{H}$, where $\hbar$ is the Planck constant, and subject to some boundary condition at $t = 0$. The solution to the equation reads

$$|\psi(t)\rangle = \exp\left(-\frac{i\mathcal{H}t}{\hbar}\right)|\psi(0)\rangle .$$

The idea of Hamiltonian encoding is to encode the initial data into the Hamiltonian $\mathcal{H}$. Consider a cloud of points $X \in \mathcal{M}_{n,n}(\mathbb{C})$. If X is Hermitian, we can define the Hamiltonian matrix $\mathcal{H}_X := X$, otherwise the augmented version

$$\mathcal{H}_X := \begin{pmatrix} \mathbf{0}_{n,n} & X \\ X^\dagger & \mathbf{0}_{n,n} \end{pmatrix}$$

is Hermitian by construction.

Our aim is, for a given precision level ε, to find a state $|\widetilde{\psi}\rangle$ (or an algorithm generating this state) such that

$$\left\| |\widetilde{\psi}\rangle - |\psi(t)\rangle \right\| \leq \varepsilon,$$

for some given norm $\|\cdot\|$, where $|\psi(t)\rangle$ solves the Schrödinger equation.

An important bottleneck of this method is the computation of $\exp(\mathcal{H}_X t)$ (where we ignore the ratio $-i\hbar$ for simplicity). Suppose that the Hamiltonian $\mathcal{H}_X$ can be written as a sum

$$\mathcal{H}_X = \sum_{i=1}^{p} \mathcal{H}_i$$

of easy-to-compute Hamiltonians $(\mathcal{H}_i)_{i=1,\ldots,p}$. If the latter do not commute, then the identity

$$\exp\left(\sum_{i=1}^{p}\mathcal{H}_i t\right)=\prod_{i=1}^{p}\mathrm{e}^{\mathcal{H}_i t}$$

does not hold, but the first-order Suzuki-Trotter [220, 286] formula (used by Lloyd in [191])

$$\exp\left(\sum_{i=1}^{p}\mathcal{H}_i t\right)=\prod_{i=1}^{p}\mathrm{e}^{\mathcal{H}_i t}+\mathcal{O}(t^2),$$

for t small enough, allows us to bypass this issue.

Indeed, even if t is not so small, we may pick $\delta > 0$ small enough and use the factorisation

$$\exp\left(\sum_{i=1}^{p}\mathcal{H}_i t\right)=\left[\exp\left(\sum_{i=1,\ldots,p}\mathcal{H}_i\delta\right)\right]^{t/\delta}=\left[\prod_{i=1}^{p}\mathrm{e}^{\mathcal{H}_i\delta}+\mathcal{O}(\delta^2)\right]^{t/\delta},$$

which has a small error (albeit with the caveat that the operation needs to be computed many times). In general, any n-qubit Hamiltonian $\mathcal{H}$ can be decomposed in at most 4^n elementary Hamiltonians (of the Pauli form) as

$$\mathcal{H}=\frac{1}{2^n}\sum_{i_1,\ldots,i_n\in\{\mathrm{I},\mathrm{X},\mathrm{Y},\mathrm{Z}\}}\mathrm{Tr}\left(\bigotimes_{k=1}^{n}\sigma_{i_k}\mathcal{H}\right)\bigotimes_{k=1}^{n}\sigma_{i_k},$$

where σ_{i_k} is a Pauli operator. Of course, 4^n appeals to Pauli operators may be too large in general, but local features of the Hamiltonian (such as sparse [34] or diluted or degree-reducted [8] Hamiltonians) help reduce the complexity.

Remark: An alternative approach, especially for QML problems analysed in the next chapter, is to encode the data using Quantum Random Access Memory (QRAM), essentially with the bucket-brigade algorithm developed in [110] (see also [140, 15]), and we refer the interested

reader to [65] for a good summary of the current state-of-the-art algorithms.



Encoding classical data into a quantum computer has seen many advances recently and several competing techniques are now available depending on the problem under investigation.

Summary

In this chapter, we introduced the concept of a parameterised quantum circuit as a generic QML model. PQCs can be trained and used as discriminative and generative QML models as well as optimisers. They can also be used to encode classical data samples into the corresponding quantum states.

We considered several popular data encoding methods. Arguably, the simplest and easiest to implement is the angle encoding algorithm – we shall use this approach in the next chapter. Other methods also have their strong points, although they tend to be either more demanding in terms of the hardware capabilities or better suited for some niche applications.

In the next chapter, we apply what we learned so far to the task of building the quantum neural network trained as a classifier and compare its performance on the binary classification problem with standard classical machine learning models.

Join our book's Discord space

Join our Discord community to meet like-minded people and learn alongside more than 2000 members at: https://packt.link/quantum



8
Quantum Neural Network

Quantum neural networks [100] are parameterised quantum circuits that can be trained as either generative or discriminative machine learning models in direct analogy with their classical counterparts. In this chapter, we will consider parameterised quantum circuits trained as classifiers. In the most general case, a classifier is a function that takes an N-dimensional input and returns one of M possible class values. The classifier can be trained on a dataset of samples with known class labels by adjusting the configurable model parameters in such a way as to minimise the classification error. Once the classifier is fully trained, it can be exposed to new unseen samples for which correct class labels are unknown. Therefore, it is critically important to avoid overfitting to the training dataset and ensure that the classifier generalises well to the new data.

There are many similarities between quantum and classical neural networks. In both cases, the key element is the forward propagation of the signal (input), which is transformed by the network activation functions. Both quantum and classical neural networks can be trained through the backpropagation of error (differentiable learning) as well as through various non-differentiable learning techniques. However, there are also fundamental differences. For example, classical neural networks derive their power from the non-linear transformation of input. In contrast, all quantum gates are linear operators and the power of quantum neural networks comes from the mapping of the input into the high-dimensional Hilbert space where classification can be more easily done.

8.1 Quantum Neural Networks

Figure 8.1 provides a schematic representation of a typical Quantum Neural Network (QNN) trained as a classifier. Let us have a look at the quantum circuit and understand how it operates. The network consists of n quantum registers, a number of one-qubit and two-qubit gates, and m measurement operators. The input is a quantum state $|\psi_k\rangle$ encoding the k-th sample from the dataset. If our dataset is classical, then every classical sample should first be encoded in the input quantum state (as explained in the previous chapter). With m measurement operators, the output is a bitstring that can encode up to 2^m integer values (class labels). In the case of a binary classifier, it is sufficient to perform measurement on a single qubit.

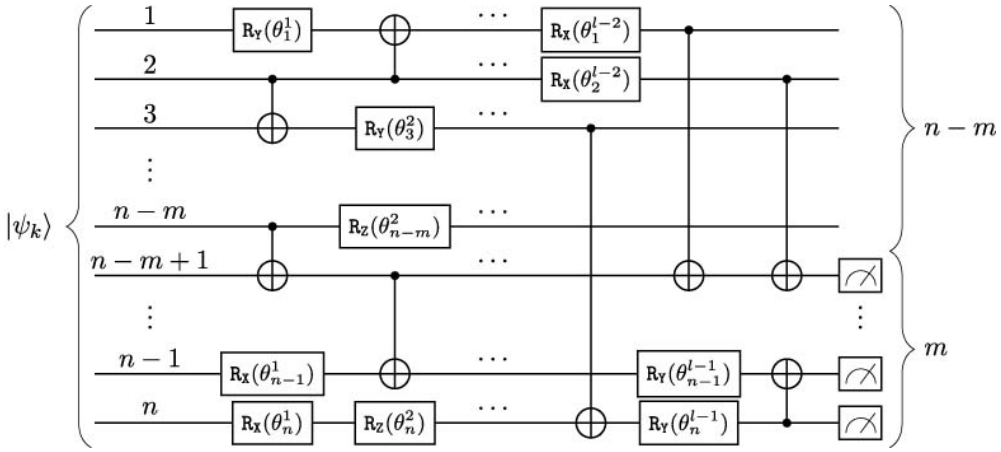


Figure 8.1: Schematic representation of a quantum neural network – parameterised quantum circuit – consisting of 1-qubit and 2-qubit gates and measurement operators on one or more quantum registers. The initial state $|\psi_k\rangle$ encodes the k-th sample from the dataset.

The measurement process produces a single sample from the probability distribution encoded in the quantum state. Therefore, we need to run the quantum circuit many times for the same input in order to collect sufficient statistics for each qubit on which we perform measurement.

For example, if our QNN is organised as a classifier that should be able to predict one of the four possible classes ("0", "1", "2", and "3"), then we would need to perform measurement on 2 qubits with possible outcomes $|00\rangle$ corresponding to class "0", $|01\rangle$ corresponding to class

"1", $|10\rangle$ corresponding to class "2", and $|11\rangle$ corresponding to class "3". Let us assume that we have run the quantum circuit 1,000 times and observed the following results as shown in Table 8.1:

Measured bitstring	Class label	Number of observations
00	0	100
01	1	550
10	2	200
11	3	150

Table 8.1: 1,000 runs of the quantum circuit.

Then we can conclude that the most likely class label for the given input is class "1" (with probability 55%). At the same time, we also obtain probabilities for all other possible class values, which may be useful in some cases.

The network is organised as l layers of one-qubit and two-qubit gates. The gates can be *adjustable*, meaning that they can be controlled by adjustable parameters, such as rotation angles, or they can be *fixed*. The 2-qubit gates in Figure 8.1 are fixed CX gates but, in principle, they can be adjustable controlled rotation gates. Although the network shown schematically in Figure 8.1 can have up to $n \times l$ adjustable parameters $(\theta_i^j)_{i=1,\ldots,n;\ j=1,\ldots,l}$, it is often the case that the two-qubit gates are fixed and we only have one-qubit rotations as available degrees of freedom in training the network.

Similar to classical neural networks, QNNs can be trained through either differentiable learning (for example, backpropagation of error with gradient descent) or non-differentiable learning (e.g., evolutionary search heuristics). Both approaches have their relative strengths and weaknesses. In theory, differentiable learning can be faster, but convergence is not guaranteed due to the well-known problem of

"barren plateaus" associated with the gradients becoming vanishingly small [207] and is problem dependent. Non-differentiable learning is, as a rule, slower but avoids being trapped in local minima and works well in situations where the cost function is not smooth. Sections 8.2 and 8.3 provide detailed descriptions of the QNN training procedures.

Obviously, the strongest motivation for using quantum classifiers is their ability to process quantum data. The input quantum states that must be classified may be outputs of some other quantum circuits. As we may not be able to store the information encoded in these quantum states classically, a quantum classifier becomes an indispensable tool. However, quantum classifiers have a realistic chance to demonstrate their advantage on purely classical data too. There are several considerations that motivate our interest in trying to apply QNNs to classical datasets.

First, parameterised quantum circuits possess a larger expressive power than equivalent classical neural networks. Second, they are structurally able to efficiently fight overfitting. Finally, quantum speedup is achievable on some types of quantum hardware for specific use cases even at these very early stages of quantum computing development. Chapter 12 investigates these questions in more detail.

In this chapter, we focus on using QNNs to efficiently solve specific finance-related classification use cases and provide a comparison with a number of standard classical classifiers. While experimentally proving quantum speedup and larger expressive power of QNNs requires powerful quantum hardware, the way QNNs fight overfitting can be verified on relatively small and shallow quantum circuits with the help of open-source quantum simulators.

> QNNs are PQCs trained as ML models such as classifiers. QNNs have a natural advantage over classical neural



networks when it comes to classifying quantum data. However, classical datasets can also be encoded as quantum states and processed by QNNs with their larger expressive power, their ability to efficiently fight overfitting, and, ultimately, with their quantum speedup.

As we learned from Chapter 5, to specify the architecture of the neural network is not sufficient to build the working ML model – it is also necessary to specify the training algorithm. In the following sections, we show how a quantum neural network can be trained with the help of differentiable and non-differentiable learning methods.

8.2 Training QNN with Gradient Descent

Since we are not only interested in building QNNs as standalone QML tools but also in comparing and contrasting them with classical neural networks, we start our review of QNN training methods with gradient descent – a ubiquitous classical ML algorithm.

8.2.1 The finite difference scheme

Training QNNs consists of specifying and executing a procedure that finds an optimal configuration of the adjustable rotation parameters θ. Assume that a QNN is specified on n quantum registers with l layers of adjustable quantum gates, where each adjustable gate is controlled by a single parameter $(\theta_i^j)_{i=1,\ldots,n;\, j=1,\ldots,l}$. In this case, $\theta \in \mathcal{M}_{n,l}$ is an $n \times l$ matrix of adjustable network parameters:

$$\theta = \begin{bmatrix} \theta_1^1 & \ldots & \theta_1^l \\ \vdots & \ddots & \vdots \\ \theta_n^1 & \ldots & \theta_n^l \end{bmatrix}.$$

Without loss of generality, we assume that we work with a binary classifier. The latter takes an input (a quantum state that encodes a sample from the dataset), applies a sequence of quantum gates (the parameterised quantum circuit controlled by at most $n \times l$ adjustable parameters), and performs the measurement of an observable M on the chosen quantum register. An example of an observable is the Pauli Z gate and the result of a single measurement is ±1 for a qubit found in the state $|0\rangle$ or $|1\rangle$, respectively. The value of the measured observable is mapped into a value of a binary variable {0, 1}. This process is repeated N times for each sample in order to collect sufficient statistics for the classification result.

The first step in finding an optimal configuration of adjustable parameters θ is to choose an appropriate cost function – an objective function that represents the total error in classifying samples from the training dataset and which can be minimised by changing the adjustable network parameters. Let y := $(y_1,\ldots,y_K)$ be a vector of binary labels and f(θ) := $(f_1(\theta),\ldots,f_K(\theta))$ a vector of binary classifier predictions for the training dataset consisting of K samples. The cost function $L(\theta)$ can then be defined, for example, as the sum of squared errors across all samples in the training dataset:

$$L(\theta) := \frac{1}{2}\sum_{k=1}^{K}\left(y_k - f_k(\theta)\right)^2 .$$

The next step is an iterative update of the adjustable parameters in the direction that reduces the value of the cost function. That direction is given by the cost function gradient – hence the name of the method. The parameters are updated towards the direction of the steepest descent of the cost function. At step $u + 1$, we update the system to

$$_{u+1}\theta_i^j \longleftarrow {}_u\theta_i^j - \eta\frac{\partial L(\theta)}{\partial \theta_i^j}, \quad \text{for each } i = 1,\ldots,n,\ j = 1,\ldots,l,$$

where η is the learning rate, namely a hyperparameter controlling the magnitude of the update. For each $i = 1,\ldots,n$, $j = 1,\ldots,l$, the derivative can be calculated numerically using a finite difference scheme:

$$\frac{\partial L(\theta)}{\partial \theta_i^j} \approx \frac{L(\theta_1^1,\ldots,\theta_i^j+\Delta\theta_i^j,\ldots,\theta_n^l) - L(\theta_1^1,\ldots,\theta_i^j-\Delta\theta_i^j,\ldots,\theta_n^l)}{2\Delta\theta_i^j},$$

with an error of order $\mathcal{O}((\Delta\theta_i^j)^2)$, where $\Delta\theta_i^j$ is a small rotation angle increment. The physical characteristics of the NISQ devices put restrictions on how small this increment can be: in most cases $\Delta\theta_i^j$ should not be smaller than 0.1 radians. The rest of the training routine follows the standard classical algorithm of training neural networks through the backpropagation of error with gradient descent.

8.2.2 The analytic gradient approach

An alternative to the finite difference method, which can be unstable and ill-conditioned due to truncation and round-off errors (for parameterised quantum circuits [29] or, in fact, for classical neural networks [27]), is the analytic gradient approach. It can be a viable choice for parameterised quantum circuits with adjustable one-qubit gates and fixed multi-qubit gates. From (8.2.1), the cost function gradient with respect to the parameter θ_i^j is given by

$$\frac{\partial L(\theta)}{\partial \theta_i^j} = -\sum_{k=1}^{K} (y_k - f_k(\theta)) \frac{\partial f_k(\theta)}{\partial \theta_i^j},$$

so that the task of calculating the gradient of the cost function is reduced to the task of calculating the partial derivative of the expected value of the measurement operator for each sample quantum state that encodes the classical sample from the training dataset. Let $|\psi_k\rangle$ be the quantum state that encodes the k-th sample from the training dataset and let $U(\theta)$ denote the unitary operator that represents the sequence of

QNN gates transforming the initial state $|\psi_k\rangle$. Then the expected value of the measurement operator M is given by

$$f_k(\theta) = \langle\psi_k|\, \mathtt{U}^\dagger(\theta)\mathtt{M}\mathtt{U}(\theta)\, |\psi_k\rangle\,.$$

According to the conventions we used in constructing the QNN ansatz, the parameter θ_i^j only affects a single gate, which we will denote as $G(\theta_i^j)$. Therefore, the sequence of gates $U(\theta)$ can be represented as

$$\mathtt{U}(\theta) = \mathtt{V}\mathtt{G}(\theta_i^j)\mathtt{W},$$

where W and V are gate sequences that precede and follow gate $G(\theta_i^j)$. Let us absorb V into the Hermitian observable $Q = V^\dagger M V$ and W into the quantum state $|\phi_k\rangle = W|\psi_k\rangle$:

$$f_k(\theta) = \langle\phi_k|\, \mathtt{G}^\dagger(\theta_i^j)\mathtt{Q}\mathtt{G}(\theta_i^j)\, |\phi_k\rangle\,.$$

Then the partial derivative of $f_k(\theta)$ with respect to parameter θ_i^j is calculated as

$\frac{\partial f_k(\theta)}{\partial \theta_i^j}$	$= \frac{\partial}{\partial \theta_i^j} \langle\varphi_k\| G^\dagger(\theta_i^j) Q G(\theta_i^j) \|\phi_k\rangle$		
	$= \langle\varphi_k\| \left(\frac{\partial G(\theta_i^j)}{\partial \theta_i^j}\right)^\dagger Q G(\theta_i^j) \|\phi_k\rangle + \langle\varphi_k\| G^\dagger(\theta_i^j) Q \left(\frac{\partial G(\theta_i^j)}{\partial \theta_i^j}\right) \|\phi_k\rangle.$	(8.2.1)	

Let us denote

$$\mathtt{B} := \mathtt{G}(\theta_i^j) \quad \text{and} \quad \mathtt{C} := \frac{\partial \mathtt{G}(\theta_i^j)}{\partial \theta_i^j},$$

and notice that

$$\langle\phi_k|\,\mathtt{C}^\dagger\mathtt{Q}\mathtt{B}\,|\phi_k\rangle + \langle\phi_k|\,\mathtt{B}^\dagger\mathtt{Q}\mathtt{C}\,|\phi_k\rangle$$
$$= \frac{1}{2}\left(\langle\phi_k|\,(\mathtt{B}+\mathtt{C})^\dagger\mathtt{Q}(\mathtt{B}+\mathtt{C})\,|\phi_k\rangle - \langle\phi_k|\,(\mathtt{B}-\mathtt{C})^\dagger\mathtt{Q}(\mathtt{B}-\mathtt{C})\,|\phi_k\rangle\right).$$

Therefore, if we can find the way to implement the operator B±C as part of an overall unitary evolution then we can evaluate (8.2.1) directly.

8.2.3 The parameter shift rule for analytic gradient calculation

Following [257], we outline the parameter shift rule for gates with generators with two distinct eigenvalues – this covers all one-qubit gates. Being unitary, the gate $G(\theta_i^j)$ above can be represented as

$$\mathtt{G}(\theta_i^j) = \exp\left(-\mathrm{i}\theta_i^j\Gamma\right),$$

for some Hermitian operator Γ (Theorem 6). The partial derivative with respect to θ_i^j reads

$$\frac{\partial\mathtt{G}(\theta_i^j)}{\partial\theta_i^j} = -\mathrm{i}\Gamma\exp\left(-\mathrm{i}\theta_i^j\Gamma\right) = -\mathrm{i}\Gamma\mathtt{G}(\theta_i^j).$$

Substituting (8.2.3) into (8.2.1) yields

$$\frac{\partial f_k(\theta)}{\partial\theta_i^j} = \langle\phi_k'|\,\mathrm{i}\Gamma\mathtt{Q}\,|\phi_k'\rangle + \langle\phi_k'|\,\mathtt{Q}(-\mathrm{i}\Gamma)\,|\phi_k'\rangle,$$

where $|\phi_k'\rangle = G(\theta_i^j)|\phi_k\rangle$. If Γ has just two distinct eigenvalues we can shift the eigenvalues to ±*r*, since the global phase is unobservable [257]. With I denoting the identity operator we can rewrite (8.2.3) as

$$\frac{\partial f_k(\theta)}{\partial \theta_i^j} = r\left(\langle \phi_k'| \frac{\mathrm{i}\Gamma}{r} \mathtt{QI} |\phi_k'\rangle - \langle \phi_k'| \mathtt{IQ} \frac{\mathrm{i}\Gamma}{r} |\phi_k'\rangle\right).$$

Denoting

$$\mathtt{B} := \mathtt{I} \quad \text{and} \quad \mathtt{C} := -\frac{\mathrm{i}}{r}\Gamma,$$

and using (8.2.2) we obtain from (8.2.3):

$$\frac{\partial f_k(\theta)}{\partial \theta_i^j} = \frac{r}{2}\left[\langle \phi_k'| \left(\mathtt{I} - \frac{\mathrm{i}}{r}\Gamma\right)^\dagger \mathtt{Q} \left(\mathtt{I} - \frac{\mathrm{i}}{r}\Gamma\right) |\phi_k'\rangle - \langle \phi_k'| \left(\mathtt{I} + \frac{\mathrm{i}}{r}\Gamma\right)^\dagger \mathtt{Q} \left(\mathtt{I} + \frac{\mathrm{i}}{r}\Gamma\right) |\phi_k'\rangle\right].$$

A straightforward computation [257, Theorem 1] shows that if the Hermitian generator Γ of the unitary operator $G(\theta) = \exp(-\mathrm{i}\theta\Gamma)$ has at most two unique eigenvalues $\pm r$, then

$$\mathtt{G}\left(\mp\frac{\pi}{4r}\right) = \frac{1}{\sqrt{2}}\left(\mathtt{I} \pm \frac{\mathrm{i}}{r}\Gamma\right).$$

In this case the gradient can be estimated using two additional evaluations of the quantum circuit. Either the gate $G(\pi/(4r))$ or the gate $G(-\pi/(4r))$ should be placed in the original circuit next to the gate we are differentiating. Since for unitarily generated one-parameter gates $G(a)G(b) = G(a + b)$, this is equivalent to shifting the gate parameter, and we obtain the "parameter shift rule" [257] with the shift $s = \pi/(4r)$:

$$\frac{\partial f_k(\theta)}{\partial \theta_i^j} = r\left(\langle \phi_k| \mathtt{G}^\dagger(\theta_i^j + s)\mathtt{QG}(\theta_i^j + s) |\phi_k\rangle - \langle \phi_k| \mathtt{G}^\dagger(\theta_i^j - s)\mathtt{QG}(\theta_i^j - s) |\phi_k\rangle\right).$$

If Γ is a one-qubit rotation generator given by Pauli X, Y, and Z operators, then $r = 1/2$ and $s = \pi/2$ [213, 257]:

$$\frac{\partial f_k(\theta)}{\partial \theta_i^j} = \frac{1}{2}\Big(\langle \phi_k | \, \mathtt{G}^\dagger \left(\theta_i^j + \frac{\pi}{2}\right) \mathtt{QG} \left(\theta_i^j + \frac{\pi}{2}\right) | \phi_k \rangle - \langle \phi_k | \, \mathtt{G}^\dagger \left(\theta_i^j - \frac{\pi}{2}\right) \mathtt{QG} \left(\theta_i^j - \frac{\pi}{2}\right) | \phi_k \rangle \Big).$$

Therefore, what we need to do in order to estimate the gradient is to execute two circuits N times to collect statistics and to calculate the expectations on the right-hand side of (8.2.3). The first circuit will have the gate parameter shifted by $\pi/2$ and the second circuit will have the gate parameter shifted by $-\pi/2$.

Although this procedure is not necessarily faster than the finite difference scheme, it can produce a more accurate estimate of the cost function gradient. The main argument here is the fact that the NISQ hardware operates with limited precision. The state-of-the-art superconducting qubits have one-qubit gate fidelity ≤ 99.9% and two-qubit gate fidelity ≤ 99.7% with rotation angle precision of order 0.05 radians. Therefore, the finite difference scheme cannot assume infinitesimal rotation angles $\Delta\theta$ – they should not be smaller than about 0.1 radians (and, probably, materially larger in most cases). This means that gradients obtained with the finite difference scheme have some degree of built-in uncertainty that can only be fixed with further improvements in the NISQ hardware.



QNNs can be trained with the gradient descent algorithm in full analogy with the backpropagation of error in classical neural networks. The gradients can be either calculated analytically or estimated numerically.

8.3 Training QNN with Particle Swarm Optimisation

Having specified the gradient descent scheme for training QNNs in the previous section, we now turn our attention to a non-differentiable learning method based on the powerful evolutionary search algorithm.

8.3.1 The Particle Swarm Optimisation algorithm

The Particle Swarm Optimisation (PSO) algorithm belongs to a wide class of evolutionary search heuristics where at each algorithm iteration ("generation" in the language of evolutionary algorithms), the population of solutions ("chromosomes" or "particles") is evaluated in terms of their fitness with respect to the environment. In the standard PSO formulation [236], a number of particles are placed in the solution space of some problem and each evaluates the fitness at its current location. Each particle then determines its movement through the solution space by combining some aspects of the history of its own fitness values with those of one or more members of the swarm, and then moves through the solution space with a velocity determined by the locations and processed fitness values of those other members, along with some random perturbations.

It is a standard procedure [127, 172] to follow three steps in specifying the PSO algorithm. First, we initialise the positions $\mathrm{x}_k^{\,i} := (x_k^{\,i}(1), \ldots, x_k^{\,i}(n)) \in \mathbb{R}^n$ of each particle i at time k moving through the n-dimensional search space and taking values in some range $[\mathrm{x}_{min}, \mathrm{x}_{max}]$. Next we initialise the velocities $\mathrm{v}_k^{\,i} := (v_k^{\,i}(1), \ldots, v_k^{\,i}(n)) \in \mathbb{R}^n$ of each particle in the swarm. The initialisation process consists of distributing swarm particles randomly across the solution space:

$$\mathrm{x}_0^i = \mathrm{x}_{\min} + \omega_x \left(\mathrm{x}_{\max} - \mathrm{x}_{\min}\right), \quad \mathrm{v}_0^i = \frac{\mathrm{x}_{\min} + \omega_v \left(\mathrm{x}_{\max} - \mathrm{x}_{\min}\right)}{\Delta t},$$

where ω_x and ω_v are uniformly distributed random variables on [0,1] and Δt is the time step between algorithm iterations.

We then update the velocities of all particles at time $k+1$ according to the specified objective function which depends on the particles' current positions in the solution space at time k. The value of the objective function determines which particle has the best position p_k^{global} in the current swarm and also determines the best position p^i of each particle over time, i.e., in the current and all previous moves. The velocity update formula uses these two pieces of information for each particle in the swarm along with the effect of the current motion v_k^i to provide a search direction p_{k+1}^i for the next iteration. The velocity update formula includes random parameters to ensure good coverage of the solution space and to avoid entrapment in local optima. The three values that affect the new search direction are the current motion, the particle's own memory, and the swarm influence. They are incorporated via a summation approach with three weight factors: inertia w, self-confidence c_1, and swarm confidence c_2:

$$v_{k+1}^i = w v_k^i + c_1 \omega_1 \frac{\left(p^i - x_k^i\right)}{\Delta t} + c_2 \omega_2 \frac{\left(p_k^{global} - x_k^i\right)}{\Delta t},$$

where ω_1 and ω_2 are uniformly distributed random variables on [0,1].

Finally, the position of each particle is updated using its velocity vector:

$$x_{k+1}^i = x_k^i + v_{k+1}^i \Delta t.$$

These steps are repeated until either a desired convergence criterion is met or until we reach the maximum number of iterations. Various reflection rules (stopping at the boundary, mirror reflection back into the allowed domain, etc.) [190] can be designed for the new position x_{k+1}^i falling outside the $[x_{min}, x_{max}]$ bounds and the dynamics can be normalised with $\Delta t \equiv 1$. If K is the last iteration of the algorithm, then the best solution found by the PSO is p_K^{global}. Figure 8.2 provides a schematic illustration of the particle movement through the solution

space under the influence of three forces: momentum, attraction to the globally best solution found by all particles at the previous iteration, and attraction to the best solution found by the given particle across all previous iterations.

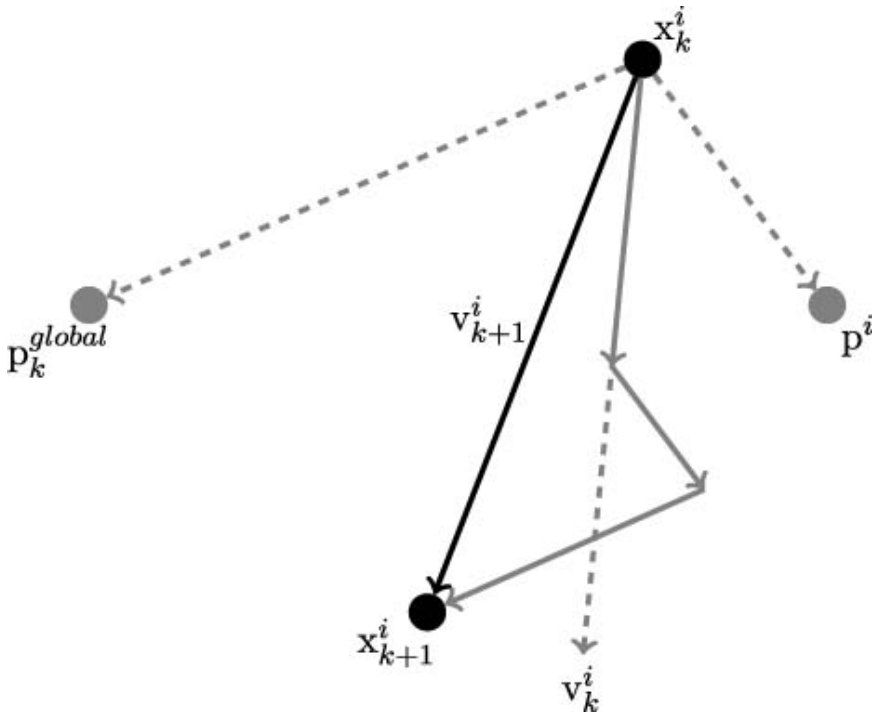


Figure 8.2: Schematic illustration of the PSO algorithm. Each particle moves through the solution space under the influence of three forces: momentum, own memory, and swarm influence.

8.3.2 PSO algorithm for training quantum neural networks

We are now ready to specify the PSO algorithm to train QNNs. We consider the most general case of an $n \times l$ matrix of adjustable parameters (rotations) θ, where n is the number of quantum registers and l is the number of network layers. The solution we look for is the matrix (8.2.1) of adjustable parameters that minimises the chosen cost function.

The cost function can be specified in many different ways depending on what particular aspects we want to encourage or penalise. Given the

training dataset, we would like to find a configuration of adjustable parameters θ such that as many samples as possible are classified correctly. One possible choice of cost function, for example, may be the ratio of incorrect to correct classification decisions. However, the classification process is probabilistic in nature – we decide on the sample label after many runs of the quantum circuit, which generate sufficient statistics. Therefore, each classification decision is not just right or wrong but can be seen as "more right" or "more wrong". If the correct sample label is "1" and we get "0" 51% of the time then the classifier is slightly wrong: the chances are that similar samples would be classified correctly or only a small change to the adjustable network parameters is required to rectify the classification process. But if we get "0", say, 90% of the time, then the classifier is "very wrong" and we need to penalise the outcome more aggressively.

One possible realisation of the cost function that takes into account the above argument is as follows: Without loss of generality, assume that we work with the binary class labels "0" and "1", and let $y := (y_1,\ldots,y_K)$ be a vector of sample labels (either "0" or "1") from the training dataset. Further, let $\mathbb{P}(\theta) := (\mathbb{P}_1(\theta),\ldots,\mathbb{P}_K(\theta))$ be a vector of QNN estimated probabilities of predicting class "1" for the given sample (i.e., the number of quantum circuit runs that returned "1" after measurement divided by the total number of quantum circuit runs). Then the cost function $L(\theta)$ is given by the following pseudo code:

```
cost_function = 0

for i in range(K):

    if y[i] == 0:

        if P[i] > 0.7:

            cost_function += 4
```

```
        if P[i] > 0.6 and P[i] <= 0.7:
            cost_function += 2
        if P[i] > 0.5 and P[i] <= 0.6:
            cost_function += 1
    else:
        if P[i] < 0.5 and P[i] >= 0.4:
            cost_function += 1
        if P[i] < 0.4 and P[i] >= 0.3:
            cost_function += 2
        if P[i] < 0.3:
            cost_function += 4
```

This cost function penalises large errors in the class probability estimate more than small errors and represents the total error across all samples in the training dataset.

We can now formulate the QNN training algorithm, which has the following inputs:

Variable	Meaning
$X := (X_1,\ldots,X_K) \in \mathbb{R}^{M\times K}$	training dataset of features encoded as rotation angles on $[0,\pi]$

$\mathrm{y} := (y_1,\ldots,y_K) \in \{0,1\}^K$	vector of binary labels
N_{iter}	number of iterations
N_{runs}	number of quantum circuit runs
M	number of particles (solutions)
w	momentum coefficient
c_1	particle memory coefficient
c_2	swarm influence coefficient
n	number of quantum registers
l	number of QNN layers

Table 8.2: Inputs of the QNN training algorithm

The algorithm operates on the following objects, where $m = 1,\ldots,M$ denotes the m-th particle, and $t = 0,\ldots,N_{\text{iter}}$ represents the algorithm iteration step:

- $\theta(t;m) \in \mathcal{M}_{\text{nl}}([-\pi,\pi])$: position of particle m at time t;
- $\mathrm{v}(t;m) \in \mathcal{M}_{\text{nl}}([-\pi,\pi])$: velocity of particle m at time t;
- $\Xi(m) \in \mathcal{M}_{\text{nl}}([-\pi,\pi])$: best position found by particle m across all iterations;
- $\Phi(t) \in \mathcal{M}_{\text{nl}}([-\pi,\pi])$: the globally best position found by all particles at time t;
- $L(\theta)$: value of the cost function for the solution θ.

Algorithm 5: Particle Swarm Optimisation

Result: Optimal configuration of adjustable QNN parameters
$\theta^* := \operatorname{argmin} L(\theta)$.

Initialisation and evaluation of the first set of solutions
(*we set* Δt *in* (8.3.1) *equal to* 1):

for *each particle* $m = 1, \ldots, M$ **do**
  for $i = 1, \ldots, n$, $j = 1, \ldots, l$ **do**
    Randomly draw the rotation angle $\theta_i^j(0; m)$ from $\mathcal{U}([-\pi, \pi])$.
    Randomly draw the rotation angle $v_i^j(0; m)$ from $\mathcal{U}([-\pi, \pi])$.
  end

  Initialise the individually best solution:

  $\Xi(m) \leftarrow \theta(0; m)$

  for $k = 1, \ldots, K$ **do**
    Run the quantum circuit N_{runs} times with configuration $\theta(0; m)$ on sample X_k to estimate the probability $\mathbb{P}_k$ of reading out "1" on the target qubit.
  end

  Evaluate the cost function $L(\theta(0; m))$ given the probabilities $\mathbb{P} := (\mathbb{P}_1, \ldots, \mathbb{P}_K)$.
end

Order solutions from best (minimal cost function) to worst (maximal cost function).

$\Phi(0) \leftarrow$ configuration corresponding to the minimum of the cost function.

Initialise the optimal configuration:

$\theta^* \leftarrow \Phi(0)$

Iterations:

for $t = 1, \ldots, N_{iter}$ **do**

  for $m = 1, \ldots, M$ **do**

    for $i = 1, \ldots, n,\ j = 1, \ldots, l$ **do**

      Generate independent random numbers $\omega_1 \sim U[0,1]$ and $\omega_2 \sim U[0,1]$.

      momentum $\leftarrow w v_i^j(t-1;m)$

      particle $\leftarrow c_1 \omega_1 [\Xi_i^j(m) - \theta_i^j(t-1;m)]$

      swarm $\leftarrow c_2 \omega_2 [\Phi_i^j(t-1) - \theta_i^j(t-1;m)]$

      $v_i^j(t;m) \leftarrow$ momentum + particle + swarm

      $\theta_i^j(t;m) \leftarrow \theta_i^j(t-1;m) + v_i^j(t;m)$

    end

    for $k = 1, \ldots, K$ **do**

      Run the quantum circuit N_{runs} times with configuration $\theta(t;m)$ on sample X_k to estimate the probability $\mathbb{P}_k$ of reading out "1" on the target qubit.

    end

    Evaluate the cost function $L(\theta(t;m))$ given $\mathbb{P} := (\mathbb{P}_1, \ldots, \mathbb{P}_K)$.

    if $L(\theta(t;m)) < L(\Xi(m))$ **then**

      $\Xi(m) \leftarrow \theta(t;m)$

    end

  end

  Order solutions from best (minimum value of the cost function) to worst (maximum value of the cost function).

  $\Phi(t) \leftarrow$ configuration corresponding to the minimum of the cost function.

  if $L(\theta^*) < L(\Phi(t))$ **then**

    $\theta^* \leftarrow \Phi(t)$

  end

end



The non-differentiable learning based on the evolutionary search heuristic works well for irregular, non-convex objective functions with many local minima.

8.4 QNN Embedding on NISQ QPU

Ideally, parameterised quantum circuits should be constructed in a hardware-agnostic way, only driven by the characteristics of the problem being solved. This, however, would require the existence of large and exceptionally well-connected quantum computing systems with very high qubit fidelity and coherence time. In other words, we would need QPUs with capabilities that significantly exceed those of existing NISQ devices. The time for such powerful quantum computing systems may come sooner than one may expect but we still have to find a way of running PQCs efficiently on NISQ QPUs.

8.4.1 NISQ QPU connectivity

A typical approach to designing a PQC executable on the NISQ QPU would start with observing two main characteristics of the quantum computing systems: the graph (qubit connectivity) and the set of native gates. We can illustrate these points by looking at Rigetti's Aspen system [72] in Figure 8.3.

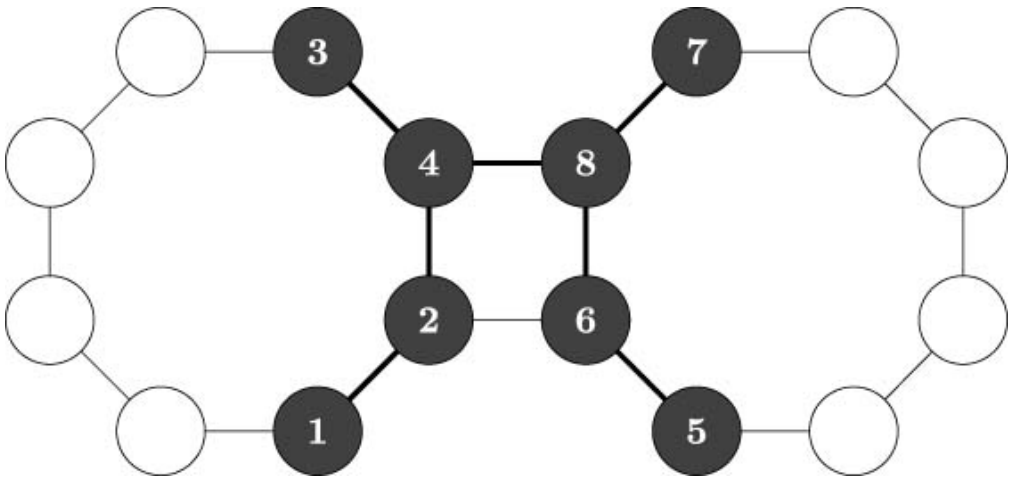


Figure 8.3: Rigetti's Aspen system.

As we can see, most qubits are only connected to their nearest neighbours on the linear grid, with only four qubits having three connections. These extra connections form a bridge between two 8-qubit islands that, otherwise, would be completely independent.

8.4.2 QNN embedding scheme

The shaded qubits in Figure 8.3 can be used to construct the 8-qubit tree network capable of processing a dataset with up to 16 continuous features (two features per quantum register) as shown in Figure 8.4. The thick lines in Figure 8.3 represent qubits connectivity used in constructing the QNN. The thin lines represent all other available qubit connections that have not been utilised in the QNN ansatz.

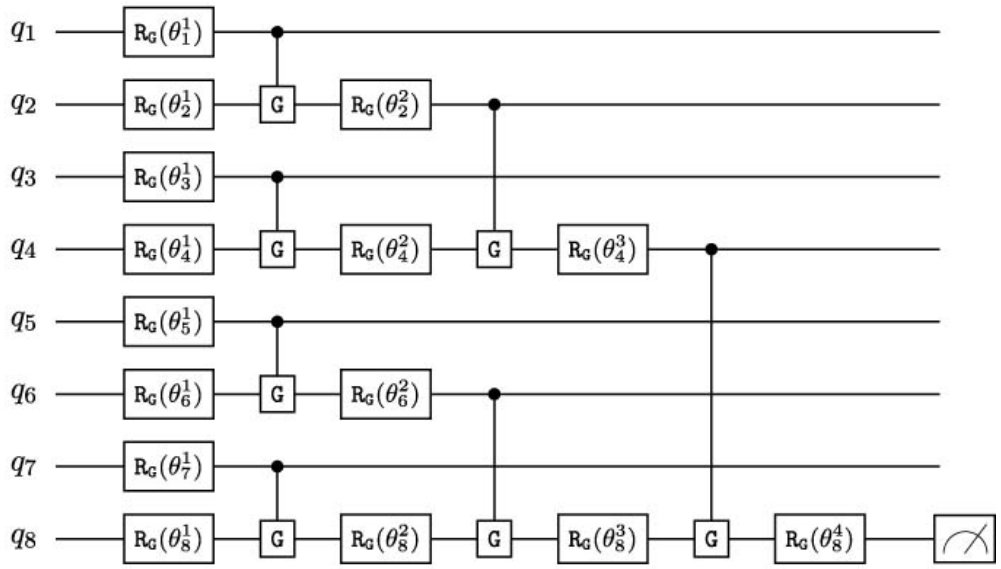


Figure 8.4: QNN for the Aspen system; the gate G is any of the {X,Y,Z} gates.



With the limited connectivity of existing QPUs, we need to fully utilise the graph structure of the quantum chips to implement the most efficient QNN embedding and extract the best possible performance.

8.5 QNN Trained as a Classifier

We now demonstrate how a binary QNN classifier can be trained on a classical credit approval dataset using the non-differentiable learning approach.

8.5.1 The ACA dataset and QNN ansatz

One of the most fundamental use cases for a binary classifier in finance is credit approval. The UCI Machine Learning Database [241, 242] holds the Australian Credit Approval (ACA) dataset consisting of 690 samples. There are 14 features (binary, integer, continuous) representing various attributes of potential borrowers and a binary class label (accept/reject credit application). The dataset is reasonably hard for classical classifiers due to the limited predictive power of the features and its relatively small size. This makes it ideal for testing and benchmarking the QNN performance.

We start with the simplest tree network that can be mapped onto Rigetti's Aspen system graph described in the previous section. Figure 8.5 shows the full quantum circuit consisting of sample encoding and sample processing modules [171]. The proposed scheme allows us to encode up to two continuous features per quantum register with the help of rotations around the x- and the y-axes.

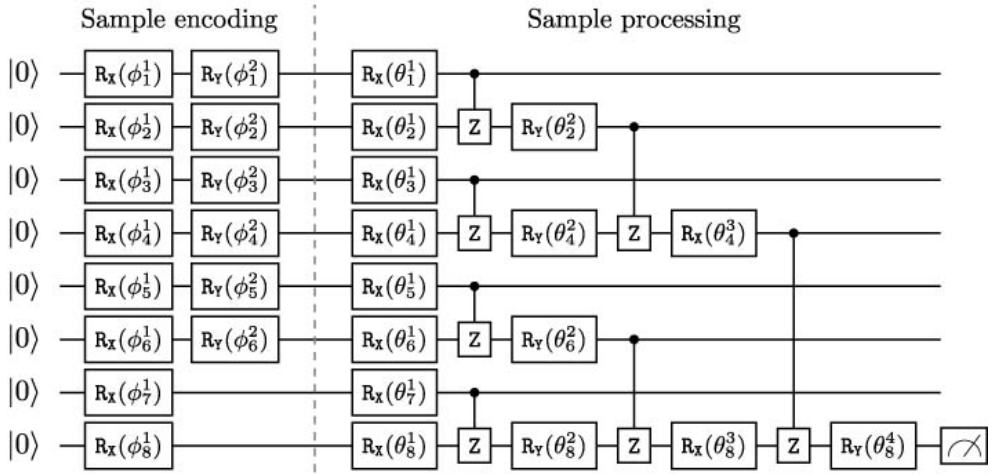


Figure 8.5: PQC for the credit approvals classifier.

The features are encoded as rotation angles $\varphi \in [0,\pi]$ according to the encoding scheme described in Section 7.2. With all qubits initialised as $|0\rangle$ in the computational basis, this ensures the uniqueness of the encoded samples. The sample processing module consists of layers of adjustable one-qubit gates (rotations around the x- and the y-axes) and fixed two-qubit gates (CZ). We split the ACA dataset 50:50 into a training and a testing dataset using the train_test_split() function provided by the `sklearn.model_selection` module. Our objective is to train the QNN and various classical classifiers (classical benchmarks) on the training dataset and compare their out-of-sample performance on the testing dataset. The classical classifiers have a number of hyperparameters that can be fine-tuned to optimise the classifier performance on the given dataset. In contrast, the QNN architecture (location and types of one-qubit and two-qubit gates) is fixed.

8.5.2 Training an ACA classifier with the PSO algorithm

We first verify that the QNN can be efficiently trained with the Particle Swarm Optimisation algorithm – a non-differentiable learning approach. Figure 8.6 illustrates PSO convergence for the set of PSO parameters given in Table 8.3.

Parameter	Notation	Value
Inertia coefficient	w	0.25
Self-confidence coefficient	c_1	0.25
Swarm confidence coefficient	c_2	0.25
Number of particles	M	10
Number of iterations	N_{iter}	20
Number of quantum circuit runs	N_{runs}	1000

Table 8.3: PSO parameters.

The sample algorithm run has reached the minimum of the objective function in just four iterations with only ten particles, exploring the search space using the `Qiskit` quantum simulator.

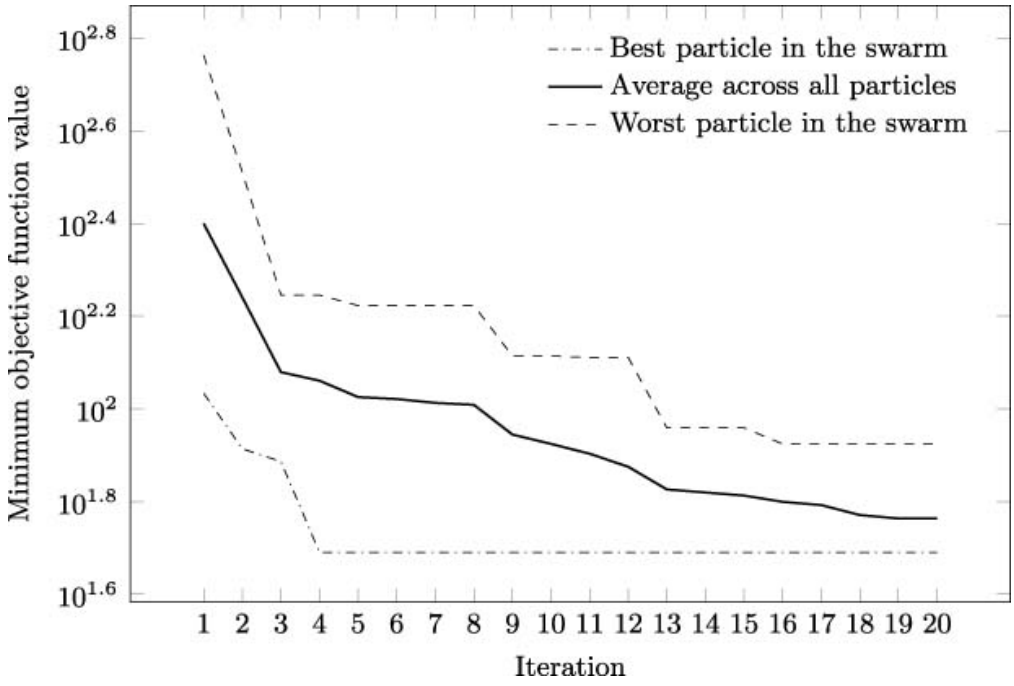


Figure 8.6: Minimum objective function values found by individual particles.

The configuration of adjustable parameters (rotations) that corresponds to the minimum of the objective function found by the PSO algorithm is given by (8.5.2).

$$\theta = \begin{bmatrix} 0.16\pi & & & \\ -0.55\pi & 0.66\pi & & \\ -0.13\pi & & & \\ 0.08\pi & 0.72\pi & 0.02\pi & \\ 0.33\pi & & & \\ 0.06\pi & 0.95\pi & & \\ 0.48\pi & & & \\ 0.19\pi & -0.91\pi & -0.83\pi & 0.59\pi \end{bmatrix}.$$

Figure 8.7 displays the in- and out-of-sample confusion matrices for the QNN classifier obtained with the Qiskit quantum simulator assuming that Class 0 is the positive class.

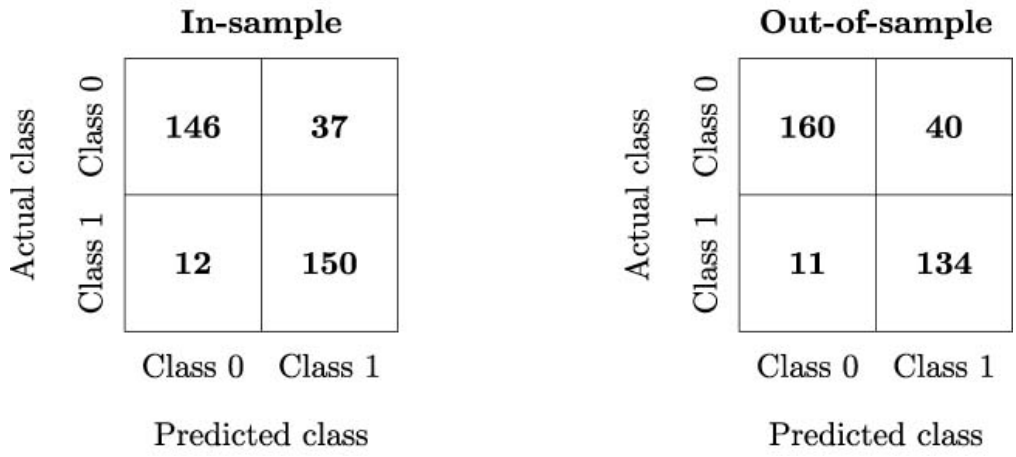


Figure 8.7: Confusion matrix for the QNN classifier (ACA dataset).

The results are robust with an in-sample accuracy of 0.86 and an out-of-sample accuracy of 0.85. Interestingly, the in-sample and out-of-sample results are very close, indicating that the QNN provides strong regularisation. The question of quantum and classical neural networks regularisation will be tackled in Chapter 12.

8.6 Classical Benchmarks

In Chapter 4, we introduced two classical classifiers: a feedforward artificial neural network (Multi-Layer Perceptron) and a decision tree algorithm. We now expand the range of classical benchmark classifiers by adding Support Vector Machine (SVM) [70], Logistic Regression [31], and Random Forest [136]. The SVM approach based on the *kernel method* is covered in Chapter 13. Here, we briefly explain the main principles of logistic regression and random forest classifiers.

8.6.1 Logistic Regression and Random Forest

Logistic regression can be seen as a special case of a feedforward neural network with a single hidden layer consisting of an activation unit with the logistic activation function. The model operates as shown in Figure 4.3 with

$$y(s) = \left(1 + \mathrm{e}^{-s}\right)^{-1}.$$

The standard logistic regression model is a *linear classifier* because the outcome always depends on the sum of the (weighted) inputs. Therefore, logistic regression performs well when working with a dataset where the classes are more or less linearly separable.

Random forest is an *ensemble learning* model and, as the name suggests, is based on combining the classification results of multiple decision trees. The ensemble technique used by random forest is known as *bootstrap aggregation*, or *bagging*, by choosing random subsets from the dataset. Hence, each decision tree is generated from samples drawn from the original dataset with replacement (row sampling). This step of row sampling with replacement is called the *bootstrap*. Each decision tree is trained independently. The final output for the given samples is based on *majority voting* after combining the results of all individual decision trees. This is the *aggregation* step.

8.6.2 Benchmarking against standard classical classifiers

The classical benchmarking can be done by training several popular `scikit-learn` models. Table 8.4 provides classical benchmarking results in terms of out-of-sample F_1 scores for the following (weakly) optimised scikit-learn classifiers:

- a feedforward neural network (MLP) classifier: `neural_network.MLPClassifier`
- a support vector machine classifier: `svm.SVC`

- an ensemble learning model: `ensemble.RandomForestClassifier`
- a logistic regression classifier: `linear_model.LogisticRegression`

The F_1 score is a harmonic average of two performance metrics, precision and recall:

$$F_1 := 2 \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}},$$

both introduced in Chapter 4. In the context of credit approvals, optimising for recall helps with minimising the chance of approving a credit application that should be rejected. However, this comes at the cost of not approving credit applications for some high-quality borrowers. If we optimise for precision, then we improve the overall correctness of our decisions at the cost of approving some applicants with bad credits. The F_1 score is used to balance the positives and negatives in optimising precision and recall.

Classifier	Average F_1 score
Logistic Regression Classifier	0.88
Random Forest Classifier	0.87
MLP Classifier	0.86
QNN Classifier	0.85
Support Vector Classifier	0.84

Table 8.4: Out-of-sample F_1 scores for the classical and QNN classifiers trained on the ACA dataset.

The QNN classifier performance, as measured by the average F_1 score for Class 0 and Class 1, falls somewhere in the middle of the range of out-of-sample F_1 scores for the chosen classical benchmarks. This is encouraging since the QNN ansatz was fixed and we did not optimise the QNN hyperparameters – the placement and types of the two-qubit

gates. The classifier performance can be further improved by deploying the standard ensemble learning techniques, as explained in the following section.



QNNs can be productively used for classification tasks on classical finance-related datasets.

8.7 Improving Performance with Ensemble Learning

The ensemble learning methods combine different weak classifiers into a strong classifier that has better generalisation capabilities than each individual standalone classifier. In Chapter 4, we saw how the principles of ensemble learning can be used in combination with the methods of quantum annealing. Here, we look at them from the QNN perspective.

8.7.1 Majority voting

The popular ensemble learning methods are majority voting (binary classification) and plurality voting (multiclass classification). Majority voting means what it says: the class label for the given sample is the one that receives more than half of the individual votes. Plurality voting chooses the class that receives the largest number of votes (the mode).

The ensemble of the individual classifiers can be built from different classification algorithms. For example, by combining neural network classifiers, support vector machines, decision trees, etc. On the other hand, the same basic classification algorithm can be used to produce multiple classifiers by choosing different configurations of hyperparameters and different subsets of the training dataset. The

random forest classifier, which combines different decision tree classifiers, illustrates the latter approach.

With these considerations in mind, we build a strong classifier from several individual QNN classifiers by changing the QNN ansatz within the restrictions imposed by the QPU qubit connectivity. In order to test the majority voting approach, we build two new QNN classifiers by adding a few more two-qubit CZ gates to the baseline parameterised quantum circuit, as shown in Figures 8.8 and 8.9.

In the case of PQC #2, we add two extra CZ gates, exploiting the "bridge" structure of the Aspen system (Figure 8.3). This improves the overall system entaglement and allows for a richer set of achievable quantum states. PQC #3 has three extra CZ gates in comparison with the baseline circuit. The new classifiers can be trained with the same algorithm (PSO) on the same training dataset but will have different optimal configurations of the adjustable parameters and will make slightly different classification decisions on the testing dataset.

With three QNN classifiers, the majority voting leads to either a unanimous or a 2:1 decision. Performance on the ACA dataset improves marginally with all three classifiers generally in full agreement with each other. There are only a handful of instances where majority voting adds value, but this improves the average out-of-sample F_1 score from 0.85 to 0.87 – on par with the random forest classifier trained on the same dataset.

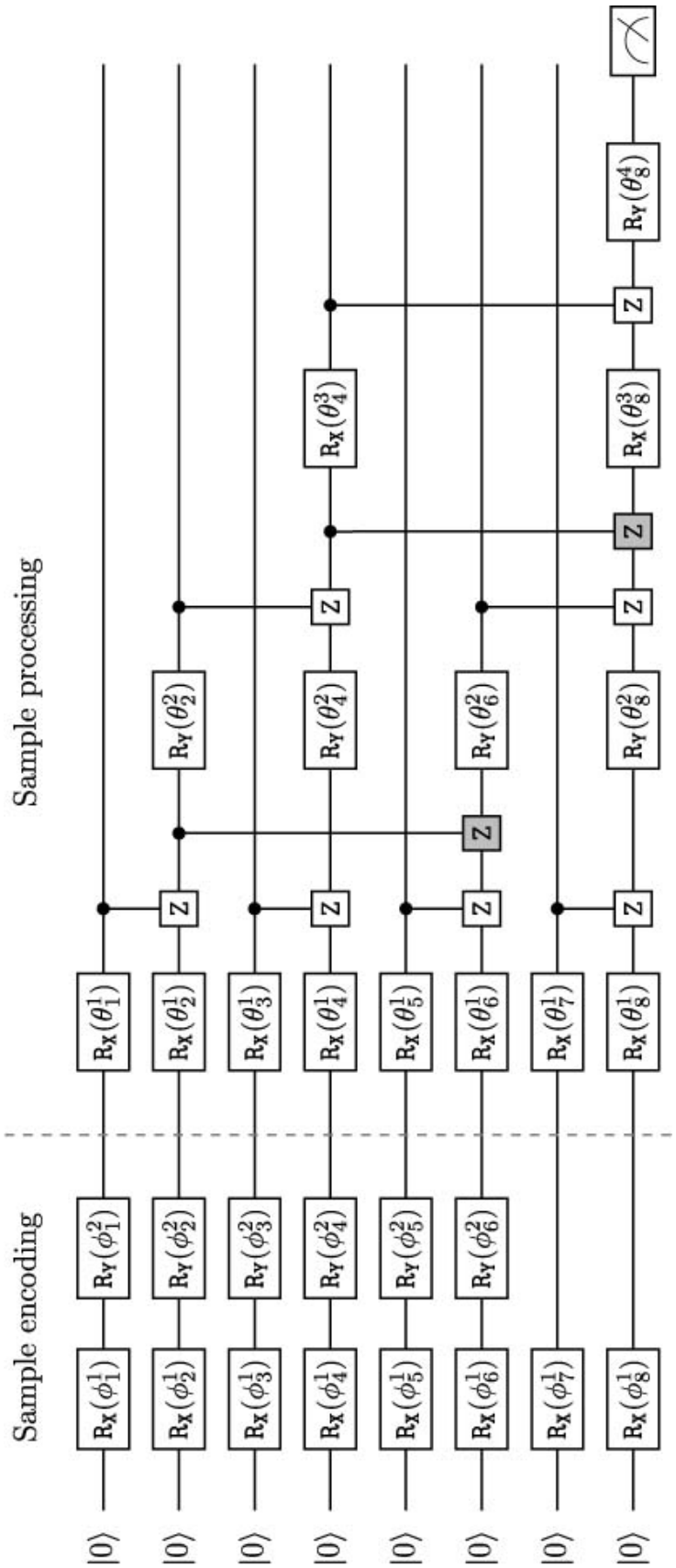


Figure 8.8: PQC #2 for the credit approvals classifier. New fixed 2-qubit gates are shaded grey.

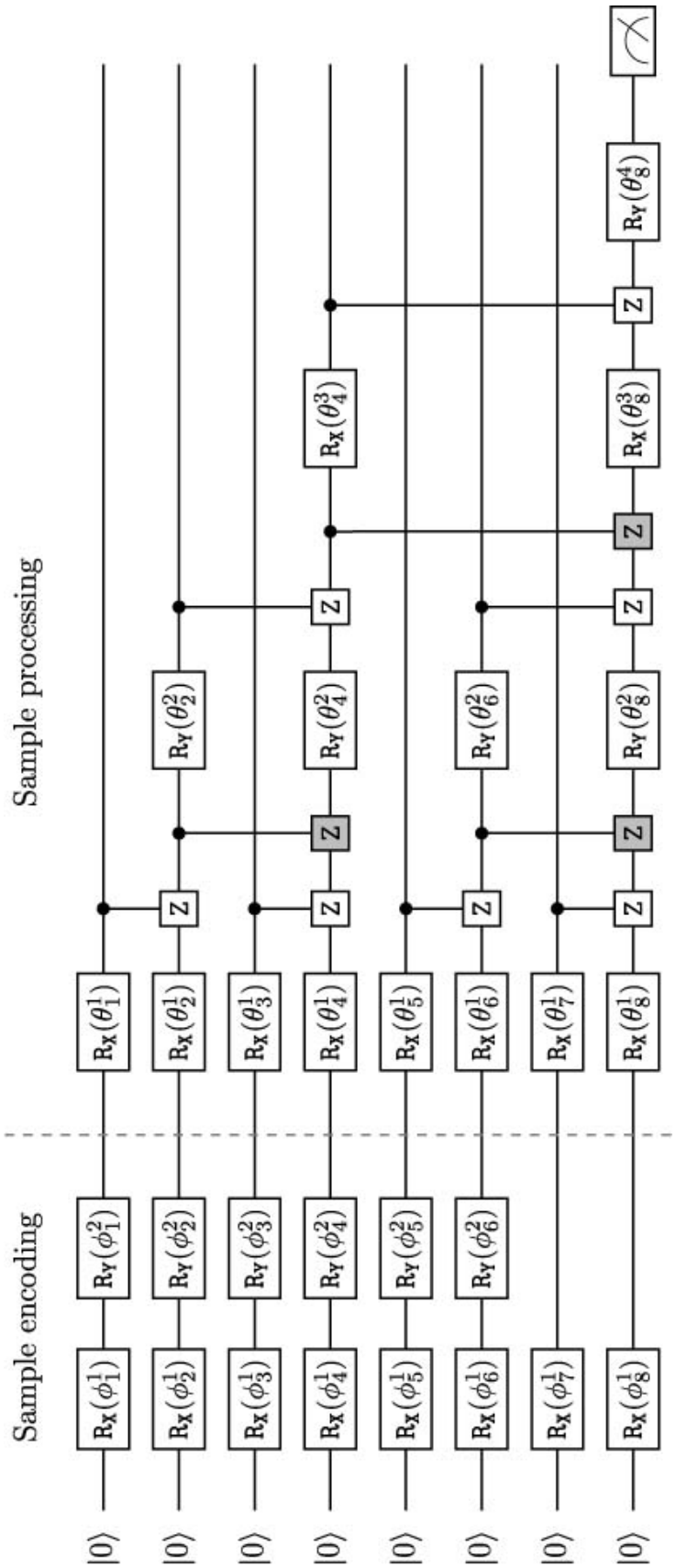


Figure 8.9: PQC #3 for the credit approvals classifier. New fixed 2-qubit gates are shaded grey.

Similar results can be achieved with the original QNN classifier trained on different subsets of the training dataset. These subsets are produced by drawing the bootstrap samples – random samples with replacement – from the original training dataset. The differently trained QNN classifiers can then be combined into a single strong classifier using the majority voting approach as described above.

8.7.2 Quantum boosting

We started by introducing the concept of ensemble learning where predictions produced by various QNNs are combined into a more robust unified prediction via a classical majority voting method. However, we can take a different approach to ensemble learning: predictions of various classical classifiers can be treated as an input into the QNN that performs their aggregation and comes up with a unified prediction. In other words, the QNN operates as a quantum booster similar to the QUBO-based QBoost model introduced in Chapter 4.

Let us come back to the classical benchmarks used in Section 8.5. There are four different machine learning models performing binary classifications. Their outputs ("0" for Class 0 and "1" for Class 1) are inputs into a 4-qubit QNN classifier. Since all quantum registers are initialised as $|0\rangle$, the outputs of individual classifiers can be encoded by either doing nothing for Class 0 output (which is equivalent to applying an identity operator I) or by applying a NOT gate X for Class 1 output.

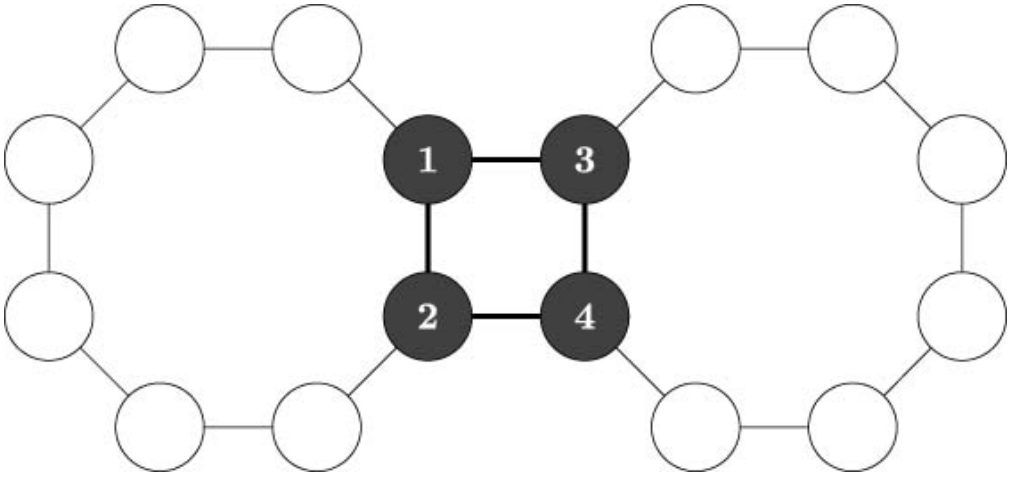


Figure 8.10: Embedding of a 4-qubit QNN onto the bridge section of Rigetti's Aspen system.

Figure 8.10 shows how a 4-qubit QNN can be efficiently embedded on the QPU and Figure 8.11 shows the corresponding parameterised quantum circuit with adjustable one-qubit gates (R_X,R_Y) and fixed two-qubit gates (CZ).

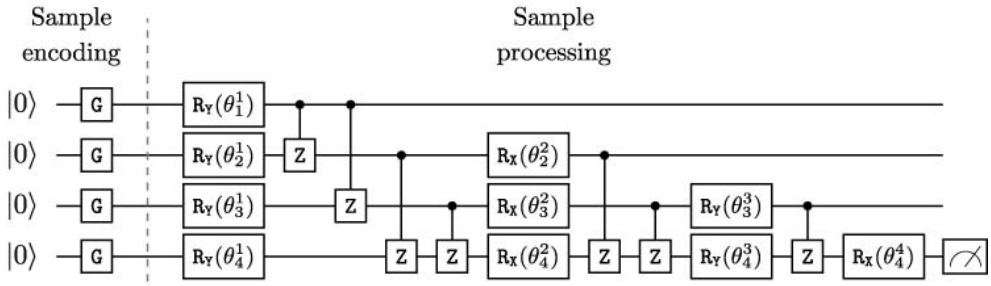


Figure 8.11: QBoost circuit. The sample encoding gate G is either an identity gate I if the input is "0", or a NOT gate X if the input is "1".



Ensemble learning can improve QNN performance in the same way it improves performance of the classical weak learners.

Summary

In this chapter, we introduced the concept of a quantum neural network as a parameterised quantum circuit trained as a classifier. We considered two approaches to training QNNs: differentiable (gradient descent) and non-differentiable (Particle Swarm Optimisation) methods. Gradient descent is generally faster but can face the problem of barren plateaus (vanishing gradients). The evolutionary search heuristics may be slower but can handle the presence of multiple local minima and strike the right balance between exploration and exploitation.

We also explored the embedding of QNNs on the NISQ QPUs with limited connectivity between the qubits. As an example, we considered Rigetti's Aspen system and proposed an efficient embedding scheme that mirrors the "tree structure" architecture of the QNN.

Once our QNN was fully specified and embedded into a QPU graph, we investigated its performance on a real-world dataset of credit approvals and provided comparisons with several standard classical classifiers.

Finally, we looked at several ensemble learning techniques that assist in improving QNN performance in the context of a hybrid quantum-classical protocol.

In the next chapter, we will study a powerful generative QML model – the Quantum Circuit Born Machine – which is a direct quantum counterpart of the classical Restricted Boltzmann Machine we considered in Chapter 5.

Join our book's Discord space

Join our Discord community to meet like-minded people and learn alongside more than 2000 members at: https://packt.link/quantum



9
Quantum Circuit Born Machine

The arrival of the new computational paradigm of quantum computing and the progress achieved in developing quantum computing hardware prompted intensive research in exploring the capabilities of quantum machine learning models and, more specifically, quantum generative models that can be viewed as quantum counterparts of the classical RBMs introduced in Chapter 5. Classical generative models form one of the most important classes of unsupervised machine learning techniques with numerous applications in finance, such as the generation of synthetic market data [48, 173], the development of systematic trading strategies [176] or data anonymisation [174], to name just a few.

Quantum generative models have all the necessary qualities needed to establish quantum advantage on NISQ devices. Probably the most well known example of such models is the Quantum Circuit Born Machine (QCBM), which consists of several layers of adjustable and fixed gates followed by measurement operators. The input is a quantum state where all qubits are initialised as $|0\rangle$ in the computational basis. The output is a bitstring, which is a sample from the probability distribution encoded in the final state constructed by the application of adjustable and fixed gates to the initial state.

The expectation of experimental proof of the quantum advantage is motivated by the following observations: First, QCBMs have strictly larger expressive power than classical RBMs when only a polynomial number of parameters is allowed (the number of qubits in QCBM or the number of visible activation units in RBM) [88]. Second, generating an independent sample from the learned distribution can be done in a single run of the quantum circuit in the case of QCBM – this compares

favorably with up to 10^3-10^4 forward and backward passes through the network in the case of RBM, which are needed to achieve the state of thermal equilibrium [173]. This points towards material quantum speedup. Third, quantum generative models can be used to load data into a quantum state, thus facilitating realisations of many promising quantum algorithms [314].

9.1 Constructing QCBM

As we have seen in Chapter 8, the art of building a QML model that can be run on a NISQ computer consists of finding an optimal PQC architecture that can be embedded into the chosen QPU graph. In this section, we show how it can be done for the QCBM compatible with IBM's Melbourne and Rochester systems.

9.1.1 QCBM architecture

QCBM is a parameterised quantum circuit where a layer of adjustable one-qubit gates is followed by a layer of fixed two-qubit gates. Such a pattern can be repeated any number of times, building a progressively deeper circuit. The input is a quantum state where all qubits are initialised as $|0\rangle$ in the computational basis. The final layer consists of measurement operators producing a bitstring sample from the learned distribution. Therefore, to specify the QCBM architecture means to specify the number of layers, the type of adjustable gates, and the type of fixed gates for each layer. Since the theory of PQC is still being developed [29], we can rely on similarities and analogies between PQCs and classical neural networks to come up with some initial guesses about the possible QCBM architecture.

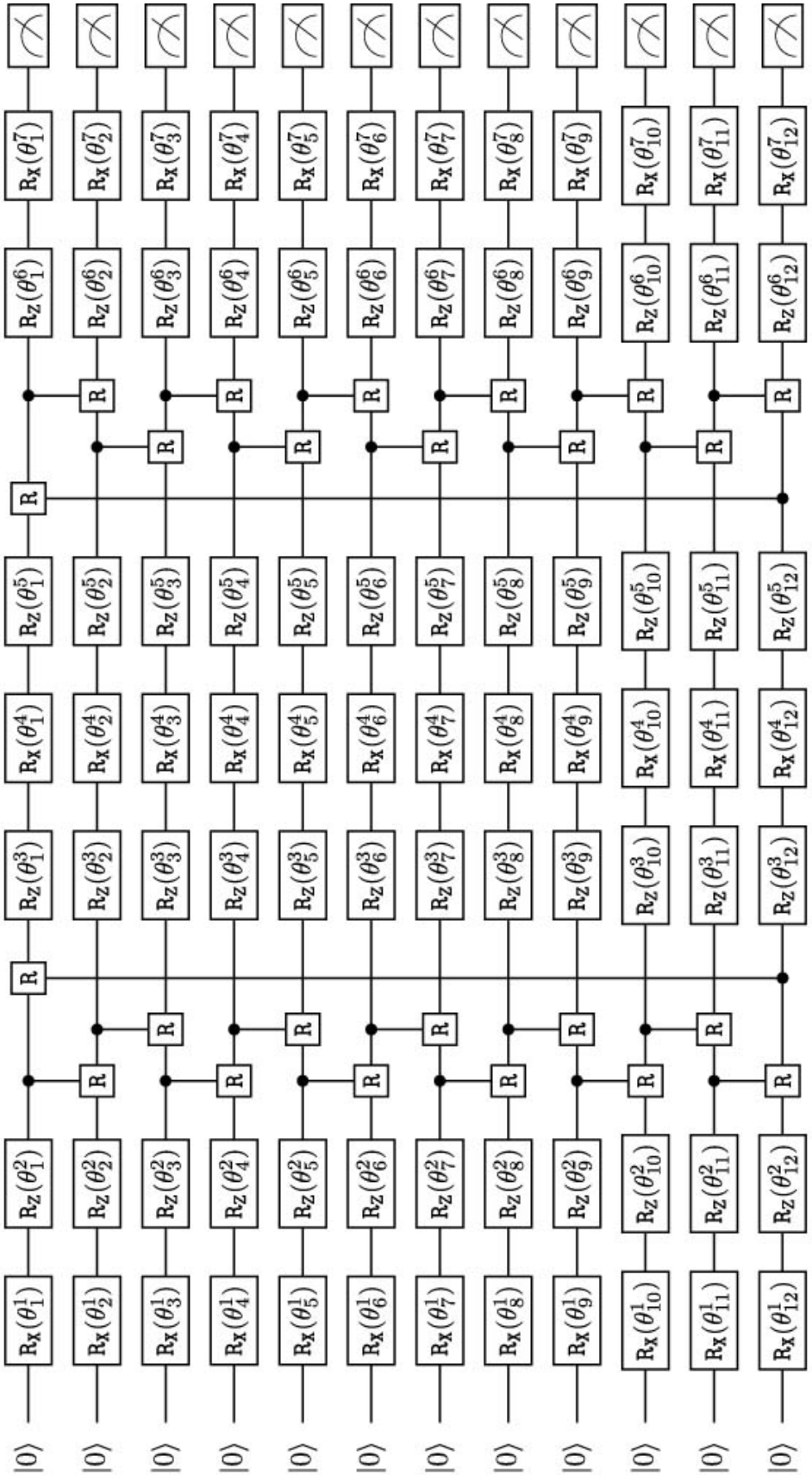


Figure 9.1: QCBM(12, 7).

Figure 9.1 displays a 12-qubit QCBM with two layers of controlled rotation gates $R = R_G(\varphi)$ for $G \in \{X,Y,Z\}$ and $\varphi \in [-\pi,\pi]$, where G and φ are fixed, and three layers of one-qubit gates $R_X(\theta)$ and $R_Z(\theta)$ with a total of seven adjustable gates per quantum register. The circuit is wide enough and deep enough to learn a complex distribution of a continuous random variable while remaining implementable on existing NISQ devices: the 12-digit binary representation of a continuous random variable provides sufficient precision and seven adjustable parameters (rotation angles) per qubit provide sufficient flexibility. At the same time, the circuit is not too deep to be compromised by the gate fidelity achievable in existing quantum hardware [46, 164].

9.1.2 QCBM embedding

The chosen QCBM architecture is compatible with the limited connectivity observed in the current generation of quantum processors. For example, the proposed circuit requires sequential qubit connectivity where qubit n is directly connected with qubits $n-1$ and $n+1$ but does not have to be directly connected with other qubits. This architecture can for example be supported by IBM's Melbourne system [208] in Figure 9.2, where the 12 shaded qubits correspond to the 12 quantum registers in Figure 9.1. The thick lines represent connections used in the QCBM ansatz while the thin lines represent all other available qubit connections.

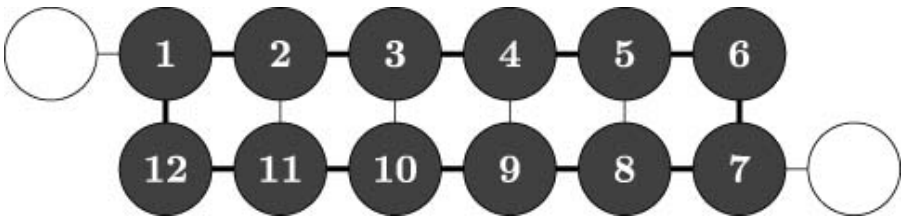


Figure 9.2: IBM's Melbourne system.

The 53-qubit Rochester device [208] in Figure 9.3 can also be used to implement this QCBM architecture. Here, we have several choices for embedding the QCBM circuit (12 linearly connected qubits forming a closed loop); shaded qubits show one such possibility.

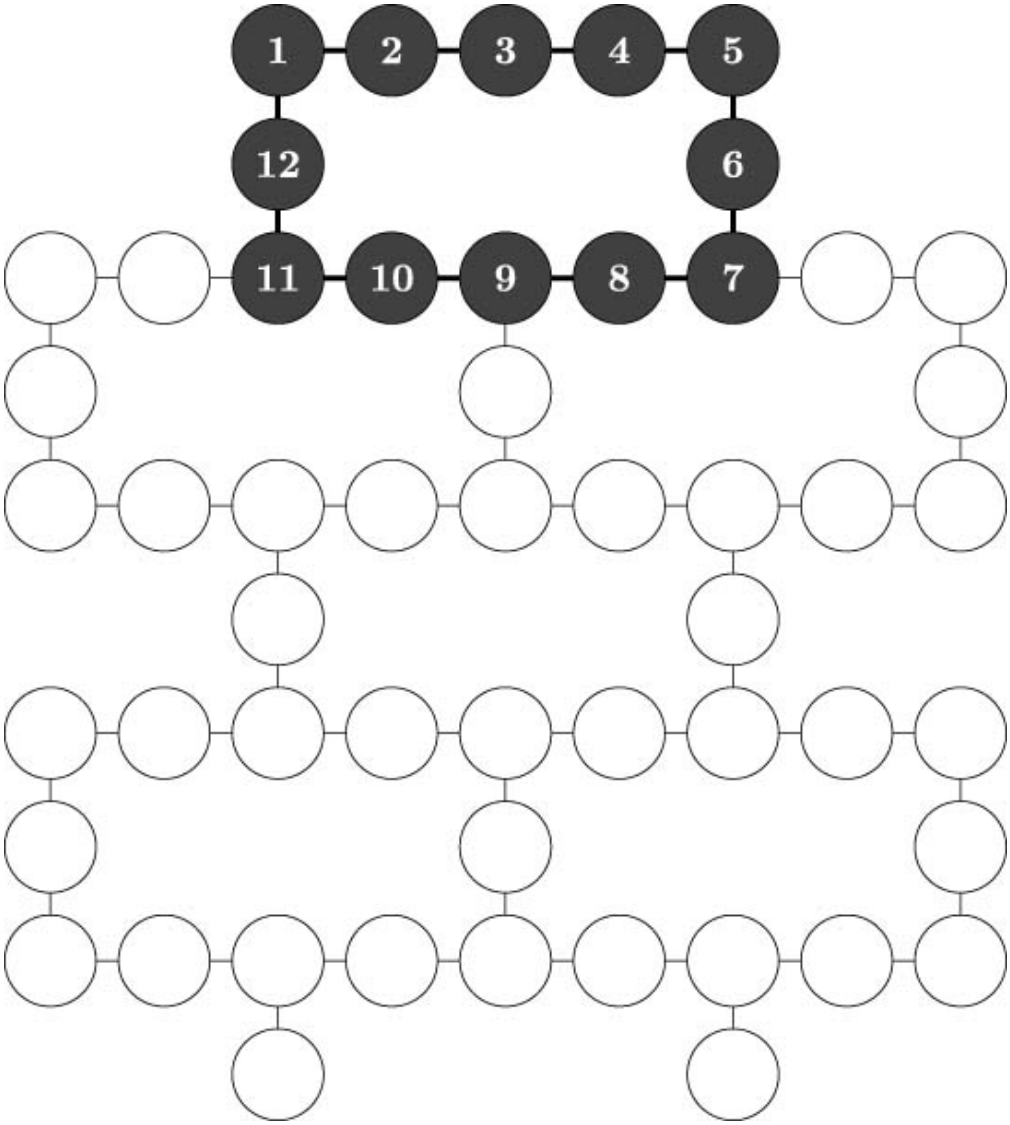


Figure 9.3: IBM's Rochester system.

IBM systems, such as Melbourne and Rochester, are based on superconducting qubits. The choice of the underlying technology means that there is a set of native gates – the quantum gates derived directly from the types of interactions that occur in the given technical realisation of the quantum chip.

In the case of IBM devices, the cross resonance gate generates the ZX interaction that leads to a CNOT gate. When it comes to single-qubit gates, we note that R_Z is a diagonal gate given by (6.3.3) and can be implemented virtually in hardware via frame change (at zero error and duration) [239]. Therefore, it is sufficient to have just an X drive to rotate the qubit on the Bloch sphere (one can move a qubit between two arbitrary points on the Bloch sphere with the help of just two gates, R_X and R_Z).

This means that we can introduce the concept of a *hardware-efficient* architecture not only in terms of connectivity but also in terms of the choice of one-qubit and two-qubit gates. Taking into account the CNOT and CPHASE gate decomposition shown in Figures 6.19 and 6.20, the hardware-efficient QCBM architecture for the Melbourne and Rochester systems would consist of a combination of R_X and R_Z adjustable single-qubit gates and CNOT and CPHASE fixed two-qubit gates [153, 30].



QCBM is a PQC trained as a generative ML model. QCBM operating on N quantum registers transforms the initial quantum state $|0\rangle^{\otimes N}$ into the quantum state encoding the learned probability distribution.

9.2 Differentiable Learning of QCBM

The output of a QCBM circuit is a bitstring that represents a sample from the probability distribution encoded in the quantum state. The circuit itself is, essentially, a mechanism of transforming an initial state $|0\rangle^{\otimes n}$ into a final state from which a sample is generated by means of measuring the qubits in the computational basis.

Different configurations of one-qubit and multi-qubit gates encode different probability distributions – the training of QCBM consists of finding an optimal circuit configuration (ansatz) and an optimal set of adjustable parameters that minimise the distance between the probability distribution encoded in the final quantum state (before measurement, or "before sampling") and the probability distribution of the training dataset.

Following the structure we adopted in Chapter 8, we start with the differentiable learning approach, before moving to the non-differentiable learning method based on a different kind of evolutionary search heuristic – Genetic Algorithm.

9.2.1 Sample encoding

In the most general case, a training dataset consists of samples containing continuous, integer and categorical features. However, QCBM operates on binary variables. Therefore, we need to design a method to convert continuous features into binary ones and a method for converting generated binary QCBM output (sampling) into continuous variables. The integer and binary features can be treated as special cases of continuous features and categorical features can be first converted into binary features through one-hot encoding. Such a method can be realised as a two-step routine (Algorithm 6):

1. Conversion of a continuous variable into the corresponding integer variable.
2. Conversion of the integer variable into the corresponding binary variable.

Given the generated binary output, the same routine can be used in reverse mode to produce continuous samples (Algorithm 7):

1. Conversion of the generated binary QCBM output into integer samples.
2. Conversion of integer samples into the corresponding continuous samples.

Algorithm 6: Continuous to integer to binary transformation (training phase)

Result: Conversion of continuous variables into M-digit binary features.

Input: $\left(X_{\text{real}}^{(n)}(l)\right)_{l=1,\ldots,N_{\text{samples}};\ n=1,\ldots,N_{\text{variables}}}$ – continuous data sample.

for $n = 1, \ldots, N_{variables}$ **do**

$X_{\min}^{(n)} \leftarrow \min_{l=1,\ldots,N_{\text{samples}}} \left(X_{\text{real}}^{(n)}(l)\right) - \varepsilon_{\min}^{(n)}$, for $\varepsilon_{\min}^{(n)} \geq 0$

$X_{\max}^{(n)} \leftarrow \max_{l=1,\ldots,N_{\text{samples}}} \left(X_{\text{real}}^{(n)}(l)\right) + \varepsilon_{\max}^{(n)}$, for $\varepsilon_{\max}^{(n)} \geq 0$

for $l = 1, \ldots, N_{samples}$ **do**

$$X_{\text{integer}}^{(n)}(l) \leftarrow \text{int}\left(\left(2^M - 1\right) \frac{X_{\text{real}}^{(n)}(l) - X_{\min}^{(n)}}{X_{\max}^{(n)} - X_{\min}^{(n)}}\right)$$

$$X_{\text{binary}}^{(n)}(l) \leftarrow \text{bin}\left(X_{\text{integer}}^{(n)}(l)\right)$$

end

end

Each data sample is represented by an M-digit binary number with every digit becoming a separate feature. The total number of features is $M \times N_{\text{variables}}$.

Algorithm 7: Binary to integer to continuous transformation (sampling phase)

Result: Conversion of the generated M-digit binary sample into continuous sample.

Input: $\left(\widehat{X}^{(n)}_{[m]}\right)_{m=0,\ldots,M-1;\ n=1,\ldots,N_{\text{variables}}}$ – generated M-digit binary sample.

for $n = 1, \ldots, N_{variables}$ **do**

$$\widehat{X}^{(n)}_{\text{integer}} := \sum_{m=0}^{M-1} 2^m \widehat{X}^{(n)}_{[M-1-m]}$$

$$\widehat{X}^{(n)}_{\text{real}} \leftarrow X^{(n)}_{\text{min}} + \frac{1}{2^M - 1} \widehat{X}^{(n)}_{\text{integer}} \left(X^{(n)}_{\text{max}} - X^{(n)}_{\text{min}}\right)$$

end

Algorithms 6 and 7 describe the transformations of continuous variables into M-digit binary variables and then back into continuous variables [173]. It is important to note the role of the parameters ε_{min} and ε_{max}. They are non-negative and expand the interval on which the variables are defined. In the case where $\varepsilon_{\text{min}} = \varepsilon_{\text{max}} = 0$, this interval is determined by the minimum and maximum values of the variable as observed in the training dataset. By allowing ε_{min} and ε_{max} to take positive values, we expand the interval of possible values the variable can take. This allows the model to generate a wider range of possible scenarios: with some (small) probability the generated values can fall outside the interval given by the samples from the training dataset.

The precision of the binary representation is feature specific. More important features can have more granular representation. The right choice of precision is important for NISQ devices that operate with a limited number of quantum registers. For example, the QCBM ansatz shown in Figure 9.1 can be used to encode two continuous variables with 6-digit binary precision each. Alternatively, the more important

variable can be encoded with, e.g., 8-digit binary precision and the less important one with only 4-digit binary precision.

Figure 9.2 illustrates how the readout from 12 quantum registers can be translated into a sample consisting of two continuous variables: the value of the first one is encoded as a 7-digit binary number and the value of the second one is encoded as a 5-digit binary number. In this example, we assume that both variables take values in the interval [−1,1].

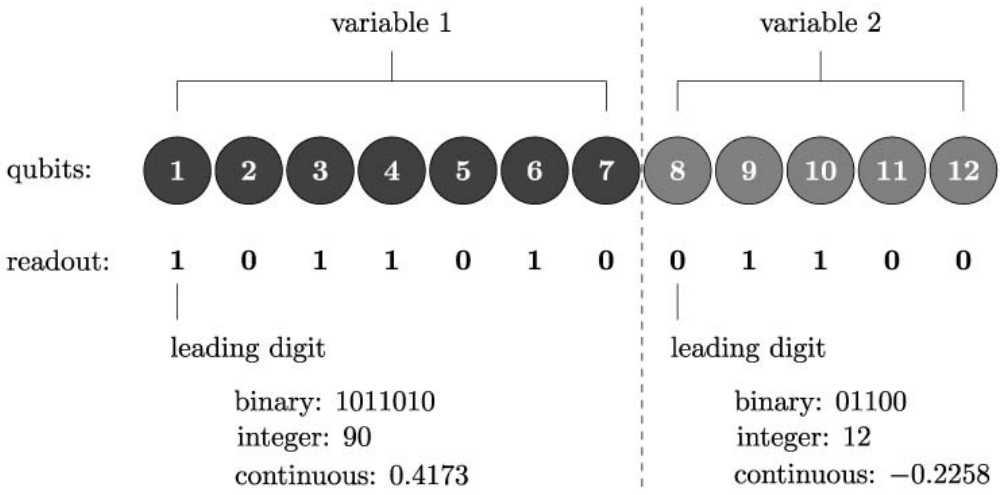


Figure 9.4: Sample QCBM readout and data transformation for two continuous variables taking values in the interval [−1,1] and where we set $\varepsilon_{min} = \varepsilon_{max} = 0$.

9.2.2 Choosing the right cost function

The differentiable learning of QCBM follows the same principles as that of training the quantum neural networks outlined in Chapter 8: minimisation of the cost function with the gradient descent method. The main difference is the form of the cost function. In the case of a QNN-based classifier, the cost function represents the classification error while the cost function for QCBM represents the distance between two probability distributions: the distribution of samples in the training dataset and the distribution of samples in the generated dataset.

Let θ denote the set of adjustable QCBM parameters, $p_\theta(\cdot)$ the QCBM distribution, and $\pi(\cdot)$ the data distribution. Then we can define the cost function $L(\theta)$ as

$$L(\theta) := \sum_{\mathrm{x}} |p_\theta(\mathrm{x}) - \pi(\mathrm{x})|,$$

where the sum goes over all samples x in the dataset. This cost function is a strong metric but may not be the easiest to deal with [73]. An efficient alternative choice of the cost function is the *maximum mean discrepancy* [189]:

$$L(\theta) := \mathop{\mathbb{E}}_{\mathrm{x}\sim p_\theta, \mathrm{y}\sim p_\theta}[K(\mathrm{x},\mathrm{y})] - 2\mathop{\mathbb{E}}_{\mathrm{x}\sim p_\theta, \mathrm{y}\sim \pi}[K(\mathrm{x},\mathrm{y})] + \mathop{\mathbb{E}}_{\mathrm{x}\sim \pi, \mathrm{y}\sim \pi}[K(\mathrm{x},\mathrm{y})],$$

where $K(\cdot,\cdot)$ is a *kernel function*, i.e., a measure of similarity between points in the sample space. A popular choice of kernel function is the Gaussian mixture:

$$K(\mathrm{x},\mathrm{y}) = \frac{1}{c}\sum_{i=1}^{c} \exp\left(-\frac{\|\mathrm{x}-\mathrm{y}\|^2}{2\sigma_i^2}\right),$$

for some $c \in \mathbb{N}$ and where $(\sigma_i)_{i=1,\ldots,c}$ are the bandwidth parameters of each Gaussian kernel and $\|\cdot\|$ is the L_2 norm.

We can also explore the possibility of using *quantum kernels*. Quantum kernels can provide an advantage over classical methods for kernels that are difficult to compute on a classical device. For example, we can consider a non-variational quantum kernel method [232], which uses a quantum circuit U(x) to map real data into a quantum state $|\varphi\rangle$ via a *feature map*:

$$|\phi(\mathrm{x})\rangle = \mathrm{U}(\mathrm{x})\ |0\rangle^{\otimes n}.$$

The kernel function is then defined as the squared inner product

$$K(\mathrm{x},\mathrm{y}) = |\langle\phi(\mathrm{x})|\phi(\mathrm{y})\rangle|^2.$$

This quantum kernel is evaluated on a quantum computer and is hard to compute on a classical one [129]. We investigate the question of expressive power of various models in Chapter 12 and provide a detailed analysis of the quantum kernel approach in Chapter 13. Taking into account the mapping (9.2.2) and denoting $|\mathbf{0}\rangle = |0\rangle^{\otimes n}$, the kernel becomes

$$K(\mathrm{x},\mathrm{y}) = |\langle\mathbf{0}|\,U^{\dagger}(\mathrm{x})U(\mathrm{y})\,|\mathbf{0}\rangle|^2,$$

which is the probability of measuring the all-zero outcome. It can be calculated by measuring, in the computational basis, the state which results from running the circuit given by U(y), followed by that of $U^{\dagger}(x)$.

9.3 Non-Differentiable Learning of QCBM

The hardware-efficient ansatz we proposed for the QCBM architecture, while simple and intuitive, may be vulnerable to barren plateaus, or regions of exponentially vanishing gradient magnitudes that make training untenable [54, 139, 299]. This provides a strong motivation for exploring a non-differentiable learning alternative such as Genetic Algorithm.

9.3.1 The principles of Genetic Algorithm

GA is a powerful evolutionary search heuristic [214] that was introduced in Chapter 3. It performs a multi-directional search by maintaining a population of proposed solutions (chromosomes) for a given problem. Each solution is represented in a fixed alphabet with an established meaning (genes). The population undergoes a simulated

evolution with relatively good solutions producing offspring, which subsequently replace the worse ones, and the quality of a solution is estimated with some objective function (environment). GAs have found applications is such diverse fields as quantitative finance (for portfolio optimisation problems [172]) and experiments with adiabatic quantum computing (as a classical benchmark [296]).

The simulation cycle is performed in three basic steps. During the selection step, a new population is formed by stochastic sampling (with replacement). Then, some of the members of the newly selected populations recombine. Finally, all new individuals are re-evaluated. The mating process (recombination) is based on the application of two operators: mutation and crossover. Mutation introduces random variability into the population, and crossover exchanges random pieces of two chromosomes in the hope of propagating partial solutions.

The training of the QCBM specified in Figure 9.1 consists of finding an optimal configuration of the rotation angles $(\theta_i^j)_{i=1,\ldots,12;\ j=1,\ldots,7}$ that would minimise a chosen cost function given a particular choice of the fixed 2-qubit gates. Since we only deal with 84 adjustable parameters (rather than tens of thousands), we do not need to implement the crossover mechanism and can rely on parameter mutations to achieve GA convergence to the minimum of the cost function. This significantly simplifies the algorithm.

9.3.2 Training QCBM with a Genetic Algorithm

Algorithm 8 outlines the proposed approach. However, before we provide a formal description of the algorithm, we have to specify the main individual components.

- **Solution.** The solution is a 12 × 7 matrix of rotation angles:

$$\theta = \begin{bmatrix} \theta_1^1 & \dots & \theta_1^7 \\ \vdots & \ddots & \vdots \\ \theta_{12}^1 & \dots & \theta_{12}^7 \end{bmatrix}.$$

In GA language, the matrix θ plays the role of a chromosome and its components θ_i^j play the roles of individual genes.

- **Mutation.** The genes can mutate from generation to generation. The mutation rate can be either constant or time dependent. For example, the mutation rate can start at some large value and then decrease exponentially such that it halves after each κ generations. In Algorithm 8, we adopt the following mutation dynamics:
  - A rotation angle (gene) can mutate to any of the allowed discrete values with equal probability.
  - Mutation is controlled by a single global parameter $\alpha \in (0,1]$, which can be either constant or exponentially decreasing at some fixed rate $\beta \geq 0$.
  - Mutations happen independently for each column in θ.
  - For each column in θ, at each generation, a single rotation angle mutation happens with probability α. All rotation angles are equally likely to mutate. After that, one more mutation can happen with probability $\alpha/2$. Again, all rotation angles are equally likely to mutate. This ensures that we can have scenarios where two rotation angles within the same column can mutate simultaneously.
- **Search Space.** The rotation angles θ_i^j are defined in $[-\pi,\pi]$, which we split into 2^m equal subintervals, so that the possible values for θ_i^j are $(-\pi + n\pi/2^{m-1})_{n=0,\dots,2^m-1}$. A rotation angle can mutate into any of these values. The search space can quickly become enormous even for the relatively small values of m. For example, for $m = 7$ we have 128 possible values for each rotation angle making the total number of possible configurations $\sim 10^{177}$. The GA can only explore a tiny fraction of the search space. But due to the GA's

ability to propagate best solutions and to avoid being trapped in local minima, the algorithm can achieve reasonably fast convergence to the solution in the vicinity of the global minimum. For a detailed analysis of the rate of convergence of genetic algorithms, we refer the interested reader to [130, 264].

- **Cost Function.** A cost function is a measure of how far the distribution of generated samples is from the distribution of original samples provided by the training dataset. Let u := (u_1, …,u_K) be a sample from the training dataset and v(θ) := ($v_1(\theta)$, …,$v_K(\theta)$) a sample from the QCBM generated dataset that corresponds to a particular configuration of rotation angles θ. Let us order these samples from the smallest to the largest with any suitable sort(·) function:

  $$\overline{\mathrm{u}} = \mathrm{sort}(\mathrm{u}), \quad \overline{\mathrm{v}}(\theta) = \mathrm{sort}(\mathrm{v}(\theta)).$$

  The cost function $L(\cdot)$ can then be defined as

  $$L(\theta) := \sum_{k=1}^{K} \left(\overline{u}_k - \overline{v}_k(\theta)\right)^2 .$$

The sort(·) function in (9.3.2) can be, e.g., quicksort [137] or mergesort [166], which belong to the class of divide-and-conquer algorithms. Alternatively, it can be, e.g., heapsort [303] – a comparison-based sorting algorithm.

Algorithm 8: Genetic Algorithm

Result: Optimal configuration of the set of QCBM parameters θ^* minimising the cost function.

Input:

$\mathrm{u} \in \mathbb{R}^K$: vector of sample training dataset;

L: number of iterations (generations);

M: number of best solutions in the given generation, chosen for further mutation;

N: number of solutions in each generation ($N = DM$, $D \in \mathbb{N}$);

α, β: mutation parameters;

m: search space parameter.

The possible values of rotation angles are $\left(-\pi + \frac{\nu\pi}{2^{m-1}}\right)_{\nu=0,\ldots,2^m-1}$.

Initialise and evaluate the first generation of solutions:

for $n = 1, \ldots, N$ **do**

  Generate a configuration $\theta(0;n)$ by randomly drawing each rotation angle $\theta_i^j(0;n)$ from the uniform distribution on the set of possible values of rotation angles given by m.

  for $k = 1, \ldots, K$ **do**

    Run the quantum circuit with configuration $\theta(0;n)$ and generate new sample $v_k(\theta(0;n))$.

  end

  Evaluate the cost function $L(\theta(0;n))$.

end

Order solutions from best (minimum of cost function) to worst (maximum of cost function).

$\theta^* \leftarrow$ configuration corresponding to the minimum of cost function.

Iterations:

for $l = 1, \ldots, L$ **do**

$\alpha \leftarrow \alpha e^{-\beta}$

Select M best solutions from generation $l-1$ and generate new solutions $(\theta(l;n))_{n=1,\ldots,N}$ by mutating the rotation angles using the updated mutation rate α. Each of the M best solutions is used to produce D new solutions.

for $n = 1, \ldots, N$ **do**

for $k = 1, \ldots, K$ **do**

Run the quantum circuit with $\theta(l;n)$ and generate new sample $v_k(\theta(l;n))$.

end

Evaluate the cost function $L(\theta(l;n))$.

end

Order solutions from best (minimum of the cost function) to worst (maximum of the cost function).

$\theta^*(l) \leftarrow$ configuration corresponding to the minimum of the cost function (l-th generation).

if $L(\theta^*(l)) < L(\theta^*)$ **then**

$\theta^* \leftarrow \theta^*(l)$

end

end

Having described the training algorithm, we now specify the classical benchmark before comparing the results obtained by the quantum and the classical generative models on the sample datasets.

9.4 Classical Benchmark

There is a deep connection between QCBM and its classical counterpart – Restricted Boltzmann Machine [60]. RBM, introduced and discussed in Chapter 5 in the context of quantum annealing, is a generative model inspired by statistical physics, where the probability of a particular data sample, v, is given by the Boltzmann distribution:

$$\mathbb{P}(\mathrm{v}) = \frac{1}{Z}\mathrm{e}^{-E(\mathrm{v})}.$$

Here, $E(\mathrm{v})$ is the (positive) *energy* of the data sample (data samples with lower energy have higher probabilities) and Z is the partition function, namely the normalisation factor of the probability density:

$$Z = \sum_{\mathrm{v}} \mathrm{e}^{-E(\mathrm{v})}.$$

Alternatively, we can use the inherent probabilistic nature of quantum mechanics that allows us to model the probability distribution using a quantum state $|\psi\rangle$:

$$\mathbb{P}(\mathrm{v}) = \langle\psi|\mathcal{P}_{\mathrm{v}}^{\dagger}\mathcal{P}_{\mathrm{v}}|\psi\rangle,$$

where $\mathcal{P}_{\mathrm{v}}$ is the measurement operator introduced in Section 1.2.3 and, since the quantum state $|\psi\rangle$ is a unit vector, we have

$$\langle\psi|\psi\rangle = 1.$$

We realise this approach in the Quantum Circuit Born Machine, where generative modelling of probability density is translated into learning a quantum state. The sole purpose of QCBM's parameterised circuit is to create the quantum state $|\psi\rangle$ that encodes the desired probability distribution starting from the initial state $|0\rangle^{\otimes \mathrm{n}}$, with sampling performed by applying the measurement operators.

Therefore, providing a classical benchmark for QCBM consists in finding a suitable RBM configuration that will allow us to compare two methods generating the probability distribution $\mathbb{P}(v)$: one given by (9.4) for RBM and another one given by (9.4) for QCBM [170].

Figure 9.5 shows an RBM with 12 stochastic binary visible activation units and 7 stochastic binary hidden activation units, where $(a_i)_{i=1,\ldots,12}$, $(b_j)_{j=1,\ldots,7}$, and $(w_{ij})_{i=1,\ldots,12;\, j=1,\ldots,7}$ denote, respectively, the biases for the visible and hidden layers and the network weights.

This network architecture makes RBM equivalent to QCBM as described in Section 9.1 in the sense that both generative models have the same number of adjustable parameters (the number of RBM weights is equal to the number of adjustable rotation angles in QCBM) and the number of visible activation units is equal to the number of quantum registers. The latter ensures that both generative models can learn the empirical distribution of a continuous random variable with the same precision (12-digit binary representation).

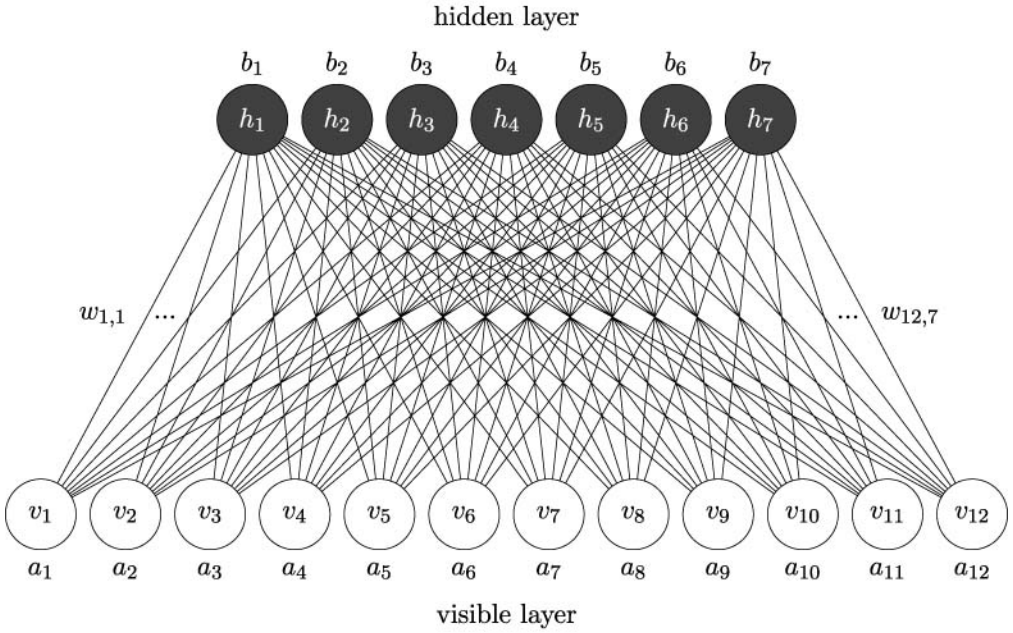


Figure 9.5: RBM(12, 7).



QCBM performance should be compared against the performance of its classical counterpart, the Restricted Boltzmann Machine. Both models operate on the binary representation of the dataset and have a comparable number of adjustable parameters.

9.5 QCBM as a Market Generator

The most obvious financial application of QCBM is as a market generator. An efficient generation of realistic market scenarios, for example sampling from the joint distribution of risk factors, is one of the most important and challenging problems in quantitative finance today. We thus need to investigate how well QCBM can execute this task, and compare it to classical benchmarks.

9.5.1 Non-parametric modelling of market risk factors

Historically, the problem of producing reliable synthetic market scenarios was solved through sampling from some easy-to-calibrate parametric models, such as the multivariate Normal distribution of risk factor log-returns (equities) or a Gaussian copula combining the multivariate Normal dependence structure with heavy-tailed univariate marginal distributions of individual risk factors (credit). However, there are well-known issues with this approach that often outweigh the benefits provided by simplicity and transparency [217].

A parametric model is often a poor approximation of reality. To be useful, it has to be relatively simple: one should be able to describe the key features of the risk factor distribution with a handful of parameters achieving the best possible fit to either the empirical distribution derived from historical data or from prices of traded instruments observed in the market at the time of model calibration. Making the

parametric model too complex would lead to overfitting and poor generalisation.

It is even more difficult to model a realistic dependence structure. A typical parametric approach used in most Monte Carlo risk engines starts with modelling the dynamics of various risk factors independently, and then imposes a dependence structure by correlating the corresponding stochastic drivers. These are, almost invariably, Brownian motions, and the linear correlations between them are supposed to be sufficient to construct the joint distribution of risk factors.

An alternative approach is to use non-parametric modelling, where the joint and marginal distributions of risk factors are learned directly from the available datasets. Classically, we can realise this approach with the help of a Restricted Boltzmann Machine – the classical benchmark of choice described in the previous section and successfully applied to a number of financial use cases [173, 174]. Another possibility is to use the Generative Adversarial Network (GAN) framework, where the distribution learned from the dataset by a generative neural network is tested by a discriminative neural network trying to judge whether samples are coming from the true distribution (data) or from the reconstructed distribution (generated samples) [114].

Chapter 12 explores the question of the larger expressive power of QCBM in comparison with classical neural networks (RBM). However, the first step should be an experimental verification of their performance characteristics. With this in mind, we would like to test the ability of both QCBM and RBM to learn relatively complex probability distributions and then efficiently sample from them.

9.5.2 Sampling from the learned probability distributions

We are going to test the performance of QCBM and RBM on two datasets:

- **Dataset A.** A heavy-tailed distribution of daily S&P 500 index returns observed between 5 January 2009 and 22 February 2011 (UCI Machine Learning Repository [10, 9]). The dataset consists of 536 samples.
- **Dataset B.** A specially constructed distribution of a continuous random variable with a highly spiky probability density function (pdf) modelled as a mixture of Normal distributions. The dataset consists of 5,000 generated samples from a mixture of four Normal distributions with the following means, standard deviations, and weights:

Mean	Standard deviation	Weight
−3	0.3	0.1
−1	0.3	0.2
1	0.3	0.3
3	0.3	0.4

Table 9.1: Parameters of the mixture of standard Normal distributions.

In both cases, we convert the continuous samples into the corresponding 12-digit binary representation as per Algorithm 6. Once the networks are trained (QCBM with Algorithm 8 and RBM with Algorithm 2), we generate new samples: 536 new samples for Dataset A and 5,000 new samples for Dataset B. This allows us to visualise the quality of the generated samples (once they are converted into the corresponding continuous representation as per Algorithm 7) by producing the empirical pdf and the QQ-plots as shown in Figures 9.6 and 9.7, which display sample simulation results for the fully trained models. We can see that both QCBM(12, 7) and RBM(12, 7) can successfully learn complex empirical distributions (heavy-tailed in the

case of Dataset A and light-tailed with spiky pdf in the case of Dataset B). We have chosen CX for the fixed gates in QCBM and used the Qiskit quantum simulator to simulate the quantum parts of the training and sampling algorithms.

The following sets of hyperparameters were used to train the models:

- **Genetic Algorithm for training QCBM (Algorithm 8)** $N = 1000$, $M = 25$, $m = 7$, $\alpha = 1.0$, $\beta = 0.013863$, $\kappa = 50$, $L = 200$. The value of β ensures that mutation rate halves after each κ generations.
- **Contrastive Divergence algorithm for training RBM** (`sklearn.neural_network.BernoulliRBM`)
  n_components = 7 – number of hidden activation units for RBM(12, 7) learning_rate = 0.0005 – learning rate η in Algorithm 2
  batch_size = 10 – size of the training minibatches S in Algorithm 2
  n_iter = 40000 – number of iterations

Although a visual inspection of the pdf and QQ-plots in Figures 9.6 and 9.7 suggests that both QCBM and RBM are doing a good job in generating high-quality samples from the learned empirical distributions encoded in model parameters, we would like to have a more objective measure of the model performance. This is especially important since we deal with generative models and very little can be concluded from a single model run.

Running the quantum circuit multiple times for a particular configuration of model parameters (e.g., an optimal set of rotation angles found with the help of GA) results in the distribution of objective function values. This gives us an idea of what metrics can be used to measure the performance of QCBM and RBM [170]. The cost function (9.3.2) we used for training QCBM can be calculated on the samples generated by RBM. In other words, we can compare the performance of QCBM and RBM by comparing the distributions of the

cost function values calculated for the samples generated by these models.

Table 9.2 shows the means and standard deviations of the cost functions calculated for 100 runs of QCBM(12, 7) and RBM(12, 7). Each run generated 5,000 samples from the learned empirical distribution (the models were trained on Dataset B, which consists of 5,000 samples from the mixture of four Normal distributions).

Model	Mean	Standard deviation
QCBM(12, 7)	30.5	23.6
RBM(12, 7)	39.6	30.8

Table 9.2: Cost function statistics for the models trained on Dataset B.

It is clear from Table 9.2 that QCBM(12, 7) with a weakly optimised set of hyperparameters performs better than RBM(12, 7) trained with equally weakly optimised hyperparameters (a small learning rate combined with a large number of iterations and the small size of the minibatches [134]). Although this cannot be seen as proper evidence of quantum advantage, this nevertheless opens the gate to promising further research.

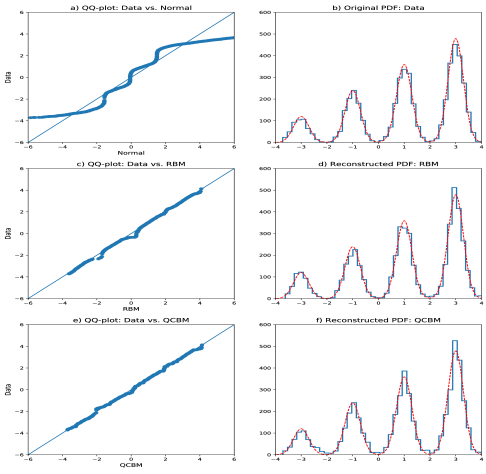


Figure 9.6: Mixture of Normal distributions.

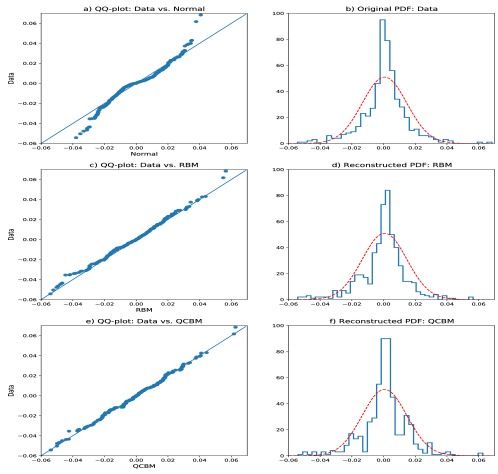


Figure 9.7: Distribution of S&P 500 index returns.

Let us now turn our attention to Dataset A. The dataset consists of just 536 samples and, as we can see in Figure 9.7, the empirical pdf displays pronounced heavy tails which are also clearly seen in the QQ-plot against the Normal distribution. The relatively small number of samples means that we have to deal with a substantial amount of noise. Therefore, we need to use a robust statistical test to compare QCBM and RBM. Since we are working with a univariate distribution, we can estimate the quality of generated samples with the Kolmogorov-Smirnov test [233].

Table 9.3 provides the p-values and Kolmogorov-Smirnov statistics for the RBM and the QCBM generated samples as well as a Normal distribution fitted to the original dataset (by matching the first two moments). The p-value represents the probability of obtaining test results supporting the null hypothesis of the two datasets coming from the same distribution. In the context of our numerical experiments, the larger the p-value the more likely the generated samples were drawn from the correct distribution.

Distribution	p-value	K-S statistic
Normal	0.004 ± 0.009	0.121 ± 0.017
RBM generated samples	0.46 ± 0.23	0.055 ± 0.011
QCBM generated samples	0.46 ± 0.11	0.053 ± 0.005

Table 9.3: p-value and K-S statistic for Normal, RBM and QCBM generated samples in the format: mean ± standard deviation. Number of Normal, RBM and QCBM generated datasets: 20. Number of samples in each generated dataset: 536 (equal to the number of samples in the original dataset).

The K-S statistic takes the largest absolute difference between the two distribution functions across all values of the random variable. The larger the K-S statistic the less likely the generated samples were drawn from the correct distribution. The K-S statistic can be compared with the critical values calculated for the given confidence level and number of samples. For example, the critical value corresponding to the 95th percentile confidence level and 536 samples in both datasets is 0.0587. If the K-S statistic is larger then, with 95% certainty, we can reject the null hypothesis that 536 generated samples were drawn from the right distribution.

The first observation is that we can definitely reject the null hypothesis that the daily S&P 500 index returns are Normally distributed. The corresponding p-value is much smaller than 1, and the K-S statistic is twice the critical value. More importantly, QCBM performs at par with RBM in terms of both the p-value and the K-S statistic: we therefore cannot reject the null hypothesis that QCBM and RBM generated samples were drawn from the same distribution as the original dataset.

9.5.3 Training algorithm convergence and hyperparameter optimisation

Next, we would like to explore the GA behaviour for various model configurations. In particular, it is interesting to investigate the

algorithm convergence for different types of fixed gates, not just CX, and for different choices of the mutation rate. The charts in Figure 9.8 confirm our intuition about CX being the best choice of fixed gate given the configuration of one-qubit gates (Figure 9.1) and the exponentially decreasing mutation rate performing better than the constant mutation rates. Here, we continue working with Dataset B.

As we can see in Figure 9.1, the fixed gates are flanked by one-qubit gates performing rotations around the z-axis. Therefore, adding another rotation around the z-axis by $\varphi = \pi$ ($Z = R_Z(\pi)$) may not offer the same flexibility as rotation around the x-axis by $\varphi = \pi$ ($X = R_X(\pi)$). Controlled rotations around the z-axis by an angle $\varphi < \pi$ are likely to perform even worse. This is exactly what we see in Figure 9.8 (left chart) for three different types of fixed gates: CX, CZ, and $CR_Z(\pi/4)$.

Our intuition about the optimal choice of mutation rate suggests that it should be productive to start the algorithm with a really large mutation rate in order to explore the search space as broadly as possible (the "exploration" phase). Then, as the algorithm finds progressively better solutions, it should be useful to reduce the mutation rate in order to perform a more detailed search in the vicinity of the best solutions found so far (the "exploitation" phase). As the algorithm converges, we may want to perform more and more refined searches by only mutating one or two parameters. Figure 9.8 (right chart) shows that this is indeed the case. Here, the maximum value of the mutation rate is $\alpha = 1.0$ and the minimum value is $\alpha = 0.0625$ – the value reached after $L = 200$ algorithm iterations when the algorithm is run with the initial value of mutation rate $\alpha = 1.0$ and exponential decay factor $\beta = 0.013863$.

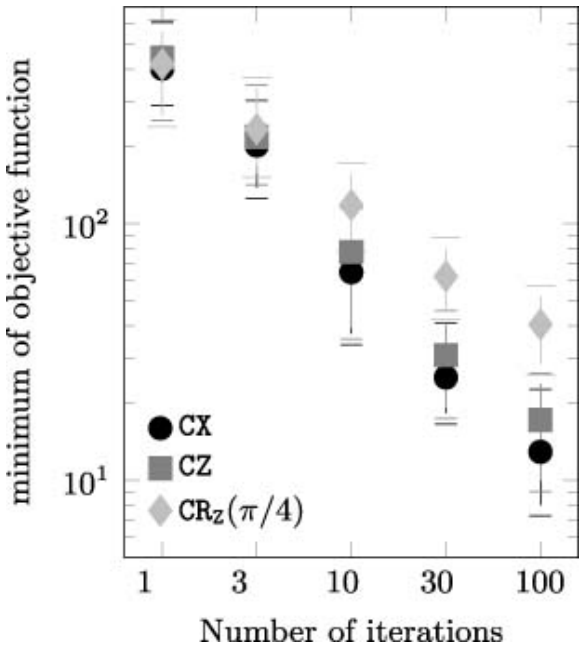


Figure 9.8: Left: GA convergence as a function of fixed gate type. Right: GA convergence as a function of mutation rate for CX fixed gates. Dots indicate mean values and error bars indicate the 10th and the 90th percentiles. GA parameters: N = 1000, M = 25, m = 7, 20 GA runs.

Finally, we need to investigate the convergence of the algorithm as a function of the rotation angle discretisation scheme. In principle, an arbitrary rotation poses a problem as it must be approximated by a sequence of discrete gates because only discrete sets of gates can be implemented fault-tolerantly [180]. Since a GA operates on a discrete set of rotation angles, we face a trade-off between higher accuracy achieved through a finer discretisation scheme and implementation efficiency in the case of a less granular set of rotation angles. Additionally, all rotation gates can be executed with finite precision and the discretisation scheme should take this into account. Hence, in order to facilitate the efficient implementation of the rotation gates $R_X(\theta)$ and $R_Z(\theta)$, the GA operates on the rotation angles θ that take discrete values $(-\pi + \nu\pi/2^{m-1})_{\nu=0,\ldots,2^m-1}$, thus splitting the $[-\pi,\pi]$ interval into 2^m equal subintervals.

Therefore, we must answer the question of GA convergence for various values of m. Figure 9.9 shows the minimum values of the objective

function (9.3.2) as a function of the number of algorithm iterations for three different values of m:

- $m = 3$, rotation angle step $\Delta\theta = \pi/4$;
- $m = 5$, rotation angle step $\Delta\theta = \pi/16$;
- $m = 7$, rotation angle step $\Delta\theta = \pi/64$.

We can see that the GA performance improves only marginally for $m > 5$. This is good news suggesting that it may be sufficient to operate with rotation angle step $\Delta\theta = \pi/16$ to achieve the desired precision in learning the target distribution.

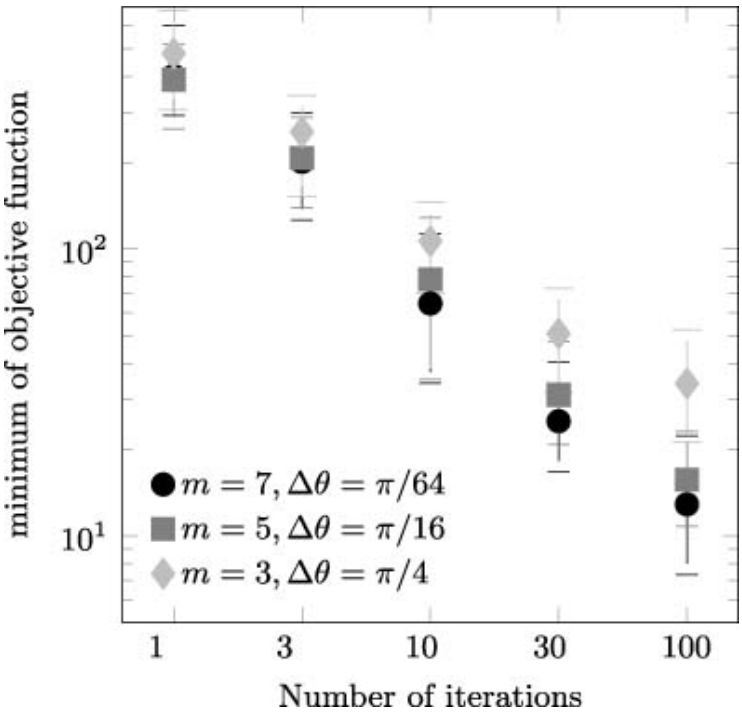


Figure 9.9: GA convergence as a function of the rotation angle discretisation scheme for CX fixed gates and exponentially decreasing mutation rate. Dots indicate mean values and error bars indicate the 10th and the 90th percentiles. GA parameters: N = 1000, M = 25, α = 1.0, β = 0.013863, 20 GA runs.

The non-differentiable learning of QCBM with Genetic Algorithm is a viable approach to the training of PQCs. QCBM trained with GA performs at least as well as an equivalent classical neural network (RBM). The performance of QCBM and its classical counterpart were

tested on two different datasets (heavy-tail distributed samples derived from the financial time series and light-tail distributed samples drawn from the specially constructed distribution with spiky pdf) and in both cases QCBM demonstrated its ability to learn the empirical distribution and generate new synthetic samples that have the same statistical properties as the original ones, as can be seen in the pdf and QQ-plots.

Analysing the GA convergence for different sets of hyperparameters, we observe that the best results were achieved with CX fixed gates and an exponentially decreasing mutation rate (starting from the maximum value of the mutation rate and setting the decay rate at a reasonably small value). More importantly, we see that more granular rotation angle discretisation schemes provide progressively less incremental value beyond some point. This means that for many practical purposes it is sufficient to implement rotations with the step $\Delta\theta = \pi/16$ in order to encode target distribution with the desired accuracy for deep enough QCBM architectures (at least two layers of fixed 2-qubit gates). Since qubit rotations on NISQ devices can be implemented with finite precision, this ensures that QCBMs can be used productively for many real-world use cases.



QCBM is a viable choice for building market generators. It performs at least as well as its classical counterpart, RBM, and demonstrates potential for achieving quantum advantage on near-term quantum processors.

Summary

In this chapter, we learned how to construct and train a generative QML model – Quantum Circuit Born Machine. We started with the general concept of a PQC as a generative model, where the readout

operation produces a sample from the probability distribution encoded in the PQC parameters.

Next, we introduced the concept of a hardware-efficient PQC ansatz. Additionally, to build a model that is compatible with QPU connectivity and can easily be embedded into a QPU graph, we tried to use adjustable (one-qubit) and fixed (two-qubit) gates from the set of the native quantum gates for the given system.

Then, we studied differentiable and non-differentiable learning algorithms and experimented with the QCBM trained using Genetic Algorithm. Comparison with the classical benchmark (RBM) demonstrated a realistic possibility of quantum advantage for generative QML models.

Finally, we explored the question of training algorithm convergence for various sets of model parameters.

In the next chapter, we will study another important and exceptionally promising QML model – Variational Quantum Eigensolver.

Join our book's Discord space

Join our Discord community to meet like-minded people and learn alongside more than 2000 members at: https://packt.link/quantum



10
Variational Quantum Eigensolver

Parameterised quantum circuits can find many possible applications outside the quantum machine learning use cases considered in the previous chapters. They can be used to solve problems as diverse as portfolio optimisation [168] and protein folding [248]. However, one aspect remains the same regardless of the specifics of the particular algorithm: the construction of a quantum state with desired characteristics through an optimal PQC configuration (ansatz) and an optimal set of adjustable PQC parameters. This, in turn, is done through the minimisation of some cost function – it can be a classification error in the case of a QNN-based classifier or a distance between two distributions in the case of QCBM.

The Variational Quantum Eigensolver (VQE) is a PQC-based algorithm that aims to find the smallest eigenvalue (the lowest energy) of a problem Hamiltonian. As we know from Chapter 3, the objective functions of many NP-hard combinatorial optimisation problems can be encoded in the Hamiltonians of quantum systems – thus finding the ground state of the Hamiltonian gives us the minimum of the objective function. The VQE was originally proposed in [231] and quickly became one of the most popular tools for experimenting with the wide range of optimisation problems solvable on NISQ devices [153, 215]. The variational part of the algorithm refers to the systematic search for the best possible approximation of the ground state by trying various PQC ansätze and configurations of adjustable PQC parameters – the variational approach.

10.1 The Variational Approach

Let us start with recollecting the details of training discriminative (QNN) and generative (QCBM) models. In both cases our task was to find an optimal configuration of PQC parameters (e.g., rotation angles of adjustable one-qubit gates) such that the resulting quantum state had the desired properties: we could either sample from the encoded probability distribution (generative model) or obtain a class label for the given sample (discriminative model). The

process of finding an optimal configuration of PQC parameters is called *learning* when we are dealing with QML use cases. This learning can be done either in a differentiable or in a non-differentiable way, but it always consists of the minimisation of some cost function through varying the adjustable circuit parameters.

What if the cost function we want to minimise is encoded in the problem Hamiltonian and the task is formulated as finding its ground state? In Chapter 3, we saw how this problem can be solved on a quantum annealer. But it is also possible to find the lowest energy state of a quantum system (or at least a good approximation) using a gate model quantum computer within the PQC framework.

The characteristic equation for the Hamiltonian $\mathcal{H}$ reads

$$\mathcal{H} \left|\psi_i\right\rangle = E_i \left|\psi_i\right\rangle ,$$

where $\left|\psi_i\right\rangle$ is an eigenstate associated with the eigenvalue E_i. The objective is to find the smallest eigenvalue E_0 (the lowest energy) of $\mathcal{H}$ corresponding to the ground state (the lowest energy state) $\left|\psi_0\right\rangle$. This would be a straightforward task if the latter was known since the eigenvalue (energy) of $\mathcal{H}$ is simply the expectation of $\mathcal{H}$:

$$\left\langle\psi_i\right| \mathcal{H} \left|\psi_i\right\rangle = \left\langle\psi_i\right| E_i \left|\psi_i\right\rangle = E_i \left\langle\psi_i|\psi_i\right\rangle = E_i.$$

We will explain below how this expectation is calculated on a quantum computer. However, in most cases the ground state is not known. In fact, the task is to find the ground state that encodes the solution to the optimisation problem by searching for the state that minimises the expectation value of $\mathcal{H}$. What we can do is to construct a progressively better approximation of the ground state, yielding a tighter and tighter upper bound for the ground state energy E_0.

The variational approach is motivated by the spectral theorem introduced in Chapter 1, which allows us to expand the Hermitian Hamiltonian $\mathcal{H}$ as

$$\mathcal{H} = \sum_i E_i \left|\psi_i\right\rangle \left\langle\psi_i\right| .$$

Assume that we constructed a state $|\psi\rangle$, which is an approximation of the true ground state $|\psi_0\rangle$. As we know from Chapter 1, the expectation value of $\mathcal{H}$ in state $|\psi\rangle$ is $\langle\psi|\mathcal{H}|\psi\rangle$. Substituting $\mathcal{H}$ given by (10.1) into this, we obtain, by linearity,

$$\begin{aligned}\langle\psi|\,\mathcal{H}\,|\psi\rangle &= \langle\psi|\left(\sum_i E_i\,|\psi_i\rangle\,\langle\psi_i|\right)|\psi\rangle \\ &= \sum_i E_i\,\langle\psi|\psi_i\rangle\,\langle\psi_i|\psi\rangle = \sum_i E_i\,|\langle\psi,\psi_i\rangle|^2\,.\end{aligned}$$

Expression (10.1) shows that the expectation of $\mathcal{H}$ on any state $|\psi\rangle$ can be expressed as a linear combination of the eigenvalues of $\mathcal{H}$ with all weights greater than or equal to zero, since $|\langle\psi,\psi_i\rangle|^2 \geq 0$ for each i. Therefore, we obtain

$$\langle\psi|\,\mathcal{H}\,|\psi\rangle \geq E_0,$$

since E_0 is the smallest eigenvalue of $\mathcal{H}$ and all coefficients (weights) in the linear combination (10.1) are non-negative.

The role of PQC is to produce the candidate state $|\psi\rangle$. The *variational* part of the algorithm consists of iterative improvements of the candidate state (iterative updates of the adjustable parameters). This is something that can be done as a classical part of the hybrid quantum-classical protocol. The quantum part of the algorithm consists of running the PQC and then measuring $\mathcal{H}$ on the constructed quantum state in order to obtain the expectation value of $\mathcal{H}$.



The variational approach allows us to solve hard optimisation problems encoded in the Hamiltonian on the digital gate model quantum computer – an alternative to adiabatic quantum computing since not all optimisation problems can be efficiently formulated in a QUBO format.

10.2 Calculating Expectations on a Quantum Computer

The key element of the VQE algorithm is the calculation of the expectation value. We now show how this can be performed on a quantum computer. We start with the one-qubit case and then generalise the proposed approach to the two-qubit and multi-qubit cases.

10.2.1 The one-qubit case

Consider the simplest case of a one-qubit system. Since any 2 × 2 unitary and Hermitian matrix can always be decomposed into a sum of the Pauli matrices X, Y, Z and an identity matrix I (more on this in Section 10.2.3), we can represent any one-qubit Hamiltonian as

$$\mathcal{H} = a\mathtt{X} + b\mathtt{Y} + c\mathtt{Z} + d\mathtt{I},$$

where a, b, c, and d are some real coefficients. For a given state $|\psi\rangle$, the expectation value of the Hamiltonian (10.2.1) is given by

$$\langle\mathcal{H}\rangle \equiv \langle\psi|\,\mathcal{H}\,|\psi\rangle = a\,\langle\psi|\,\mathtt{X}\,|\psi\rangle + b\,\langle\psi|\,\mathtt{Y}\,|\psi\rangle + c\,\langle\psi|\,\mathtt{Z}\,|\psi\rangle + d\,\langle\psi|\,\mathtt{I}\,|\psi\rangle\,.$$

The expectation value of $\mathcal{H}$ is computed by adding the expectation values of all its terms, which means that we can compute the expectation values of the Pauli terms independently and then sum them up to obtain $\langle\mathcal{H}\rangle$. We can do it by first constructing the state $|\psi\rangle$ with the help of a PQC and then by performing measurement in the computational basis. The cycle of constructing the state and performing measurement should be repeated a sufficient number of times in order to obtain accurate statistics. Let us go through the terms of the Hamiltonian (10.2.1) one by one to see how it can be done.

We start with the last term, which is the identity operator I multiplied by the coefficient d. This is a trivial case and we do not even need to run a quantum circuit to compute its expectation value, since the expectation value of I is 1:

$$\langle\psi|\,\mathtt{I}\,|\psi\rangle = \langle\psi|\psi\rangle = 1,$$

so that this term will contribute d to $\langle\mathcal{H}\rangle$.

We move now to the next term, cZ. The measurement is performed in the computational basis, which is the z-basis. In this basis, $|\psi\rangle$ can be represented

as a superposition of the basis states $|0\rangle$ and $|1\rangle$ as

$$|\psi\rangle = \alpha_z\,|0\rangle + \beta_z\,|1\rangle\,,$$

with $\alpha_z, \beta_z \in \mathbb{C}$. The expectation $\langle\psi|\mathrm{Z}|\psi\rangle$ is then calculated as

$$\begin{aligned}\langle\psi|\mathrm{Z}|\psi\rangle &= |\alpha_z|^2\,\langle 0|\mathrm{Z}|0\rangle + \alpha_z{}^*\beta_z\,\langle 0|\mathrm{Z}|1\rangle + \alpha_z\beta_z{}^*\langle 1|\mathrm{Z}|0\rangle + |\beta_z|^2\,\langle 1|\mathrm{Z}|1\rangle \\ &= |\alpha_z|^2\langle 0|0\rangle - \alpha_z{}^*\beta_z\langle 0|1\rangle + \alpha_z\beta_z{}^*\langle 1|0\rangle - |\beta_z|^2\langle 1|1\rangle \\ &= |\alpha_z|^2 - |\beta_z|^2,\end{aligned}$$

using the definition of the Z-gate (Chapter 6) and orthogonality of the basis states.

By definition, $|\alpha_z|^2$ and $|\beta_z|^2$ are the probabilities that after the z-basis measurement, the quantum state $|\psi\rangle$ will become $|0\rangle$ or $|1\rangle$ respectively. In order to find that value, we should run the quantum circuit (to construct state $|\psi\rangle$) and then perform measurement N times. The probability of finding a qubit in state $|0\rangle$ is then estimated as n_0/N, where n_0 is the number of state $|0\rangle$ measurements. Similarly, the probability of finding a qubit in state $|1\rangle$ can be estimated as n_1/N, where n_1 is the number of state $|1\rangle$ measurements.

Therefore, the contribution of the Z-term to $\langle\mathcal{H}\rangle$ is given by

$$c\,\langle\psi|\,\mathrm{Z}\,|\psi\rangle = c\frac{n_0 - n_1}{N}.$$

Now we can move to the first two terms on the right side of (10.2.1). Recall that $|0\rangle$ and $|1\rangle$ are the eigenstates of Z with corresponding eigenvalues +1 and −1, namely

$$\mathrm{Z}\,|0\rangle = |0\rangle \quad \text{and} \quad \mathrm{Z}\,|1\rangle = -\,|1\rangle\,.$$

Furthermore, the eigenstates of X are

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \quad \text{and} \quad |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}},$$

and the eigenstates of Y are

$$|\mathrm{R}\rangle = \frac{|0\rangle + \mathrm{i}\,|1\rangle}{\sqrt{2}} \quad \text{and} \quad |\mathrm{L}\rangle = \frac{|0\rangle - \mathrm{i}\,|1\rangle}{\sqrt{2}}.$$

Their corresponding eigenvalues also are +1 and −1, so that

$$\mathtt{X}\,|+\rangle = |+\rangle\,, \quad \mathtt{X}\,|-\rangle = -\,|-\rangle\,, \quad \mathtt{Y}\,|\mathrm{R}\rangle = |\mathrm{R}\rangle\,, \quad \mathtt{Y}\,|\mathrm{L}\rangle = -\,|\mathrm{L}\rangle\,.$$

Therefore, the quantum state $|\psi\rangle$ can also be decomposed into the superposition of the basis states $\{|\mathrm{R}\rangle,|\mathrm{L}\rangle\}$ (y-basis) and $\{|+\rangle,|-\rangle\}$ (x-basis):

$$|\psi\rangle = \alpha_x\,|+\rangle + \beta_x\,|-\rangle = \alpha_y\,|\mathrm{R}\rangle + \beta_y\,|\mathrm{L}\rangle\,.$$

If we can perform measurement in the x-basis and the y-basis, the expectations $\langle\psi|\mathrm{X}|\psi\rangle$ and $\langle\psi|\mathrm{Y}|\psi\rangle$ can be calculated in exactly the same way as the expectation $\langle\psi|\mathrm{Z}|\psi\rangle$, namely

$$a\,\langle\psi|\,\mathtt{X}\,|\psi\rangle = a\frac{n_+ - n_-}{N}, \quad b\,\langle\psi|\,\mathtt{Y}\,|\psi\rangle = b\frac{n_\mathrm{R} - n_\mathrm{L}}{N}.$$

Here, n_+ and n_- are the numbers of measurements in the x-basis that correspond, respectively, to the $|+\rangle$ and $|-\rangle$ outcomes, and n_R and n_L are the numbers of measurements in the y-basis that correspond, respectively, to $|\mathrm{R}\rangle$ and $|\mathrm{L}\rangle$ outcomes.

However, it may be the case that we can only perform measurement in the z-basis. In this case, we need to apply some additional gates to $|\psi\rangle$ before the measurement, such that the probability of measuring $|0\rangle$ in the z-basis is the same as the probability of measuring $|+\rangle$ in the x-basis if we are calculating $\langle\psi|\mathrm{X}|\psi\rangle$, or the probability of measuring $|0\rangle$ in the z-basis is the same as the probability of measuring $|\mathrm{R}\rangle$ in the y-basis if we are calculating $\langle\psi|\mathrm{Y}|\psi\rangle$. Denoting these gates H and G, we have

$$\mathtt{H}\,|\psi\rangle = \mathtt{H}\,(\alpha_x\,|+\rangle + \beta_x\,|-\rangle) = \alpha_x\,|0\rangle + \beta_x\,|1\rangle\,,$$

with $\mathrm{H}|+\rangle = |0\rangle$ and $\mathrm{H}|-\rangle = |1\rangle$ and

$$\mathtt{G}\,|\psi\rangle = \mathtt{G}\,(\alpha_y\,|\mathrm{R}\rangle + \beta_y\,|\mathrm{L}\rangle) = \alpha_y\,|0\rangle + \beta_y\,|1\rangle\,,$$

with $\mathrm{G}|\mathrm{R}\rangle = |0\rangle$ and $\mathrm{G}|\mathrm{L}\rangle = |1\rangle$.

The operators H (which is simply the Hadamard operator) and G admit the matrix representations

$$\mathtt{H} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad \text{and} \quad \mathtt{G} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -\mathrm{i} \\ 1 & \mathrm{i} \end{bmatrix}.$$

10.2.2 The two-qubit case

What if the problem Hamiltonian has terms involving more than a single qubit? Consider a Hamiltonian with terms consisting of tensor products of Pauli matrices such as X⊗Y, Y⊗Z, etc. The general approach remains the same: the expectation value of the Hamiltonian consists of the sum of expectation values of all its terms. Thus, we need to know how to calculate the expectation value of the product of Pauli matrices. Without loss of generality, consider the X⊗Y term – as the very same logic applies to all other Pauli tensor products.

Recall that X⊗Y is the tensor product of the two Pauli operators X and Y, each acting on their own qubits, not a sequential application of gates X and Y to the same qubit. Indeed, given for two unitary operators U_1 and U_2, the tensor product $U_1 \otimes U_2$ acts on the state of a two-qubit system as

$$(\mathtt{U}_1 \otimes \mathtt{U}_2)\, |\psi_1\rangle \otimes |\psi_2\rangle = \Big(\mathtt{U}_1\, |\psi_1\rangle\Big) \otimes \Big(\mathtt{U}_2\, |\psi_2\rangle\Big).$$

We immediately see from (10.2.2) that the tensor product of eigenvectors of X and Y is an eigenvector of X⊗Y. Indeed, if $\mathtt{U}|\psi_\mathtt{U}\rangle = E_\mathtt{U}|\psi_\mathtt{U}\rangle$ with $E_\mathtt{U} \in \mathbb{C}$, for $\mathtt{U} \in \{\mathtt{X},\mathtt{Y}\}$, then

$$(\mathtt{X}\otimes\mathtt{Y})\, |\psi_\mathtt{X}\rangle \otimes |\psi_\mathtt{Y}\rangle = \mathtt{X}\, |\psi_\mathtt{X}\rangle \otimes \mathtt{Y}\, |\psi_\mathtt{Y}\rangle = E_\mathtt{X}\, |\psi_\mathtt{X}\rangle \otimes E_\mathtt{Y}\, |\psi_\mathtt{Y}\rangle = E_\mathtt{X} E_\mathtt{Y}\, |\psi_\mathtt{X}\rangle \otimes |\psi_\mathtt{Y}\rangle\,.$$

We also remember that all eigenvectors of Pauli operators have eigenvalues equal to either +1 or −1 (as detailed in (10.2.1) and (10.2.1)). Now, for the X⊗Y Pauli term, the eigenvectors with eigenvalue +1 are

$$|+\rangle \otimes |\mathrm{R}\rangle = |+\mathrm{R}\rangle \quad \text{and} \quad |-\rangle \otimes |\mathrm{L}\rangle = |-\mathrm{L}\rangle\,,$$

and the eigenvectors with eigenvalue −1 are

$$|+\rangle \otimes |\mathrm{L}\rangle = |+\mathrm{L}\rangle \quad \text{and} \quad |-\rangle \otimes |\mathrm{R}\rangle = |-\mathrm{R}\rangle\,,$$

which follow directly from the computations

$$\begin{aligned}
(\mathtt{X}\otimes\mathtt{Y})\,|+\mathrm{R}\rangle &= (\mathtt{X}\otimes\mathtt{Y})\,|+\rangle\otimes|\mathrm{R}\rangle = \mathtt{X}\,|+\rangle\otimes\mathtt{Y}\,|\mathrm{R}\rangle = |+\rangle\otimes|\mathrm{R}\rangle = |+\mathrm{R}\rangle\,,\\
(\mathtt{X}\otimes\mathtt{Y})\,|+\mathrm{L}\rangle &= (\mathtt{X}\otimes\mathtt{Y})\,|+\rangle\otimes|\mathrm{L}\rangle = \mathtt{X}\,|+\rangle\otimes\mathtt{Y}\,|\mathrm{L}\rangle = |+\rangle\otimes(-\,|\mathrm{L}\rangle) = -\,|+\mathrm{L}\rangle\,,\\
(\mathtt{X}\otimes\mathtt{Y})\,|-\mathrm{R}\rangle &= (\mathtt{X}\otimes\mathtt{Y})\,|-\rangle\otimes|\mathrm{R}\rangle = \mathtt{X}\,|-\rangle\otimes\mathtt{Y}\,|\mathrm{R}\rangle = (-\,|-\rangle)\otimes|\mathrm{R}\rangle = -\,|-\mathrm{R}\rangle\,,\\
(\mathtt{X}\otimes\mathtt{Y})\,|-\mathrm{L}\rangle &= (\mathtt{X}\otimes\mathtt{Y})\,|-\rangle\otimes|\mathrm{L}\rangle = \mathtt{X}\,|-\rangle\otimes\mathtt{Y}\,|\mathrm{L}\rangle = (-\,|-\rangle)\otimes(-\,|\mathrm{L}\rangle) = |-\mathrm{L}\rangle\,.
\end{aligned}$$

Let us write down the representation of a quantum state of the two-qubit system $|\psi\rangle = |\psi_1\rangle\otimes|\psi_2\rangle$ in the basis of X⊗Y eigenvectors:

$$\begin{aligned}
|\psi\rangle &= |\psi_1\rangle\otimes|\psi_2\rangle\\
&= (\alpha_x\,|+\rangle + \beta_x\,|-\rangle)\otimes(\alpha_y\,|\mathrm{R}\rangle + \beta_y\,|\mathrm{L}\rangle)\\
&= \alpha_x\alpha_y\,|+\mathrm{R}\rangle + \alpha_x\beta_y\,|+\mathrm{L}\rangle + \beta_x\alpha_y\,|-\mathrm{R}\rangle + \beta_x\beta_y\,|-\mathrm{L}\rangle\,,
\end{aligned}$$

with $(\alpha_x,\beta_x,\alpha_y,\beta_y) \in \mathbb{C}^4$. We want to apply an operator allowing us to perform measurements in the z-basis such that the probability amplitudes of the corresponding states remain the same. It is easy to see that this operator is a tensor product of the H and G gates:

$$\begin{aligned}
(\mathrm{H}\otimes\mathrm{G})|\psi\rangle &= (\mathrm{H}\otimes\mathrm{G})(\alpha_x\alpha_y\,|+\mathrm{R}\rangle + \alpha_x\beta_y\,|+\mathrm{L}\rangle + \beta_x\alpha_y\,|-\mathrm{R}\rangle + \beta_x\beta_y\,|-\mathrm{L}\rangle)\\
&= \alpha_x\alpha_y(\mathrm{H}\otimes\mathrm{G})|+\rangle\otimes|\mathrm{R}\rangle + \alpha_x\beta_y(\mathrm{H}\otimes\mathrm{G})|+\rangle\otimes|\mathrm{L}\rangle\\
&+ \beta_x\alpha_y(\mathrm{H}\otimes\mathrm{G})|-\rangle\otimes|\mathrm{R}\rangle + \beta_x\beta_y(\mathrm{H}\otimes\mathrm{G})|-\rangle\otimes|\mathrm{L}\rangle\\
&= \alpha_x\alpha_y(\mathrm{H}|+\rangle\otimes\mathrm{G}|\mathrm{R}\rangle) + \alpha_x\beta_y(\mathrm{H}|+\rangle\otimes\mathrm{G}|\mathrm{L}\rangle)\\
&+ \beta_x\alpha_y(\mathrm{H}|-\rangle\otimes\mathrm{G}|\mathrm{R}\rangle) + \beta_x\beta_y(\mathrm{H}|-\rangle\otimes\mathrm{G}|\mathrm{L}\rangle)\\
&= \alpha_x\alpha_y(|0\rangle\otimes|0\rangle) + \alpha_x\beta_y(|0\rangle\otimes|1\rangle) + \beta_x\alpha_y(|1\rangle\otimes|0\rangle) + \beta_x\beta_y(|1\rangle\otimes|1\rangle)\\
&= \alpha_x\alpha_y|00\rangle + \alpha_x\beta_y|01\rangle + \beta_x\alpha_y|10\rangle + \beta_x\beta_y|11\rangle.
\end{aligned}$$

The eigenvalues of Z⊗Z corresponding to the eigenstates $\{|00\rangle,|01\rangle,|10\rangle,|11\rangle\}$ are the same as the eigenvalues of X⊗Y corresponding to the eigenstates $\{|+\mathrm{R}\rangle,|+\mathrm{L}\rangle,|-\mathrm{R}\rangle,|-\mathrm{L}\rangle\}$:

$$
\begin{aligned}
(\mathrm{Z}\otimes\mathrm{Z})\,|00\rangle &= (\mathrm{Z}\otimes\mathrm{Z})\,|0\rangle\otimes|0\rangle = \mathrm{Z}\,|0\rangle\otimes\mathrm{Z}\,|0\rangle = |0\rangle\otimes|0\rangle = |00\rangle\,,\\
(\mathrm{Z}\otimes\mathrm{Z})\,|01\rangle &= (\mathrm{Z}\otimes\mathrm{Z})\,|0\rangle\otimes|1\rangle = \mathrm{Z}\,|0\rangle\otimes\mathrm{Z}\,|1\rangle = |0\rangle\otimes(-|1\rangle) = -|01\rangle\,,\\
(\mathrm{Z}\otimes\mathrm{Z})\,|10\rangle &= (\mathrm{Z}\otimes\mathrm{Z})\,|1\rangle\otimes|0\rangle = \mathrm{Z}\,|1\rangle\otimes\mathrm{Z}\,|0\rangle = (-|1\rangle)\otimes|0\rangle = -|10\rangle\,,\\
(\mathrm{Z}\otimes\mathrm{Z})\,|11\rangle &= (\mathrm{Z}\otimes\mathrm{Z})\,|1\rangle\otimes|1\rangle = \mathrm{Z}\,|1\rangle\otimes\mathrm{Z}\,|1\rangle = (-|1\rangle)\otimes(-|1\rangle) = |11\rangle\,.
\end{aligned}
$$

Therefore, the expectation $\langle\varphi|\mathrm{Z}\otimes\mathrm{Z}|\phi\rangle$, with

$$
|\phi\rangle = \alpha_x\alpha_y\,|00\rangle + \alpha_x\beta_y\,|01\rangle + \beta_x\alpha_y\,|10\rangle + \beta_x\beta_y\,|11\rangle\,,
$$

in the z-basis is given by

$$
|\alpha_x\alpha_y|^2 - |\alpha_x\beta_y|^2 - |\beta_x\alpha_y|^2 + |\beta_x\beta_y|^2.
$$

The values of the probabilities $|\alpha_x\alpha_y|^2$, $|\alpha_x\beta_y|^2$, $|\beta_x\alpha_y|^2$, and $|\beta_x\beta_y|^2$ can be found using quantum computers in exactly the same way we found probabilities in the one-qubit case. By counting the numbers n_{ij} of outcomes $|ij\rangle$ (for $i,j \in \{0,1\}$, with $\sum_{i,j\in\{0,1\}} n_{ij} = N$), the expectation value of $\mathrm{X}\otimes\mathrm{Y}$ is given by

$$
\langle \mathrm{X}\otimes\mathrm{Y}\rangle = \frac{n_{00} - n_{01} - n_{10} + n_{11}}{N}.
$$

10.2.3 The multi-qubit case

It is straightforward to scale this approach to more complex Pauli products and larger Hamiltonians since any Hamiltonian may be written as

$$
\mathcal{H} = \sum_{i\alpha} h^i_\alpha \sigma^i_\alpha + \sum_{ij\alpha\beta} h^{ij}_{\alpha\beta}\sigma^i_\alpha\sigma^j_\beta + \ldots
$$

for real h, where the superscripts $i,j,\ldots$ identify the subsystem (qubit) on which the operator acts, and the subscripts $\alpha,\beta,\ldots$ identify the Pauli operator. For example, $i = 1$, $\alpha = x$, and $\sigma_x{}^1 = \mathrm{X}$ acting on qubit 1. No assumption about the dimension or structure of the Hermitian Hamiltonian is needed for this expansion to be valid [231].

We have already used the linearity of quantum observables that allows us to calculate the expectation of the Hamiltonian as a sum of expectations of the individual terms:

$$\langle \mathcal{H} \rangle = \sum_{i\alpha} h_{\alpha}^{i} \langle \sigma_{\alpha}^{i} \rangle + \sum_{ij\alpha\beta} h_{\alpha\beta}^{ij} \left\langle \sigma_{\alpha}^{i} \sigma_{\beta}^{j} \right\rangle + \ldots$$

As long as we consider Hamiltonians that can be written as a polynomial number of terms with respect to the system size, the evaluation of $\langle \mathcal{H} \rangle$ is reduced to the sum of a polynomial number of expectation values of simple Pauli operators for some quantum state $|\psi\rangle$, multiplied by some real constants. As we have seen above, a quantum computer can efficiently evaluate the expectation value of a tensor product of an arbitrary number of simple Pauli operators [227].



Quantum computers can be used to efficiently calculate expectation values of Hamiltonians consisting of tensor products of Pauli operators. Any Hamiltonian may be represented as a sum of tensor products of Pauli operators (X, Y, Z, and I gates).

10.3 Constructing the PQC

The question of how to construct a high-quality candidate state used to calculate expectations is of fundamental importance. Unless we have some prior knowledge about the ground state and where to search for it in the Hilbert space of the *n*-qubit system, the first task would be to generate a range of candidate states that will cover the whole Hilbert space without being heavily concentrated in any one region. Let us see how this can be done for the single-qubit and multi-qubit systems.

10.3.1 One-qubit ansatz

We return to the Bloch sphere that visualises the possible states of a one-qubit system. Figure 10.1 shows how the qubit state can change from its initial state $|0\rangle$ to the intermediate state $|\psi_i\rangle$ and then to the final state $|\psi_f\rangle$ through a rotation around the *y*-axis followed by a rotation around the *z*-axis.

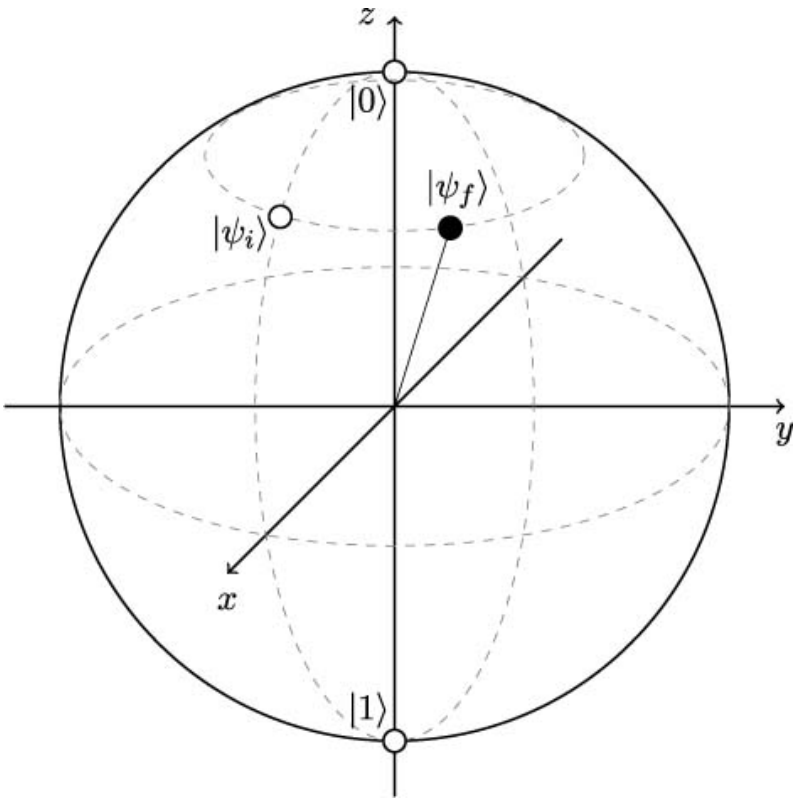


Figure 10.1: Bloch sphere: visualisation of one-qubit rotations.

It is possible to reach any point on the Bloch sphere starting from $|0\rangle$ with just two rotations around any two orthogonal axes. The corresponding circuit is shown in Figure 10.2.

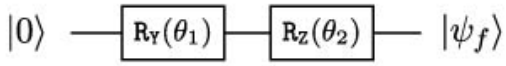


Figure 10.2: PQC for a one-qubit system.

The PQC shown in Figure 10.2 is everything we need in the one-qubit case if we only have the Z and I terms in the problem Hamiltonian. If we want to calculate the expectation value of the X term, we have to add an H gate to the circuit as shown in Figure 10.3.

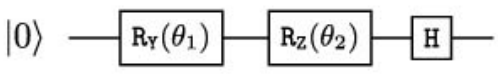


Figure 10.3: PQC with H gate to calculate $\langle X \rangle$.

Similarly, if we want to calculate the expectation value of the Y term, we have to add a G gate to the circuit as shown below:

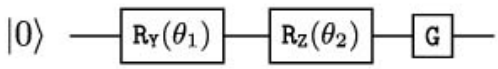


Figure 10.4: PQC with G gate to calculate $\langle \mathrm{Y} \rangle$.

10.3.2 Multi-qubit ansatz

We now look at the multi-qubit case, and assume that the optimisation problem is encoded in the two-qubit Hamiltonian

$$\mathcal{H} = a\mathrm{X} \otimes \mathrm{Y} + b\mathrm{Y} \otimes \mathrm{Z} + c\mathrm{Z} \otimes \mathrm{X},$$

for some coefficients $a,b,c \in \mathbb{R}$. As we know, the expectation value of the Hamiltonian (10.3.2) is given by the sum of expectation values of individual terms:

$$\langle \mathcal{H} \rangle \equiv \langle \psi | \mathcal{H} | \psi \rangle = a \langle \psi | \mathrm{X} \otimes \mathrm{Y} | \psi \rangle + b \langle \psi | \mathrm{Y} \otimes \mathrm{Z} | \psi \rangle + c \langle \psi | \mathrm{Z} \otimes \mathrm{X} | \psi \rangle .$$

We need to calculate all these expectations for the same quantum state $|\psi\rangle$. To do so, we need to construct a quantum circuit with sufficiently flexible adjustable gates to support a wide range of possible candidate states. Since the problem Hamiltonian operates on two-qubit states, the PQC that constructs the candidate states may look like the one shown in Figure 10.5:

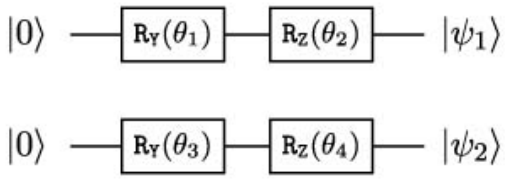


Figure 10.5: PQC for the construction of candidate states.

In the PQC shown in Figure 10.5, the parameters (rotation angles) θ_1 and θ_2 uniquely specify state $|\psi_1\rangle$ of the first qubit, and the parameters θ_3 and θ_4 uniquely specify state $|\psi_2\rangle$ of the second qubit. The full quantum circuit for the calculation of the expectation value $\langle \mathrm{X} \otimes \mathrm{Y} \rangle$ is shown in Figure 10.6, where the gates H (first quantum register) and G (second quantum register) form the

change of basis layer before the measurement in the computational basis. Figures 10.7 and 10.8 display the PQCs for the calculation of $\langle Y \otimes Z \rangle$ and $\langle Z \otimes X \rangle$.

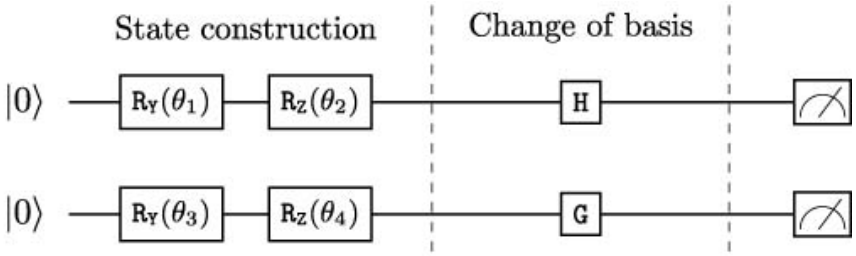


Figure 10.6: PQC for the calculation of the $\langle X \otimes Y \rangle$ term.

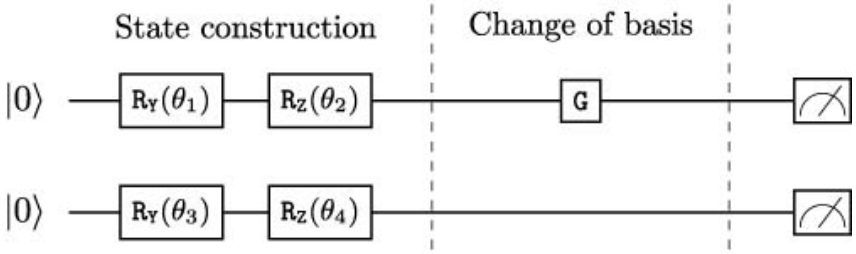


Figure 10.7: PQC for the calculation of the $\langle Y \otimes Z \rangle$ term.

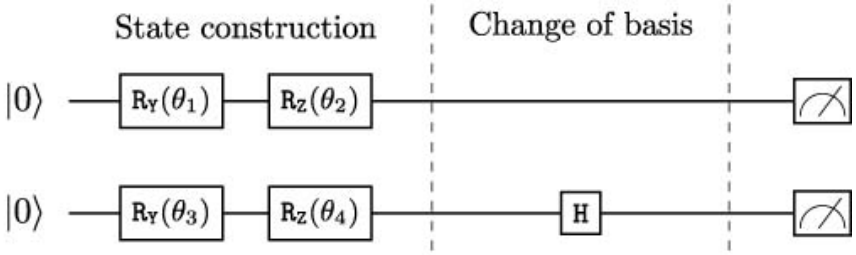


Figure 10.8: PQC for the calculation of the $\langle Z \otimes X \rangle$ term.

Note that the state construction circuits shown in Figures 10.6-10.8 consist of one-qubit gates and do not contain two-qubit gates that would create entanglement. Adding two-qubit gates, such as CNOT and CPHASE, would help explore a wider range of possible quantum states and should be done as a matter of course as we know from the previous chapters. However, our task here is to illustrate the general principle and compare the results obtained by running the PQC with the calculations done "by hand", as we shall see in the next section.

10.4 Running the PQC

We now run some numerical experiments for the optimisation problem encoded in the Hamiltonian (10.3.2) and compare the obtained results with direct calculations to better understand the mechanics of the algorithm and to build intuition.

10.4.1 Experimenting with the two-qubit ansatz

In line with the variational approach and taking into account the PQC architecture specified in Figure 10.5, we need to build the candidate states controlled by the four adjustable parameters θ_1, θ_2, θ_3, and θ_4. In Chapters 8 and 9, we considered two possible approaches to train the PQC: differentiable and non-differentiable learning. Both methods have their strong and weak points and can be used to find an optimal set of adjustable parameters for the PQC of arbitrary size. However, the PQC shown in Figure 10.5 is only two-layer deep and two-quantum register wide – it is perfectly feasible in this case to apply the brute force method.

The brute force method consists of discretising the range of possible values of the rotation angles with the elementary step (increment) kept reasonably small. The parameters θ_1 and θ_3 are rotation angles around the y-axis and are defined on the interval $[0,\pi]$, which we discretise as

$$\left\{\frac{(2k+1)\pi}{2m}\right\}_{k=0,\ldots,m-1},$$

for some integer m. Similarly, θ_2 and θ_4 are rotation angles around the z-axis defined on $[0,2\pi]$, with range

$$\left\{\frac{(2k+1)\pi}{2m}\right\}_{k=0,\ldots,2m-1}.$$

In both cases the increment is π/m. Taking, for example, $m = 8$ is a sensible compromise between speed and accuracy. What is more, once the optimal configuration of adjustable parameters is found, we can run an additional, more granular search in the vicinity of the candidate optimal configuration in order to further improve it.

The algorithm looks as follows. For the given configuration of R_Y and R_Z rotation angles drawn from the sets (10.4.1) and (10.4.1), we execute 100,000 runs of PQCs specified in Figures 10.6, 10.7, and 10.8 on the Qiskit quantum simulator. This gives us the expectation values of X⊗Y, Y⊗Z, and Z⊗X for the quantum state $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$, where

$$|\psi_1\rangle = \cos\left(\frac{\theta_1}{2}\right)|0\rangle + e^{i\theta_2}\sin\left(\frac{\theta_1}{2}\right)|1\rangle$$

and

$$|\psi_2\rangle = \cos\left(\frac{\theta_3}{2}\right)|0\rangle + e^{i\theta_4}\sin\left(\frac{\theta_3}{2}\right)|1\rangle .$$

We select the state $|\psi^*\rangle$ with the smallest value of $\langle\mathcal{H}\rangle$ given by (10.3.2), and denote the corresponding rotation angles $\theta_1^*,\ldots,\theta_4^*$. After that, we perform a more refined search in the neighbourhood of $|\psi^*\rangle$. The new set values of the rotation angles are now

$$\left\{\theta_i^* + \frac{(k-4)\pi}{4m}\right\}_{k=0,\ldots,m}, \quad i = 1,\ldots,4,$$

with increment $\pi/(4m)$. Again, we select the quantum state $|\psi'\rangle$ with the smallest value of $\langle\mathcal{H}\rangle$ and denote the corresponding rotation angles $\theta_1,\ldots,\theta_4$.

With $m = 8$, $a = 4$, $b = 3$, $c = 2$, we obtain the results in Table 10.1:

1st search:	$\theta_1^* = 1.7671$	$\theta_2^* = 3.0434$	$\theta_3^* = 1.7671$	$\theta_4^* = 1.4726$	$\langle\mathcal{H}\rangle_{\min} = -3.93$
2nd search:	$\theta'_1 = 1.5708$	$\theta'_2 = 3.1416$	$\theta'_3 = 1.5708$	$\theta'_4 = 1.5708$	$\langle\mathcal{H}\rangle_{\min} = -4.00$

Table 10.1: Optimal configurations of adjustable PQC parameters that minimise the expectation value of the Hamiltonian.

In Table 10.1, the values of $\langle\mathcal{H}\rangle$ are in the units of coefficients a, b, and c, and the values of rotation angles are in radians. Note that $3.1416 = \pi$ and $1.5708 = \pi/2$. Therefore, the optimal configuration of rotation angles that minimises $\langle\mathcal{H}\rangle$ is

$$\theta_1' = \theta_3' = \theta_4' = \frac{\pi}{2} \quad \text{and} \quad \theta_2' = \pi.$$

The corresponding quantum states are

$$|\psi_1\rangle = \frac{1}{\sqrt{2}}\,|0\rangle - \frac{1}{\sqrt{2}}\,|1\rangle\,, \quad |\psi_2\rangle = \frac{1}{\sqrt{2}}\,|0\rangle + \frac{\mathrm{i}}{\sqrt{2}}\,|1\rangle\,,$$
$$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle = \frac{1}{2}\,|00\rangle + \frac{\mathrm{i}}{2}\,|01\rangle - \frac{1}{2}\,|10\rangle - \frac{\mathrm{i}}{2}\,|11\rangle\,.$$

10.4.2 Analysis of the obtained results

Do the obtained results make sense? Since the problem size is small and the circuit is not too deep, we can verify the results by direct manual calculations. First, we visualise the states $|\psi_1\rangle$ and $|\psi_2\rangle$. Figure 10.4.2 shows their positions on the Bloch sphere. State $|\psi_1\rangle$ is the black dot at the intersection of the x-axis and the equator. We get to $|\psi_1\rangle$ from state $|0\rangle$ by performing a $\pi/2$ radian rotation around the y-axis and then a π radian rotation around the z-axis. State $|\psi_2\rangle$ is the gray dot at the intersection of the y-axis and the equator, which is reached from $|0\rangle$ by performing a $\pi/2$ radian rotation around the y-axis and then a $\pi/2$ radian rotation around the z-axis.

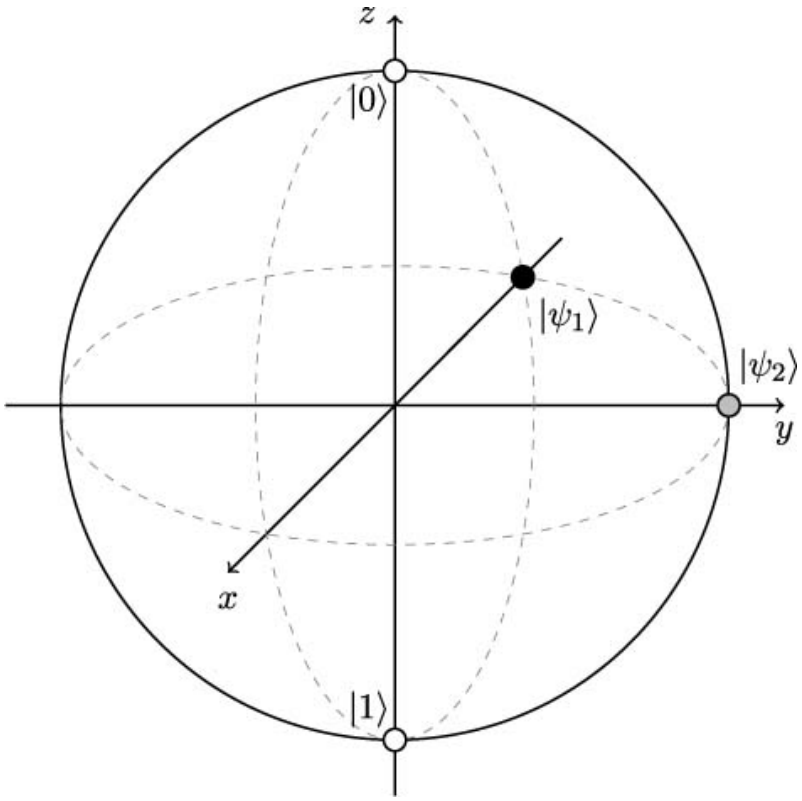


Figure 10.9: Visualisation of states $|\psi_1\rangle$ (black dot) and $|\psi_2\rangle$ (grey dot).

Consider the expectation $\langle \mathtt{X} \otimes \mathtt{Y} \rangle$. The operator X acts on $|\psi_1\rangle$, which is one of its eigenstates in the *x*-basis:

$$|\psi_1\rangle = \frac{1}{\sqrt{2}} |0\rangle - \frac{1}{\sqrt{2}} |1\rangle = |-\rangle ,$$

and Y acts on $|\psi_2\rangle$, which is also one of its eigenstates in the *y*-basis:

$$|\psi_2\rangle = \frac{1}{\sqrt{2}} |0\rangle + \frac{\mathrm{i}}{\sqrt{2}} |1\rangle = |R\rangle .$$

We then perform measurements in the computational basis (*z*-basis). Before measurement, the Hadamard operator H will transform $|\psi_1\rangle = |-\rangle$ into $|1\rangle$, which is a basis state in the *z*-basis. Similarly, the operator G will transform $|\psi_2\rangle = |R\rangle$ into $|0\rangle$, which is also a basis state in the *z*-basis.

Thus, measurement in the *z*-basis will give us the state $|1\rangle \otimes |0\rangle = |10\rangle$ with probability one. If we perform N measurements, we will obtain the state $|10\rangle$ N times and the corresponding expectation value $\langle \mathtt{X} \otimes \mathtt{Y} \rangle$ will be

$$\langle \mathtt{X} \otimes \mathtt{Y} \rangle = \frac{n_{00} - n_{01} - n_{10} + n_{11}}{N} = \frac{0 - 0 - N + 0}{N} = -1.$$

The value of the coefficient *a* in front of the X⊗Y term was set equal to 4. Therefore, the contribution of the first term to the expectation value of the Hamiltonian (10.3.2) is −4.

The expectations $\langle \mathtt{Y} \otimes \mathtt{Z} \rangle$ and $\langle \mathtt{Z} \otimes \mathtt{X} \rangle$ are equally straightforward to calculate. Let us start with $\langle \mathtt{Y} \otimes \mathtt{Z} \rangle$. The operator Y acts on the first qubit in the state $|\psi_1\rangle$. Since we measure the resulting state in the *z*-basis, we need to apply G before measurement. The operator G transforms $|\psi_1\rangle$ into the state

$$\frac{1+\mathrm{i}}{2} |0\rangle + \frac{1-\mathrm{i}}{2} |1\rangle .$$

Measurement in the *z*-basis will result in $|0\rangle$ and $|1\rangle$ with equal probability 1⁄2.

The operator Z acts on the second qubit in state $|\psi_2\rangle$. Since we measure the operator Z in the *z*-basis, we do not need to apply any gates. Measuring $|\psi_2\rangle$ in

the z-basis will also result in obtaining $|0\rangle$ and $|1\rangle$ with equal probability 1⁄2. Therefore, we are equally likely to measure all four basis states ($|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$) with probability 1⁄4, and, as N tends to infinity, the expectation value of $\langle \mathrm{Y} \otimes \mathrm{Z} \rangle$ in our experiment should converge to

$$\langle \mathrm{Y} \otimes \mathrm{Z} \rangle = \frac{n_{00} - n_{01} - n_{10} + n_{11}}{N} = \frac{\frac{1}{4}N - \frac{1}{4}N - \frac{1}{4}N + \frac{1}{4}N}{N} = 0.$$

We are left now with $\langle \mathrm{Z} \otimes \mathrm{X} \rangle$. The operator Z acts on $|\psi_1\rangle$. The measurement is done in the z-basis, so no transformation is needed. The measurement will result in obtaining the states $|0\rangle$ and $|1\rangle$ with equal probability 1⁄2. The operator X acts on the second qubit in the state $|\psi_2\rangle$. The measurement is performed in the z-basis, so we need to apply the Hadamard gate H before measurement. Applying the H gate to $|\psi_2\rangle$ transforms it into

$$\frac{1+\mathrm{i}}{2}|0\rangle + \frac{1-\mathrm{i}}{2}|1\rangle,$$

with measurement in the z-basis producing outcomes $|0\rangle$ and $|1\rangle$ with probability 1⁄2. Thus, we find ourselves in the same situation as with $\langle \mathrm{Y} \otimes \mathrm{Z} \rangle$: all basis states are equally likely. As N tends to infinity, the expectation value of $\langle \mathrm{Z} \otimes \mathrm{X} \rangle$ should also converge to

$$\langle \mathrm{Z} \otimes \mathrm{X} \rangle = \frac{n_{00} - n_{01} - n_{10} + n_{11}}{N} = \frac{\frac{1}{4}N - \frac{1}{4}N - \frac{1}{4}N + \frac{1}{4}N}{N} = 0.$$

This is exactly what we observed in our numerical experiment. The total contribution of all three terms of the Hamiltonian $\mathcal{H}$ given by (10.3.2) with a = 4, b = 3, and c = 2 is equal to −4. The X⊗Y term has the largest coefficient, therefore it makes sense that the ground state of $\mathcal{H}$ is the one that minimises the expectation value $\langle \mathrm{X} \otimes \mathrm{Y} \rangle$ (with expectation values $\langle \mathrm{Y} \otimes \mathrm{Z} \rangle$ and $\langle \mathrm{Z} \otimes \mathrm{X} \rangle$ being zero).



PQC can be used to construct the candidate states for the VQE algorithm. The selection and improvement of the candidate states is performed classically. This makes VQE a perfect example of a hybrid quantum-classical algorithm.

10.5 Discrete Portfolio Optimisation with VQE

In Chapter 3, we investigated quantum annealing for NP-hard discrete portfolio optimisation problems. The same type of QUBO problems can be solved on gate model quantum computers with the help of a hybrid VQE algorithm. The QUBO formulation of the discrete portfolio optimisation problem consists of minimising the cost function (3.3.1):

$$L(\mathrm{q}) = \sum_{i=1}^{N} a_i q_i + \sum_{i=1}^{N} \sum_{j=i+1}^{N} b_{ij} q_i q_j,$$

where q := (q_1,…,q_N) is a vector of binary decision variables indicating which (equally weighted) assets are selected (from the universe of N investable assets): $q_i = 1$ means that asset i is selected and $q_i = 0$ means that asset i is not selected. The task is to find a configuration of q that minimises L(q).

For each $i,j = 1,\ldots N$, the coefficients a_i, a_j, and b_{ij} reflect, respectively, the individual and joint attractiveness of assets i and j. For example, assets with larger expected returns and lower volatilities would be assigned large negative values of a. Similarly, pairs of assets with low positive or even negative correlation would be assigned negative values of b to reward diversification. Assets with lower expected returns, higher volatility, and strongly positively correlated with other assets would be penalised with positive values of a and b.

Quantum annealers solve QUBO problems in their Ising model formulation where binary decision variables q := (q_1,…,q_N) are translated into spin variables s := (s_1,…,s_N) taking values {+1,−1} through the transformation $s_i = 2q_i - 1$. We analyse the simplest case of an investable universe consisting of just two assets. In this case, the QUBO cost function reads

$$L(\mathrm{q}) = a_1 q_1 + a_2 q_2 + b_{12} q_1 q_2,$$

or, in the Ising model formulation,

$$L(\mathrm{s}) = g_1 s_1 + g_2 s_2 + J_{12} s_1 s_2 + \mathrm{const},$$

where

$$g_1 = \frac{1}{2}a_1 + \frac{1}{4}b_{12}, \quad g_2 = \frac{1}{2}a_2 + \frac{1}{4}b_{12}, \quad J_{12} = \frac{1}{4}b_{12}.$$

The constant term in (10.5) does not depend on the decision variables s_1 and s_2 and can hence be ignored. The cost function we want to minimise thus becomes

$$L(\mathrm{s}) = g_1 s_1 + g_2 s_2 + J_{12} s_1 s_2.$$

As we know from Chapter 2, transition from the classical to the quantum mechanical description of the system consists of replacing variables corresponding to physical observables by their respective operators. In the case of the QUBO problem in its Ising model formulation, it means replacing classical spin variables with the corresponding Pauli operators σ_x, σ_y, and σ_z, which are represented in the quantum circuit by, respectively, the quantum gates X, Y, and Z.

As we remember from Chapter 3, the Ising cost function (10.5) corresponds to the following *final* Hamiltonian that encodes the same optimisation problem:

$$\mathcal{H}_F = g_1 \sigma_z^1 + g_2 \sigma_z^2 + J_{12} \sigma_z^1 \sigma_z^2.$$

Here, the classical spin variables s_1 and s_2 are replaced by the σ_z operators, and $\sigma_z{}^1$ is the Z gate acting on qubit 1 while $\sigma_z{}^2$ is the Z gate acting on qubit 2.

Note that the mapping between the binary QUBO decision variables q, classical spin variables s, and the eigenstates of Z is as follows:

$$q = 0 \rightarrow s = -1 \rightarrow |1\rangle \quad \text{since} \quad \mathrm{Z}\,|1\rangle = -\,|1\rangle\,,$$

and

$$q = 1 \rightarrow s = +1 \rightarrow |0\rangle \quad \text{since} \quad \mathrm{Z}\,|0\rangle = |0\rangle\,.$$

We now solve the QUBO problem (10.5) with

$$a_1 = -2, \quad a_2 = 3, \quad b_{12} = -2,$$

using both the classical method (in this case a simple exhaustive search given that the solution space consists of just four possible solutions) and the VQE approach.

The classical exhaustive search results are straightforward to obtain via direct calculations and are summarised in Table 10.2 with the optimal solution $q^* = (1,0)$: asset 1 is selected, while asset 2 is not.

q_1	q_2	L(q)
0	0	0
0	1	3
1	0	-2
1	1	-1

Table 10.2: Classical exhaustive search results.

The VQE calculations are as follows. We first rewrite the Hamiltonian (10.5) in the quantum gate form:

$$\mathcal{H}_F = g_1 \mathrm{Z}^1 + g_2 \mathrm{Z}^2 + J_{12} \mathrm{Z}^1 \otimes \mathrm{Z}^2,$$

where Z^1 is the Z gate acting on qubit 1, Z^2 is the Z gate acting on qubit 2, and $Z^1 \otimes Z^2$ is the tensor product of the Z gates acting on qubits 1 and 2 respectively.

The next step is to calculate the expectation values $\langle \mathrm{Z} \rangle$ and $\langle \mathrm{Z} \otimes \mathrm{Z} \rangle$. As we know, Z is a PHASE gate that flips the phase of a qubit:

$$\mathrm{Z} \left|0\right\rangle = \left|0\right\rangle, \quad \mathrm{Z} \left|1\right\rangle = - \left|1\right\rangle.$$

Therefore, we have

$$\begin{aligned}
\langle 0| \mathrm{Z}^1 |0\rangle &= +1, & \langle 1| \mathrm{Z}^1 |1\rangle &= -1, \\
\langle 0| \mathrm{Z}^2 |0\rangle &= +1, & \langle 1| \mathrm{Z}^2 |1\rangle &= -1, \\
\langle 00| \mathrm{Z}^1 \otimes \mathrm{Z}^2 |00\rangle &= +1, & \langle 01| \mathrm{Z}^1 \otimes \mathrm{Z}^2 |01\rangle &= -1, \\
\langle 10| \mathrm{Z}^1 \otimes \mathrm{Z}^2 |10\rangle &= -1, & \langle 11| \mathrm{Z}^1 \otimes \mathrm{Z}^2 |11\rangle &= +1.
\end{aligned}$$

Here we calculated the expectation values of operators Z and Z⊗Z analytically but we would obtain exactly the same values if we were to calculate them using a quantum computer as described in Section 10.2. Now we need to calculate the values of the coefficients g_1, g_2, and J_{12} using the transformation (10.5):

$$g_1 = -1.5, \quad g_2 = 1, \quad J_{12} = -0.5.$$

The expectation values of $\mathcal{H}_F$ in states $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$ can then be calculated as

$$\begin{aligned}
|00\rangle: \quad \langle\mathcal{H}\rangle &= g_1 \cdot (+1) + g_2 \cdot (+1) + J_{12} \cdot (+1) = -1, \\
|01\rangle: \quad \langle\mathcal{H}\rangle &= g_1 \cdot (+1) + g_2 \cdot (-1) + J_{12} \cdot (-1) = -2, \\
|10\rangle: \quad \langle\mathcal{H}\rangle &= g_1 \cdot (-1) + g_2 \cdot (+1) + J_{12} \cdot (-1) = 3, \\
|11\rangle: \quad \langle\mathcal{H}\rangle &= g_1 \cdot (-1) + g_2 \cdot (-1) + J_{12} \cdot (+1) = 0.
\end{aligned}$$

The best solution found using VQE is therefore $|01\rangle = |0\rangle \otimes |1\rangle$, i.e., asset 1 is selected while asset 2 is not, which is the same as the best solution found by exhaustive search.



A VQE is a viable tool for solving finance-related NP-hard optimisation problems.

Summary

In this chapter, we introduced the Variational Quantum Eigensolver algorithm, a powerful QML model based on the variational approach that allows us to solve hard optimisation problems. We learned how to calculate expectation values using a quantum computer and how to construct a PQC, creating the candidate quantum states.

We also experimented with running the VQE model on a sample problem encoded in a two-qubit Hamiltonian and analysed and verified the results by performing manual calculations. Finally, we demonstrated the applicability of

the VQE to finance-related optimisation problems, in particular to a stylised example of a discrete portfolio optimisation.

In the next chapter, we will introduce the Quantum Approximate Optimisation Algorithm, another example of a hybrid quantum-classical approach to solving hard optimisation problems.

Join our book's Discord space

Join our Discord community to meet like-minded people and learn alongside more than 2000 members at: https://packt.link/quantum



11
Quantum Approximate Optimisation Algorithm

As the name suggests, the Quantum Approximate Optimisation Algorithm (QAOA) is an optimisation algorithm. It is motivated by and draws upon two optimisation algorithms considered in previous chapters: AQC and VQE. From AQC it borrows the concept of solving an optimisation problem through encoding the corresponding objective function in the problem Hamiltonian and then evolving the system in such a way that the ground state of the final Hamiltonian provides the solution we are after (in a bitstring format). From VQE it borrows the variational principle applied to the parameterised quantum circuit. Roughly speaking, QAOA is a gate-model version of an optimisation solver that otherwise could have been tackled with an analog AQC approach. We can also look at QAOA as a special case of VQE with the constraints on the form of the Hamiltonian.

QAOA was introduced in the pioneering work by Farhi, Goldstone, and Gutmann [96] in 2014 and its potential for establishing quantum supremacy was investigated by Farhi and Harrow in [99]. QAOA and its sister algorithm that generalises it, the Quantum Alternating Operator Ansatz (with the same acronym!), have been tested on a number of financial use cases. Here we can mention the work by Hodson, Ruck, Ong, Garvin, and Dulma [138] on portfolio rebalancing experiments and the one by Barkoutsos, Nannicini, Robert, Tavernelli, and Woerner [26] using Conditional Value-at-Risk (CVaR) as the QAOA objective function. The algorithm has significant potential and promises to become a standard tool in the arsenal of quantum computing methods aimed at financial applications.

11.1 Time Evolution

Consider again the description of the dynamics of quantum mechanical systems, briefly covered in Chapter 1 (as one of the postulates of quantum mechanics) and Chapter 2 (where we introduced the principles of Adiabatic Quantum Computing). These dynamics are governed by the Schrödinger equation (1.2.2):

$$\mathrm{i}\hbar\frac{\mathrm{d}\,|\psi(t)\rangle}{\mathrm{d}t} = \mathcal{H}\,|\psi(t)\rangle\,,$$

with some initial condition $|\psi(0)\rangle$, where $|\psi(t)\rangle$ is the quantum state at time t and $\mathcal{H}$ is the time-independent Hamiltonian. Its solution is given by (1.2.2), namely

$$|\psi(t)\rangle = \mathcal{U}(0,t)\,|\psi(0)\rangle\,,$$

where the operator $\mathcal{U}(0,t)$ is obtained from the Hamiltonian $\mathcal{H}$ by (1.2.2):

$$\mathcal{U}(0,t) = \exp\left(-\frac{\mathrm{i}\mathcal{H}t}{\hbar}\right).$$

We work with units where $\hbar$ is set to 1, so that the system dynamics reads

$$|\psi(t)\rangle = \mathrm{e}^{-\mathrm{i}\mathcal{H}t}\,|\psi(0)\rangle\,.$$

If the initial state of the system $|\psi(0)\rangle$ is known then the state of the system at time t is also known and is determined by the action of the Hamiltonian $\mathcal{H}$ over the period of time t.

However, the solution (11.1) assumes that the system Hamiltonian is time-independent. At the same time, AQC works with time-dependent Hamiltonians of the form (2.2):

$$\mathcal{H}(t) = \left(1 - \frac{t}{T}\right)\mathcal{H}_0 + \frac{t}{T}\mathcal{H}_F,$$

for some *initial* Hamiltonian $\mathcal{H}_0$ and some *final* or *problem* (encoding the optimisation problem) Hamiltonian $\mathcal{H}_F$. How do we reconcile this mismatch? The answer is that we can *approximate* [272] the time-dependent Hamiltonian $\mathcal{H}(t)$ that transforms the state over $[0,T]$ by a sequence of time-independent Hamiltonians:

$$\mathcal{H}_1,\ \mathcal{H}_2,\ \ldots,\ \mathcal{H}_m,$$

transforming the state over the corresponding shorter time intervals:

$$[t_0 = 0, t_1],\ [t_1, t_2],\ \ldots,\ [t_{m-1}, t_m = T].$$

A good analogy is the approximation of a continuous function (e.g., $\sin(\cdot)$) by a piecewise linear function as shown in Figure 11.1. The more granular the time intervals $[t_{i-1}, t_i]$, the better the approximation.

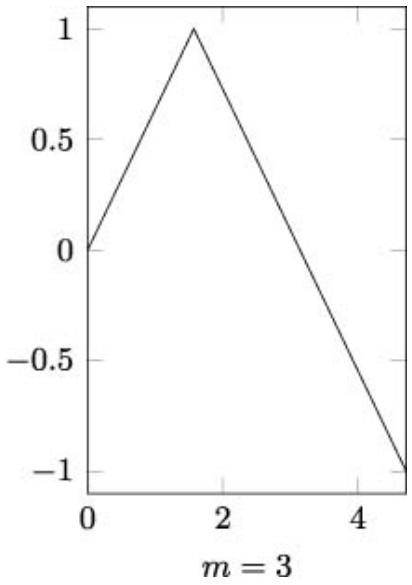


Figure 11.1: Piecewise linear approximation of t↦sin(t).

Similarly, we can approximate the operator $\mathcal{U}(0,T)$ as

$$\mathcal{U}(0,T) \approx \mathcal{U}(t_{m-1}, t_m)\mathcal{U}(t_{m-2}, t_{m-1}) \cdots \mathcal{U}(t_2, t_1)\mathcal{U}(t_0, t_1).$$

Again, larger values of n give a better approximation.



The evolution of a quantum mechanical system over a long time interval can be modelled as a sequence of time-independent Hamiltonians transforming the system state over the corresponding shorter time intervals.

11.2 The Suzuki-Trotter Expansion

A particularly useful approximation of $\mathcal{U}(0,T)$ can be obtained using the Suzuki-Trotter expansion [279]. If $\mathcal{A}_1,\mathcal{A}_2,\ldots,\mathcal{A}_k$ are operators that do not necessarily commute, then

$$\exp\left(\mathcal{A}_1+\mathcal{A}_2+\ldots+\mathcal{A}_k\right)=\lim_{m\to\infty}\left[\exp\left(\frac{\mathcal{A}_1}{m}\right)\exp\left(\frac{\mathcal{A}_2}{m}\right)\cdots\exp\left(\frac{\mathcal{A}_k}{m}\right)\right]^m.$$

Recall that two operators $\mathcal{A}$ and $\mathcal{B}$ are said to commute if $\mathcal{AB}=\mathcal{BA}$. Many operators introduced in previous chapters do not commute, for example, rotations around different axes do not, and the end result (the end quantum state) depends on how rotations are ordered.

As mentioned in Chapter 1, the expectation values of Hermitian operators are real and correspond to physical observables (e.g., the expectation of a Hermitian Hamiltonian is the physically observable energy). If operators commute we can *measure* them in an arbitrary order and obtain the same answer. There is no uncertainty in the values of the corresponding physical observables.

The Suzuki-Trotter expansion, however, does not require operators to commute to remain valid. This has important implications for QAOA as we shall see below. If $\mathcal{U}(0,T)$ has the form $\exp([\mathcal{A}+\mathcal{B}]T)$ then we can use the Suzuki-Trotter expansion to obtain

$$\exp([\mathcal{A}+\mathcal{B}]T) = \lim_{m\to\infty}\left[\exp\left(\frac{\mathcal{A}T}{m}\right)\exp\left(\frac{\mathcal{B}T}{m}\right)\right]^m,$$

namely the time evolution of $[\mathcal{A}+\mathcal{B}]T$ can be approximated by applying alternatively $\mathcal{A}$ and $\mathcal{B}$ over time intervals of length T/m.

In Chapter 3, we introduced detailed specifications of the AQC, where the Hamiltonians $\mathcal{H}_0$ and $\mathcal{H}_F$ have the general form

$$\mathcal{H}_0 = \sum_{i=1}^{n}\sigma_x^i \quad \text{and} \quad \mathcal{H}_F = \sum_{i=1}^{n} a_i\sigma_z^i + \sum_{i=1}^{n}\sum_{j=i+1}^{n} b_{ij}\sigma_z^i\sigma_z^j,$$

for some coefficients $(a_i)_{i=1,\ldots,n}$ and $(b_{ij})_{i,j=1,\ldots,n}$. We also refer the interested reader to [304] for a precise connection between QAOA and AQC, in particular in the case of a finite number of steps m.

The initial Hamiltonian $\mathcal{H}_0$ is the operator $\mathcal{A}$ – called the *mixing* Hamiltonian – and the final Hamiltonian $\mathcal{H}_F$ is the operator $\mathcal{B}$ – called the *phase* Hamiltonian. Recall from Chapter 10 that the Pauli operators σ_x, σ_y, and σ_z are represented by the quantum gates X, Y, and Z, and the superscript in σ_x^i refers to the qubit on which it acts.

The initial state is set in the equal superposition state of all possible solutions [122]:

$$|\psi(0)\rangle = \frac{1}{\sqrt{2^n}}\Big(|0\ldots00\rangle + |0\ldots01\rangle + \ldots + |1\ldots11\rangle\Big) = \frac{1}{\sqrt{2^n}}\sum_{i=0}^{2^n-1}|i\rangle,$$

which is the ground state of $\mathcal{A}$ and can be easily constructed from $|0\rangle^{\otimes n}$ by applying the Hadamard gate H to every qubit, i.e.,

$$|\psi(0)\rangle = \mathrm{H}^{\otimes n}|0\rangle^{\otimes n}.$$

Remark: For a one-qubit system, $\mathcal{A}$ is the X gate and the initial state is

$$|\psi(0)\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle\,.$$

As we know from Chapter 10, state $|+\rangle$ is the eigenstate of X with eigenvalue equal to 1, namely $X|+\rangle = |+\rangle$.

11.3 The Algorithm Specification

Now everything is in place for the formulation of the QAOA procedure [122].

Algorithm 9: Quantum Approximate Optimisation Algorithm

Input: $\mathcal{A}$ and $\mathcal{B}$.

1: A parameterised quantum state $|\psi(\boldsymbol{\beta},\boldsymbol{\gamma})\rangle$, $\boldsymbol{\beta} := (\beta_1,\ldots,\beta_m)$, $\boldsymbol{\gamma} := (\gamma_1,\ldots,\gamma_m)$, is created by alternately applying the operators $\mathcal{A}$ and $\mathcal{B}$ for m rounds, where the duration in round i $(i = 1,\ldots,m)$ is specified by the parameters β_i and γ_i respectively:

$$|\psi(\boldsymbol{\beta},\boldsymbol{\gamma})\rangle = \mathrm{e}^{-\mathrm{i}\beta_m\mathcal{A}}\mathrm{e}^{-\mathrm{i}\gamma_m\mathcal{B}}\cdots\mathrm{e}^{-\mathrm{i}\beta_2\mathcal{A}}\mathrm{e}^{-\mathrm{i}\gamma_2\mathcal{B}}\mathrm{e}^{-\mathrm{i}\beta_1\mathcal{A}}\mathrm{e}^{-\mathrm{i}\gamma_1\mathcal{B}}\left(\mathrm{H}^{\otimes n}|0\rangle^{\otimes n}\right).$$

2: A computational basis (z-basis) measurement is performed on the obtained state, which returns a candidate solution. Repeating the above state preparation and measurement, the expected value of the cost function f over the returned solution samples is given by

$$\langle f\rangle = \langle\psi(\boldsymbol{\beta},\boldsymbol{\gamma})|\,\mathcal{B}\,|\psi(\boldsymbol{\beta},\boldsymbol{\gamma})\rangle,$$

which can be statistically estimated from the samples produced (as explained in Chapter 10).

3: The above steps may then be repeated with the updated sets of time parameters $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ – the variational part of the algorithm – within the classical optimisation loop that aims to minimise the expectation of the cost function $\langle f\rangle$.

Result: The algorithm returns the best found solution.

It is important to apply operators $\exp(-\mathrm{i}\beta\mathcal{A})$ and $\exp(-\mathrm{i}\gamma\mathcal{B})$ alternately to ensure that we are not trapped in a local minimum. It is also important that $\mathcal{A}$ and $\mathcal{B}$ do not commute [272]. Indeed, by applying only $\exp(-\mathrm{i}\gamma\mathcal{B})$, we are facing ending in an eigenstate of the phase Hamiltonian. If this happens, we will be trapped there: any further application of a linear operator to its eigenvector may change its length but not its direction. The same applies to alternating between two commuting operators: if $\mathcal{A}$ and $\mathcal{B}$ commute, then we can come up with a set of basis states that are eigenstates of both $\mathcal{A}$ and $\mathcal{B}$, and once we

get into one of these eigenstates we remain trapped in it. However, since σ_x and σ_z do not commute, there is always a chance to escape from the local minimum.

The ongoing exploration of QAOA potential started with the foundational paper by Farhi, Goldstone, and Gutmann [96] where it outperformed classical algorithms on the Max-Cut problem on connected 3-regular graphs. A regular graph is one where each vertex has the same number of neighbours. In the case of a 3-regular graph (also known as a *cubic* graph), each vertex is connected with three other vertices. We consider the Max-Cut problem in the next section in its most general formulation. It was a decisive result that prompted active development of classical algorithms and, eventually, one with asymptotically much better performance was constructed by Barak *et al* [25]. This, in turn, triggered further investigations: the performance comparison between QAOA and the best classical algorithms was studied by Hastings in [128], and Bravyi, Kliesch, Koenig and Tang [44] established that the locality and the symmetry of QAOA severely limit its performance. To overcome these limitations, they proposed a non-local version that significantly outperforms standard QAOA for a frustrated Ising model on random 3-regular graphs.

11.4 The Max-Cut Problem

The Max-Cut problem is one of the special cases of the graph partitioning problem introduced in Chapter 3. The objective is to divide the vertices of the graph into two groups such that either the maximum possible number of edges going between the two groups are "cut" (if all edges have the same weight) or the total weight of these edges is maximised (if they have different weights).

The problem of maximisation of the total weight (or number of edges being cut) can be formulated as the minimisation of a cost function, which is the sum of the costs of all individual edges. Each individual cost c_{ij}, associated with the edge connecting vertices i and j, is given by

$$c_{ij} = \frac{1}{2} w_{ij}(1 - s_i s_j),$$

where s_i and s_j are classical spin variables taking values {−1,+1} and w_{ij} is the weight associated with the edge connecting vertices i and j. The two groups of vertices are those where the spin variables take the same values (either −1 or +1). We can see from (11.4) that when s_i and s_j have the same sign, the cost c_{ij} is zero; however, when s_i and s_j have opposite signs, the cost c_{ij} is equal to the weight w_{ij}.

The cost function for the whole graph then has the form

$$L(\mathrm{s}) = \sum_{\{ij\} \in G} \frac{1}{2} w_{ij}(1 - s_i s_j),$$

where s := (s_1,…,s_n) is the set of decision variables associated with the n-node graph G and the sum goes over all pairs of nodes connected by the graph edges.

There are many possible applications of the Max-Cut problem in finance, for example *client clustering* or *client segmentation*, where the task can be formulated by creating a graph containing a node for each client and an edge between each pair of clients. The weight of an edge connecting any two clients is determined by the relative closeness of clients' characteristics: the closer the clients, the smaller the weight of the edge that connects them. The clusters that are formed by finding maximum weight cuts have the property that clients in one cluster are more dissimilar from clients in other clusters.

However, the flagship application of Max-Cut in finance is portfolio optimisation. Dees, Stanković, Constantinides, and Mandi [81] have shown that the graph-theoretic portfolio partitioning technique can help devise robust and tractable asset allocation schemes by virtue of a rigorous graph framework for considering smaller, computationally feasible, and economically meaningful clusters of assets, based on graph cuts. Barkoutsos, Nannicini, Robert, Tavernelli, and Woerner [26] improved variational quantum optimisation using a Conditional Value-at-Risk technique – ubiquitous in financial risk management. A portfolio optimisation QAOA use case addressed by solving the maximum independent set problem on a quantum simulator was presented in [276]. It is also necessary to mention portfolio rebalancing experiments using the Quantum Alternating Operator Ansatz conducted by Hodson, Ruck, Ong, Garvin, and Dulma [138].

11.4.1 QAOA gates

The mixing Hamiltonian $\mathcal{A}$ and the phase Hamiltonian $\mathcal{B}$ corresponding to the cost function (11.4) read

$$\mathcal{A} = \sum_{i=1}^{n} \sigma_x^i \quad \text{and} \quad \mathcal{B} = \sum_{\{ij\}\in G} \frac{1}{2} w_{ij} \left(1 - \sigma_z^i \sigma_z^j\right),$$

where the spin variables s are replaced by the corresponding Pauli operators σ. We therefore need to find the quantum gate representation of the operators

$$\exp\left(-\mathrm{i}\beta\sigma_x^i\right) \quad \text{and} \quad \exp\left(-\frac{1}{2}\mathrm{i}\gamma\sigma_z^i\sigma_z^j\right).$$

To do so, we require the following (see also Lemma 1, albeit with a slightly different proof):

Theorem 9. *With $\mathcal{I}$ denoting the identity operator, the following holds for any unitary Hermitian operator $\mathcal{H}$ and any $\theta \in \mathbb{R}$:*

$$R_\theta(\mathcal{H}) \equiv \exp\left(-\frac{1}{2}\mathrm{i}\theta\mathcal{H}\right) = \cos\left(\frac{\theta}{2}\right)\mathcal{I} - \mathrm{i}\sin\left(\frac{\theta}{2}\right)\mathcal{H}.$$

Proof. Since $\mathcal{H}$ is a unitary Hermitian operator its eigenvalues are +1 and −1 (Chapter 1). Let $\mathcal{P}_\pm$ be projectors onto the eigenspace of eigenvalues ±1 respectively, so that

$$\mathcal{I} = \mathcal{P}_+ + \mathcal{P}_-, \quad \mathcal{H} = \mathcal{P}_+ - \mathcal{P}_-.$$

When a function (in this case R_θ) is applied to a matrix (in this case $\mathcal{H}$) it is applied to each of the eigenvalues:

$$\begin{aligned} R_\theta(\mathcal{H}) &= \exp\left(-\frac{1}{2}\mathrm{i}\theta(+1)\right)\mathcal{P}_+ + \exp\left(-\frac{1}{2}\mathrm{i}\theta(-1)\right)\mathcal{P}_- \\ &= \exp\left(-\frac{1}{2}\mathrm{i}\theta\right)\mathcal{P}_+ + \exp\left(\frac{1}{2}\mathrm{i}\theta\right)\mathcal{P}_-. \end{aligned}$$

From (11.4.1) we have

$$\mathcal{P}_+ = \frac{1}{2}(\mathcal{I} + \mathcal{H}) \quad \text{and} \quad \mathcal{P}_- = \frac{1}{2}(\mathcal{I} - \mathcal{H}).$$

Substituting (11.4.1) into (11.4.1) yields

$$\begin{aligned} R_\theta(\mathcal{H}) &= \frac{1}{2}\exp\left(-\frac{1}{2}\mathrm{i}\theta\right)_{(\mathcal{I}+\mathcal{H})} + \frac{1}{2}\exp\left(\frac{1}{2}\mathrm{i}\theta\right)_{(\mathcal{I}-\mathcal{H})} \\ &= \frac{1}{2}\left[\exp\left(-\frac{1}{2}\mathrm{i}\theta\right) + \exp\left(\frac{1}{2}\mathrm{i}\theta\right)\right]_{\mathcal{I}} + \frac{1}{2} \\ &\left[\exp\left(-\frac{1}{2}\mathrm{i}\theta\right) - \exp\left(\frac{1}{2}\mathrm{i}\theta\right)\right]_{\mathcal{H}} \end{aligned}$$

$$= \cos\left(\frac{\theta}{2}\right)_{\mathcal{G}} - i\sin\left(\frac{\theta}{2}\right)_{\mathcal{H}}.$$

□

We can use (9) to write down expressions for the operators (11.4.1) in matrix form. We start with the first operator:

$$\begin{aligned}\exp(-i\beta\sigma_x) &= \cos(\beta)\mathtt{I} - i\sin(\beta)\mathtt{X} \\ &= \cos(\beta)\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} - i\sin(\beta)\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix} \\ &= \begin{bmatrix}\cos(\beta) & -i\sin(\beta)\\ -i\sin(\beta) & \cos(\beta)\end{bmatrix}.\end{aligned}$$

Since the operator $R_X(\theta)$ has the following matrix representation

$$\mathtt{R_X}(\theta) = \begin{bmatrix}\cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right)\\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right)\end{bmatrix},$$

(see, for example, Chapter 6 or Theorem 9), we have

$$\exp(-i\beta\sigma_x) = \mathtt{R_X}(2\beta),$$

so that opeator $\exp(-i\beta\sigma_x^i)$ should be represented in the circuit by the gate $R_X(2\beta)$ placed on the quantum register i.

We can proceed now with the gate representation of operator $\exp\left(-\frac{1}{2}i\gamma\sigma_z\sigma_z\right)$, where $\sigma_z\sigma_z$ represents the tensor product of two σ_z operators acting on two different qubits:

$$\exp\left(-\frac{1}{2}\mathrm{i}\gamma\sigma_z\sigma_z\right) = \cos\left(\frac{\gamma}{2}\right)\mathtt{I}\otimes\mathtt{I} - \mathrm{i}\sin\left(\frac{\gamma}{2}\right)\mathtt{Z}\otimes\mathtt{Z}$$

$$= \cos\left(\frac{\gamma}{2}\right)\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix} - \mathrm{i}\sin\left(\frac{\gamma}{2}\right)\begin{bmatrix}1&0&0&0\\0&-1&0&0\\0&0&-1&0\\0&0&0&1\end{bmatrix}$$

$$= \begin{bmatrix}\cos\left(\frac{\gamma}{2}\right)-\mathrm{i}\sin\left(\frac{\gamma}{2}\right) & 0 & 0 & 0\\ 0 & \cos\left(\frac{\gamma}{2}\right)+\mathrm{i}\sin\left(\frac{\gamma}{2}\right) & 0 & 0\\ 0 & 0 & \cos\left(\frac{\gamma}{2}\right)+\mathrm{i}\sin\left(\frac{\gamma}{2}\right) & 0\\ 0 & 0 & 0 & \cos\left(\frac{\gamma}{2}\right)-\mathrm{i}\sin\left(\frac{\gamma}{2}\right)\end{bmatrix}$$

$$= \begin{bmatrix}\mathrm{e}^{-\mathrm{i}\gamma/2} & 0 & 0 & 0\\ 0 & \mathrm{e}^{\mathrm{i}\gamma/2} & 0 & 0\\ 0 & 0 & \mathrm{e}^{\mathrm{i}\gamma/2} & 0\\ 0 & 0 & 0 & \mathrm{e}^{-\mathrm{i}\gamma/2}\end{bmatrix}.$$

Where we used the fact that

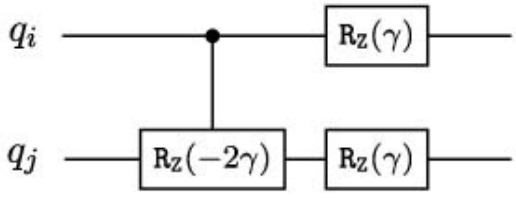
$$\exp\left(-\frac{1}{2}\mathrm{i}\gamma\right) = \cos\left(\frac{\gamma}{2}\right) - \mathrm{i}\sin\left(\frac{\gamma}{2}\right)$$

and

$$\exp\left(\frac{1}{2}\mathrm{i}\gamma\right) = \cos\left(\frac{\gamma}{2}\right) + \mathrm{i}\sin\left(\frac{\gamma}{2}\right).$$

The following lemma provides a quantum circuit for the operator $\exp\left(-\frac{1}{2}\mathrm{i}\gamma\sigma_z\sigma_z\right)$.

Lemma 8. *The operator* $\exp\left(-\frac{1}{2}\mathrm{i}\gamma\sigma_z^i\sigma_z^j\right)$ *can be represented by the following circuit:*



Proof. From (6.3.3), we can write

$$\mathrm{R_Z}(\gamma) = \begin{bmatrix} \mathrm{e}^{-\mathrm{i}\gamma/2} & 0 \\ 0 & \mathrm{e}^{\mathrm{i}\gamma/2} \end{bmatrix} = \mathrm{e}^{-\mathrm{i}\gamma/2} \begin{bmatrix} 1 & 0 \\ 0 & \mathrm{e}^{\mathrm{i}\gamma} \end{bmatrix}.$$

The multiplier $\exp\left(-\frac{1}{2}\mathrm{i}\gamma\right)$ is an unobservable global phase and can be ignored. Now,

$$\mathrm{R_Z}(\gamma) \otimes \mathrm{R_Z}(\gamma) = \begin{bmatrix} 1 & 0 \\ 0 & \mathrm{e}^{\mathrm{i}\gamma} \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & \mathrm{e}^{\mathrm{i}\gamma} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \mathrm{e}^{\mathrm{i}\gamma} & 0 & 0 \\ 0 & 0 & \mathrm{e}^{\mathrm{i}\gamma} & 0 \\ 0 & 0 & 0 & \mathrm{e}^{2\mathrm{i}\gamma} \end{bmatrix}.$$

Finally, the matrix representation of the $\mathrm{CR}_z(-2\gamma)$ gate is

$$\mathrm{CR_Z}(-2\gamma) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \mathrm{e}^{-2\mathrm{i}\gamma} \end{bmatrix},$$

and the matrix representation of the whole circuit shown in the lemma is

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \mathrm{e}^{\mathrm{i}\gamma} & 0 & 0 \\ 0 & 0 & \mathrm{e}^{\mathrm{i}\gamma} & 0 \\ 0 & 0 & 0 & \mathrm{e}^{2\mathrm{i}\gamma} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \mathrm{e}^{-2\mathrm{i}\gamma} \end{bmatrix} = \mathrm{e}^{\mathrm{i}\gamma/2} \begin{bmatrix} \mathrm{e}^{-\mathrm{i}\gamma/2} & 0 & 0 & 0 \\ 0 & \mathrm{e}^{\mathrm{i}\gamma/2} & 0 & 0 \\ 0 & 0 & \mathrm{e}^{\mathrm{i}\gamma/2} & 0 \\ 0 & 0 & 0 & \mathrm{e}^{-\mathrm{i}\gamma/2} \end{bmatrix}.$$

Again, the global phase can be ignored and we arrive at the same matrix expression we obtained for $\exp\left(-\frac{1}{2}\mathrm{i}\gamma\sigma_z\sigma_z\right)$. □

11.4.2 QAOA circuit

As pointed out in Chapters 8 and 9, existing quantum processors often have limited qubit connectivity, so that we need to use a particular embedding scheme in order to map the Max-Cut graph onto the QPU graph. The simplest case is a one-to-one mapping of the graph nodes and the connectivity edges. Figure 11.2 displays one such graph (implemented in Rigetti's Aspen system) consisting of eight nodes (embedded in qubits 1,…,8) and eight edges. Nodes 1, 3, 5, and 7 each have one connection and nodes 2, 4, 6, and 8 each have three connections.

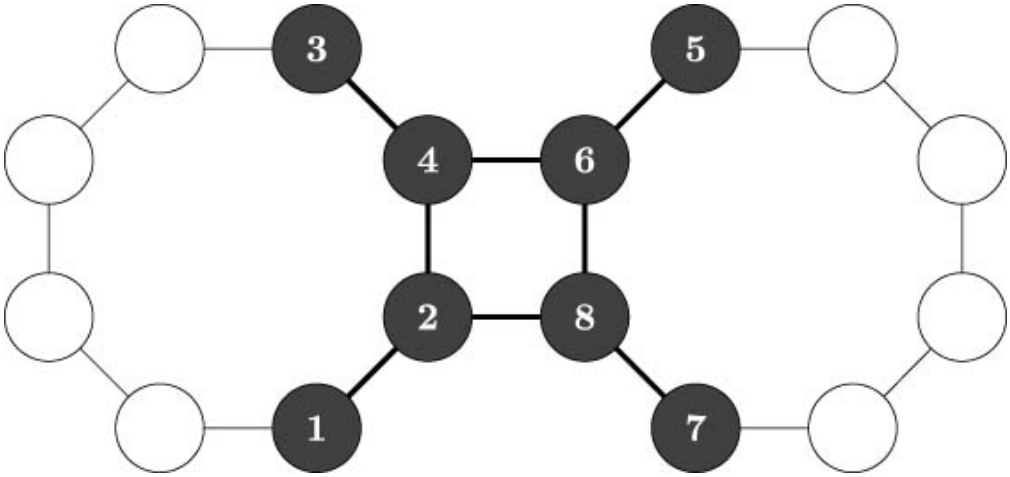


Figure 11.2: Embedding of the Max-Cut optimisation problem on Rigetti's Aspen system.

The corresponding QAOA circuit is shown in Figure 11.3.

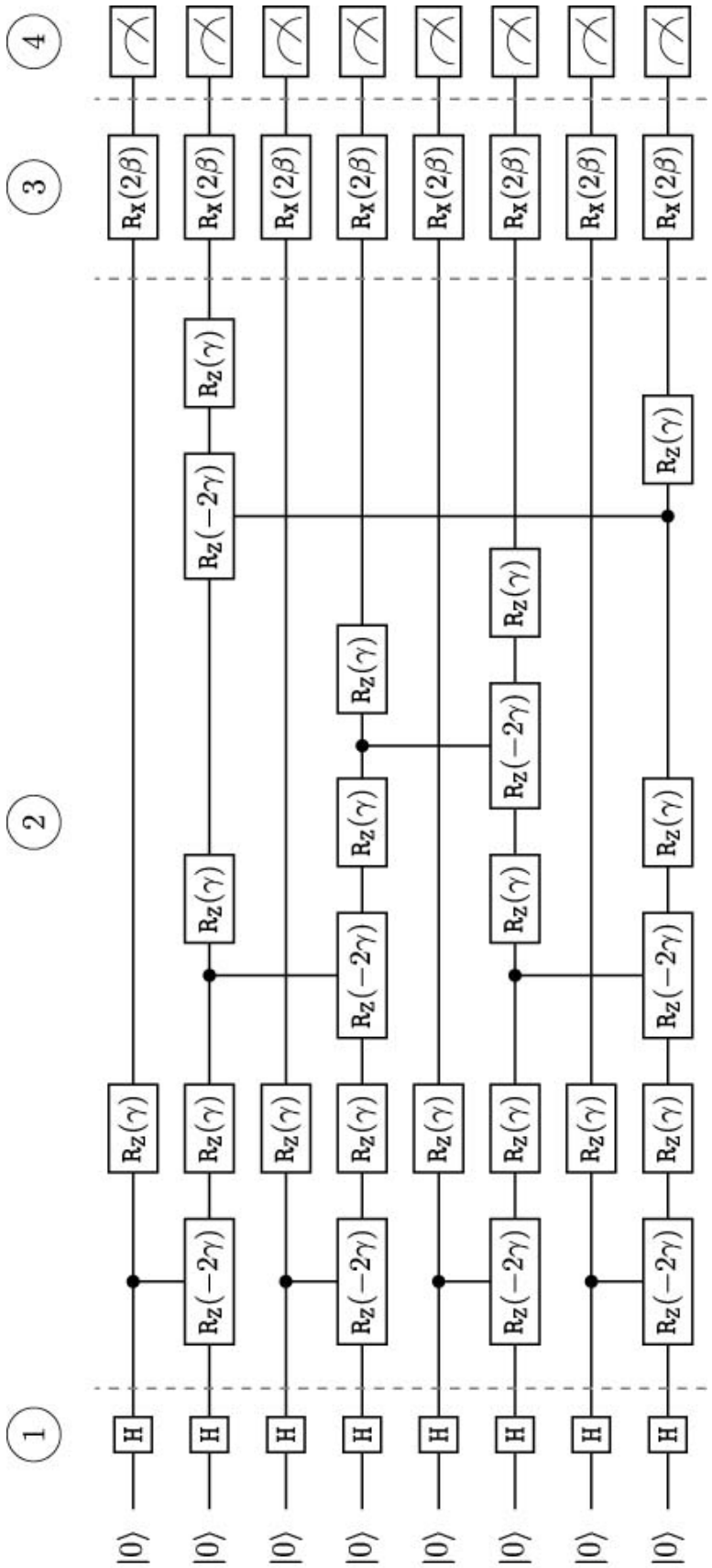


Figure 11.3: QAOA circuit for the Max-Cut problem.

The circuit consists of four distinct layers. The first one is a layer of Hadamard gates creating an equal superposition of states $|0\rangle$ and $|1\rangle$ – transformation of the basis states $|0\rangle$ into the basis states $|+\rangle$. The second layer represents the action of the phase Hamiltonian controlled by the adjustable parameter γ. The third layer represents the action of the mixing Hamiltonian controlled by the adjustable parameter β. The final layer consists of measurement operators. The second and third layers can be applied multiple times with different values of parameters γ and β.

The optimal solution found by the QAOA using `Qiskit` quantum simulator for the case of equal weights ($w_{ij} = 1$ for all $\{i,j\} \in G$) is shown in Figure 11.4. The optimal solution reads as the bitstring 10011001, and is represented by the dashed curve that separates nodes into two equal subsets and cuts across all edges of the graph.

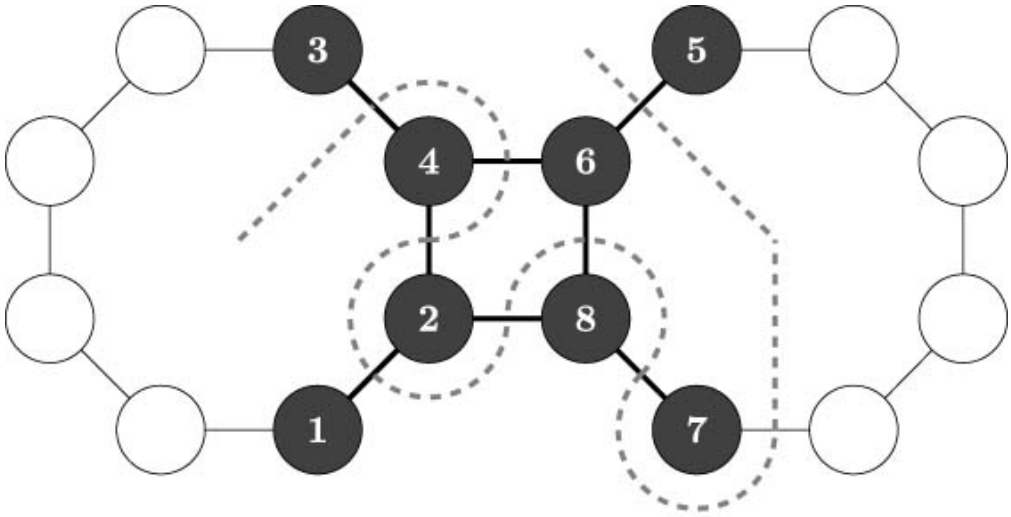


Figure 11.4: Visualisation of the Max-Cut problem solution.

Here, the graph nodes (qubits) {1,4,5,8} belong to group "1" and the graph nodes {2,3,6,7} belong to group "0".

In the case of unequal weights the circuit layout remains the same but the adjustable gate parameters reflect the relative magnitude of the weights assigned to different edges. For example, if the weight

assigned to the connection between nodes 1 and 2 is increased from 1 to 5 and all other weights remain equal to 1, then the phase Hamiltonian term that corresponds to the connection between nodes 1 and 2 changes from $\exp\left(-\frac{1}{2}\mathrm{i}\gamma\sigma_z\sigma_z\right)$ to $\exp\left(-\frac{5}{2}\mathrm{i}\gamma\sigma_z\sigma_z\right)$, and the corresponding segment of the quantum circuit changes to the one shown in Figure 11.5, with the rest of the circuit remaining the same.

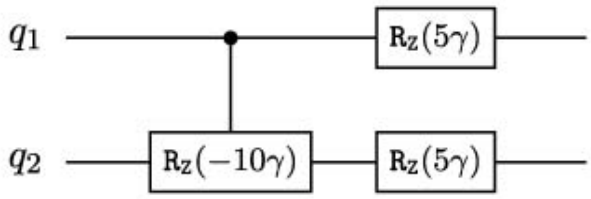


Figure 11.5: Weight w_{12} increases from 1 to 5.



QAOA can successfully solve NP-hard combinatorial optimisation problems in their QUBO formulations.

Summary

In this chapter, we studied a quantum optimisation algorithm inspired by the principles of adiabatic evolution of quantum systems. In this regard, QAOA can be seen as a quantum gate model counterpart of the AQC approach to solve classically hard optimisation problems.

We discussed the Suzuki-Trotter expansion, which provides an approximation of the time evolution of quantum mechanical systems and which lies at the heart of the algorithm. We illustrated QAOA implementation (gates and circuit) on the Max-Cut problem, which has many applications in finance.

In the next chapter, we will explore the sources of power of parameterised quantum circuits.

12
The Power of Parameterised Quantum Circuits

As we have seen in the previous chapters, there is a wide range of QML models based on parameterised quantum circuits. One reason for this is their tolerance to noise [222], which is important when we work with the NISQ hardware. However, this does not fully explain the popularity of PQCs or why they are considered strong competitors to classical ML models. There must be some fundamental properties of PQCs that make them superior to their classical counterparts. In this chapter, we discuss two such properties: resistance to overfitting and larger expressive power.

Resistance to overfitting is a direct consequence of the fact that a typical PQC – one without mid-circuit measurement – can be represented by a linear unitary operator. Linear models impose strong regularisation, thus preventing overfitting. At the same time, the model remains powerful due to the mapping of the input into the higher-dimensional Hilbert space where it may be easier to perform classification if the PQC is trained as a discriminative model (QNN).

Expressive power is related to the model's ability to express different relationships between variables, i.e., its ability to learn complex data structures. It appears that PQCs trained as generative models (QCBM) have strictly larger expressive power than their equivalent classical versions (such as RBM).

12.1 Strong Regularisation

Parameterised quantum circuits trained as classifiers face the same challenge as classical models: the need to generalise well to unseen data

points. Classically, we have a wide range of supervised learning models and regularisation techniques to choose from. These regularisation techniques that fight overfitting are model specific. For example, we can try to restrict the depth of the decision trees or to impose a penalty term in the cost function when training neural networks.

Consider a conventional feedforward neural network as, arguably, the most direct classical counterpart of a quantum classifier. In both classical and quantum cases, the signal travels through the network in one direction and the layers of quantum gates can be compared to the layers of classical activation units. Regardless of whether we apply L_1 (Lasso) or L_2 (Ridge) penalty terms, or use dropout techniques, we would like to have a measure of regularisation present in the network. This is an interesting theoretical problem as well as an important practical task that allows us to develop an optimal strategy for fighting overfitting. Ideally, such a measure should be applicable to both classical and quantum neural networks to provide a meaningful comparison of their respective regularisation properties.

Very often, relatively small network weights are associated with a high degree of regularisation and relatively high network weights are symptoms of overfitting. However, it would be highly desirable to have a formal mathematical tool for quantifying the network capacity to overfit. One such possible well-defined measure that captures the degree of regularisation is the Lipschitz constant.

12.1.1 Lipschitz constant

Following Gouk [115], given two metric spaces $(\mathcal{X}, d_{\mathcal{X}})$ and $(, d)$ a function $f: \mathcal{X} \rightarrow$, is said to be Lipschitz continuous if there exists a constant $k \geq 0$ such that

$$d(f(\mathrm{x}_1), f(\mathrm{x}_2)) \leq k d_{\mathcal{X}}(\mathrm{x}_1, \mathrm{x}_2), \quad \text{for all } \mathrm{x}_1, \mathrm{x}_2 \in \mathcal{X}.$$

The value of k is known as the Lipschitz constant, and the function is referred to as k-Lipschitz. We are interested in the smallest possible Lipschitz constant or, at least, its upper bound. To obtain the upper bound estimate, we should note some useful properties of feedforward neural networks.

In the case of a j-th layer of a feedforward neural network, x_1 and x_2 are the n-dimensional sample outputs of the previous layer, $j-1$, and $f(x_1)$ and $f(x_2)$ are the m-dimensional outputs of layer j. The metrics d_X and d can be, for example, L_1 or L_2 norms.

A feedforward neural network consisting of l fully connected layers can be expressed as a series of function compositions:

$$f(\mathrm{x}) = (\phi_l \circ \phi_{l-1} \circ \ldots \circ \phi_1)(\mathrm{x}),$$

where each φ_j implements the j-th layer affine transformation of the n-dimensional input x, parameterised by an $m \times n$ weight matrix W_j and an m-dimensional bias vector b_j:

$$\phi_j(\mathrm{x}) = \mathrm{W}_j\mathrm{x} + \mathrm{b}_j.$$

The composition of a k_1-Lipschitz function with a k_2-Lipschitz function is a k_1k_2-Lipschitz function [115]. Therefore, we can compute the Lipschitz constants for each layer separately and combine them together to obtain an upper bound on the Lipschitz constant for the entire network.

Choose d_X and d to be the L_2 norms $\|\cdot\|_2$. In this case, we obtain the following relationship from the definition of Lipschitz continuity for the fully connected network layer j:

$$\|(\mathrm{W}_j\mathrm{x}_1 + \mathrm{b}_j) - (\mathrm{W}_j\mathrm{x}_2 + \mathrm{b}_j)\|_2 \leq k\|\mathrm{x}_1 - \mathrm{x}_2\|_2.$$

Introducing $a = x_1 - x_2$ and assuming that $x_1 \neq x_2$ we arrive at the estimate

$$\frac{\|\mathrm{W}_j \mathrm{a}\|_2}{\|\mathrm{a}\|_2} \leq k.$$

The smallest Lipschitz constant of the fully connected network layer, $L(\varphi_j)$, is equal to the supremum of the left-hand side of inequality (12.1.1):

$$L(\phi_j) := \sup_{\mathrm{a} \neq 0} \frac{\|\mathrm{W}_j \mathrm{a}\|_2}{\|\mathrm{a}\|_2}.$$

The operator norm (12.1.1) is given by the largest singular value of the weight matrix W_j, which corresponds to the spectral norm – the maximum scale by which the matrix can stretch a vector. It is straightforward to calculate using any of the suitable open-source packages, for example sklearn.decomposition.TruncatedSVD from the `scikit-learn` package.

In the case of quantum neural networks, any parameterised quantum circuit operating on n qubits, regardless how complex and deep, can be represented by a $2^n \times 2^n$ unitary matrix. Since all singular values of the unitary matrix are equal to one, this gives us a natural benchmark for comparison of regularisation capabilities of various networks.

12.1.2 Regularisation example

The Australian Credit Approval (ACA) dataset [241, 242] we analysed in Chapter 8 can serve as a good illustrative example. We can compare the performance of classical and quantum neural networks while monitoring regularisation as measured by the Lipschitz constant.

The classical neural network is an MLP Classifier with two hidden layers. Each hidden layer holds the same number of activation units as the number of features in the ACA dataset (14), so that we have to calculate the largest singular values for two 14 × 14 square matrices. The features are standardised with `sklearn.preprocessing.StandardScaler`.

We also use `sklearn.neural_network.MLPClassifier` to construct the classifier with the set of hyperparameters shown in Table 12.1:

Hyperparameter	Value
Number of hidden layers:	2
Number of activation units in each layer:	14
Activation function:	tanh
Solver:	adam
Intial learning rate:	0.01
Number of iterations:	5000
Random state:	0
Regularisation parameter, α:	variable

Table 12.1: MLP Classifier hyperparameters.

The MLP Classifier regularisation parameter α is our control variable. It controls the L_2 regularisation term in the network cost function: the larger this parameter, the more large network weights are penalised. All other parameters were set at their default values.

The quantum neural network is shown in Figure 8.5. The parameterised quantum circuit consists of just 7 fixed two-qubit gates (CZ) and 15 adjustable one-qubit gates (R_X and R_Y). Table 12.2 compares the MLP and the QNN classifiers on the in-sample and out-of-sample datasets (the ACA dataset was split 50:50 into training and test datasets using sklearn.preprocessing.StandardScaler).

We observe that QNN provides strong regularisation with similar performance on the in-sample and out-of-sample datasets as expected from the network represented by the unitary matrix.

Classifier	Average F_1 score	Average F_1 score	Lipschitz Constant

	(in-sample)	(out-of-sample)	(upper bound)
MLP, $\alpha = 0.001$	1.00	0.78	36.2
MLP, $\alpha = 0.01$	1.00	0.79	33.5
MLP, $\alpha = 0.1$	1.00	0.80	18.6
MLP, $\alpha = 1$	0.99	0.83	7.4
MLP, $\alpha = 10$	0.90	0.86	1.3
MLP, $\alpha = 40$	0.85	0.86	0.5
MLP, $\alpha = 50$	0.35	0.37	1e-05
QNN	0.86	0.85	1.0

Table 12.2: F_1 scores and Lipschitz constants for MLP and QNN classifiers trained on the ACA dataset.

Further, we observe that the equivalent degree of regularisation can be achieved by MLP only with exceptionally large values of the regularisation parameter α. Making α any larger completely destroys the learning abilities of the network. For the chosen MLP configuration, the critical value of α is between 40 and 50.



Parameterised quantum circuits can be represented as (high-dimensional) norm-preserving unitary matrices. This ensures strong regularisation properties of the quantum neural networks.

Now we can move to the next feature of the parameterised quantum circuits: their expressive power. We can define the expressivity of a PQC as the circuit's ability to generate pure quantum states that are well representative of the Hilbert space [266]. In other words, from the QML point of view, the expressive power of a PQC is its ability to learn ("express") complex data structures. In the following section, we will try to quantify the degree of expressivity inherent in different PQC types.

12.2 Expressive Power

We saw in previous chapters how PQCs can be applied to solving optimisation problems (QAOA and VQE) as well as to various machine learning tasks covering both discriminative (QNN classifier) and generative (QCBM market generator) use cases. In general, the PQCs we used for quantum machine learning tasks can be divided into two types [88]: tensor network PQC (similar to the QNN circuit in Figure 8.4) and multilayer PQC (similar to the QCBM circuit in Figure 9.1). What is their expressive power and how can we rank them? Before trying to answer this question, let us have a look at a simple illustrative example: quantum circuits specified on a single quantum register.

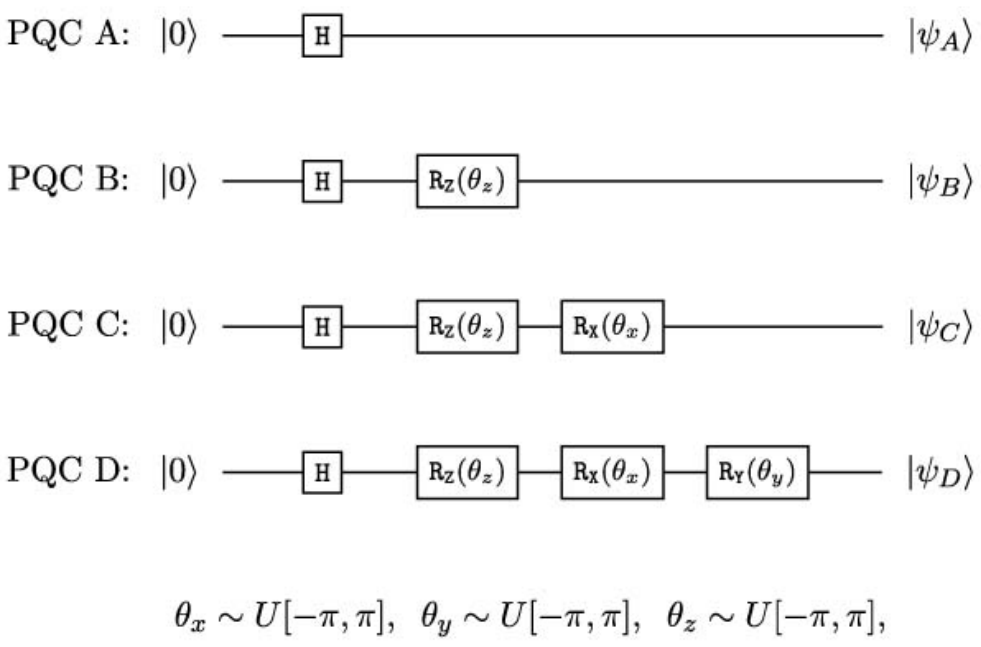


Figure 12.1: PQCs with different expressive powers.

Figure 12.1 displays four one-qubit circuits with dramatically different expressive powers, where $U[-\pi,\pi]$ denotes the Uniform distribution over the closed interval $[-\pi,\pi]$. Let us go through them one by one.

PQC A starts with the qubit state initialised as $|0\rangle$ – North Pole on the Bloch sphere (Figure 7.2). The only gate is the Hadamard gate H that moves $|0\rangle$ to $(|0\rangle + |1\rangle)/\sqrt{2}$. Thus, state $|\psi_A\rangle$ can only be a single point on the Bloch sphere.

PQC B also starts with the qubit state initialised as $|0\rangle$ and applies the Hadamard gate transforming the initial state into $(|0\rangle + |1\rangle)/\sqrt{2}$ before applying the rotation R_Z around the z-axis by an angle θ_z drawn from the Uniform distribution on $[-\pi,\pi]$. The final state $|\psi_B\rangle$ can be any point on the equator, all reached with equal probability.

PQC C adds a rotation R_X to PQC B, by an angle θ_x drawn from the Uniform distribution on $[-\pi,\pi]$. With two rotations around two orthogonal axes we can reach any point on the Bloch sphere. However, with angles θ_z and θ_x drawn from the Uniform distribution on $[-\pi,\pi]$ we do not have a Uniform distribution of points on the Bloch sphere for state $|\psi_C\rangle$. We observe the highest density around points $(|0\rangle + |1\rangle)/\sqrt{2}$ and $(|0\rangle - |1\rangle)/\sqrt{2}$, which are the points where the equator crosses the 0° and 180° meridians, and the lowest density along the 90° and 270° meridians.

Finally, PQC D adds one more rotation R_Y around the y-axis by an angle θ_y drawn from the Uniform distribution on $[-\pi,\pi]$. This rotation results in spreading the previously clustered points more evenly around the Bloch sphere, thus making all points on the Bloch sphere equally accessible.

Therefore, in terms of our ability to explore the Hilbert space, we have the following hierarchy of the expressive power of the PQCs introduced above:

PQC D > PQC C > PQC B > PQC A.

We can now return to the PQCs developed in the previous chapters.

12.2.1 Multilayer PQC

A multilayer PQC (MPQC) consists of multiple blocks of quantum circuits in which the arrangement of quantum gates in each block is identical [28, 189]. Figure 12.2 shows a schematic representation of the MPQC.

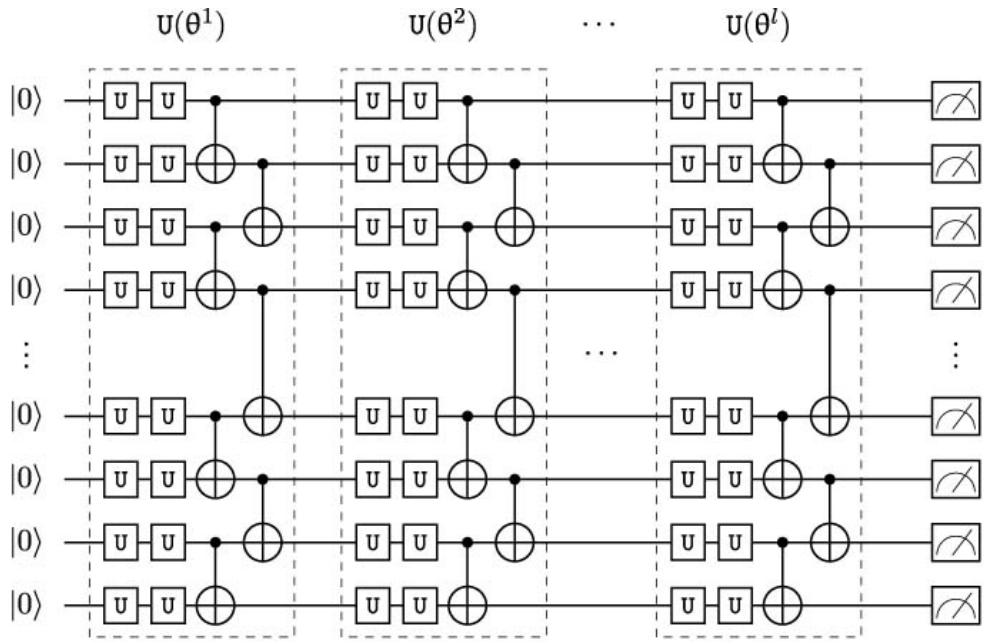


Figure 12.2: Schematic representation of a multilayer PQC.

The following mathematical formalism can be used to describe MPQC. The input n-qubit quantum state with all qubits initialised as $|0\rangle$ in the computational basis is $|0\rangle^{\otimes n}$, the total number of blocks is denoted l, and the i-th block is denoted $U(\theta^i)$, where the number of parameters is proportional to the number of qubits, and n is logarithmically proportional to the dimension of the generated data (this reflects our assumption about the data encoding scheme). The generated output state of the circuit thus reads

$$|\psi\rangle = \prod_{i=1}^{l} U(\theta^i)|0\rangle^{\otimes n}.$$

12.2.2 Tensor network PQC

A tensor network PQC (TPQC) treats each block as a local tensor. The arrangement of the blocks follows a particular network structure, such as matrix product states or tree tensor networks [144]. A schematic representation of TPQC is shown in Figure 12.3.

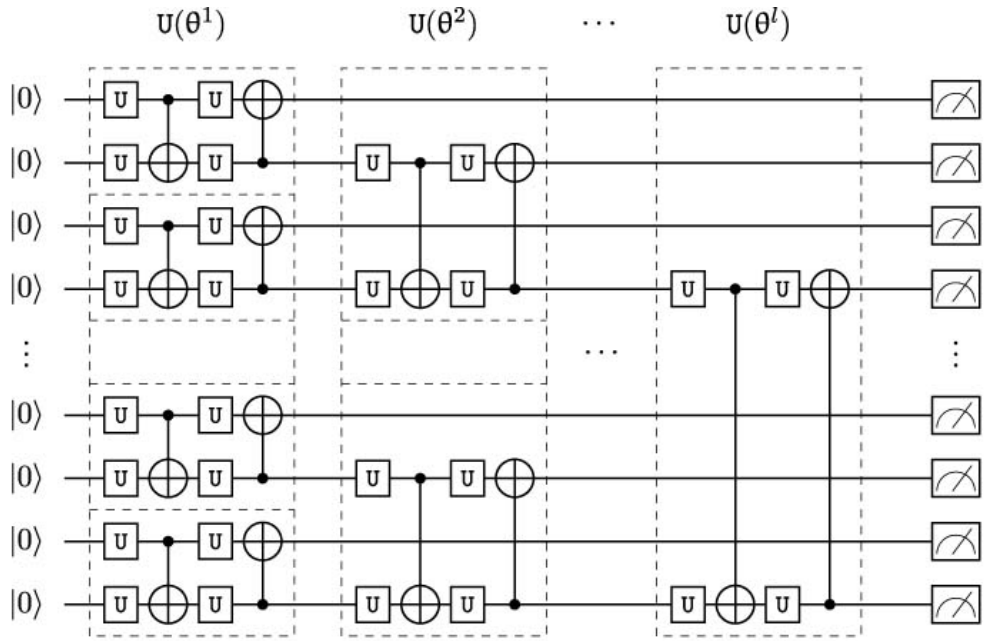


Figure 12.3: Schematic representation of a tensor network PQC.

Mathematically, the i-th block $U(\theta^i)$ is composed of M_i local tensor blocks, with $M_i \propto n/2^i$, denoted as $U(\theta^i) = \otimes_{j=1}^{M_i} U(\theta_j^i)$. Note that many of these tensor blocks may be identity operators. The generated state is thus of the form

$$|\psi\rangle = \prod_{i=1}^{l} \bigotimes_{j=1}^{M_i} U(\theta_j^i)|0\rangle^{\otimes n}.$$

12.2.3 Measures of expressive power

The main question to answer is whether MPQC and TPQC have larger expressive power in comparison with their classical counterparts, such as classical neural networks. The expressive power of a model can be defined in many different ways, for example, as a model capacity to

express different relationships between variables [22]. Deep neural networks serve as a good example of powerful models capable of learning complex data structures [94]. Therefore, the power of a model can be quantified by its complexity, with the *Vapnik-Chervonenkis dimension* being a complexity measure of choice [293]. The objective is to provide an estimate of how well a model generalises to the unseen data.

Another popular approach is the *Fisher information*, which describes the geometry of a model parameter space [247]. Arguably, the *effective dimension* based on Fisher information, rather than Vapnik-Chervonenkis dimension, is a better measure to study the power of quantum and classical neural networks [1].

However, one of the most natural metrics of expressive power is *entanglement entropy*, which allows us to establish a well-defined ranking of quantum and classical machine learning models. In this chapter, we will present the expressive power estimates obtained in [88] for TPQC and MPQC based on entanglement entropy.

Let us recall the definitions of entropy in statistical mechanics (the Gibbs entropy S) and in information theory (the Shannon entropy H) introduced in Chapter 6:

$$S := -k_B \sum_i p_i \log(p_i) \quad \text{and} \quad \mathrm{H} := -\sum_i p_i \log_2(p_i).$$

Here, p_i is the probability that the microstate i is taken from an equilibrium ensemble in the case of the Gibbs entropy, and that the message i is picked from the message space in the case of the Shannon entropy.

These definitions of entropy can be extended to the quantum case. In Chapter 1, we introduced the density matrix as a universal tool for describing pure and mixed quantum states:

$$\rho := \sum_{i=1}^{N} \sum_{j=1}^{N} \rho_{ij} \, |i\rangle \, \langle j| \,,$$

where $(|i\rangle)_{i=1,\ldots,N}$ are the basis vectors of a given quantum system. The *von Neumann entropy* $\mathcal{S}$ is then defined as

$$\mathcal{S}(\rho) := -\mathrm{Tr}(\rho \log(\rho)).$$

Since the density matrix is Hermitian, it is *diagonalisable*, so that there exists a basis $(|k\rangle)_{k=1,\ldots,N}$ such that

$$\rho = \sum_{k=1}^{N} \rho_{kk} \, |k\rangle \, \langle k| =: \sum_{k=1}^{N} p_k \, |k\rangle \, \langle k| \,, \quad \text{where} \sum_{k=1}^{N} p_k = 1.$$

The eigenvalues of the operator $\rho\log(\rho)$ are thus $(p_k \log(p_k))_{k=1,\ldots,N}$, and we obtain the following expression for the von Neumann entropy:

$$\mathcal{S}(\rho) = -\mathrm{Tr}(\rho \log(\rho)) = -\sum_{k} p_k \log(p_k).$$

From (12.2.3) and (12.2.3) we see that for the orthogonal mixture of quantum states, the quantum and classical entropies coincide.

If the system has two component parts, *A* and *B*, we can define the *reduced density matrix* as the *partial trace* of the density matrix over the subspace of the Hilbert space we are not interested in. Let $(|a_i\rangle)_{i=1,\ldots,N}$ be the standard orthonormal basis of the Hilbert space $\mathbb{H}_A$ of system *A*, and $(|b_j\rangle)_{j=1,\ldots,M}$ be the standard orthonormal basis of the Hilbert space $\mathbb{H}_B$ of system *B*. The density matrix ρ_{AB} of the bipartite system *AB* on the tensor product Hilbert space $\mathbb{H}_A \otimes \mathbb{H}_B$ can then be represented as

$$\rho_{AB} = \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{k=1}^{N} \sum_{l=1}^{M} c_{ijkl} \, |a_i\rangle \, \langle a_k| \otimes |b_j\rangle \, \langle b_l| \,,$$

for some coefficients c_{ijkl}. The partial traces then read

$$\mathrm{Tr}_B(\rho_{AB}) = \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{N}\sum_{l=1}^{M} c_{ijkl}\, |a_i\rangle \langle a_k| \langle b_l|b_j\rangle ,$$

which is a reduced density matrix ρ_A on $\mathbb{H}_A$, and

$$\mathrm{Tr}_A(\rho_{AB}) = \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{N}\sum_{l=1}^{M} c_{ijkl}\, |b_j\rangle \langle b_l| \langle a_k|a_i\rangle ,$$

which is a reduced density matrix ρ_B on $\mathbb{H}_B$. Note that $\mathrm{Tr}(|a_i\rangle\langle a_k|) = \langle a_k|a_i\rangle$ and $\mathrm{Tr}(|b_j\rangle\langle b_l|) = \langle b_l|b_j\rangle$.

Example: Consider the two-qubit system in the state

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle),$$

which is one of the four maximally entangled Bell states (Section 6.5.2). We assume that the first qubit is system A and the second qubit is system B. This state corresponds to the following density matrix:

$$\rho_{AB} := |\psi\rangle\langle\psi| = \frac{1}{2}\Big(|01\rangle\langle 01| + |01\rangle\langle 10| + |10\rangle\langle 01| + |10\rangle\langle 10| \Big).$$

Let us now act on this state with the partial trace $\mathrm{Tr}_B(\cdot)$:

$$\begin{aligned}\rho_A := \mathrm{Tr}_B(\rho_{AB}) &= \frac{1}{2}\Big(|0\rangle\langle 0|\langle 1|1\rangle + |0\rangle\langle 1|\langle 0|1\rangle + |1\rangle\langle 0|\langle 1|0\rangle + |1\rangle\langle 1|\langle 0|0\rangle \Big) \\ &= \frac{1}{2}\Big(|0\rangle\langle 0| + |1\rangle\langle 1| \Big) = \frac{1}{2}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.\end{aligned}$$

The reduced density matrix ρ_A in (12.2.3) is the same as the density matrix ρ in (1.3.3), which describes a statistical ensemble of states $|0\rangle$

and $|1\rangle$ (mixed state), i.e., a physical system prepared to be either in state $|0\rangle$ or state $|1\rangle$ with equal probability.

The *entanglement entropy* of a bipartite system AB is then defined as

$$\mathcal{S}(\rho_A) := -\mathrm{Tr}(\rho_A \log(\rho_A)) = -\mathrm{Tr}(\rho_B \log(\rho_B)) =: \mathcal{S}(\rho_B),$$

and can be used as a measure of expressive power of a model in the following way. First, note that TPQC, MPQC and classical neural networks have a close connection with *tensor networks*, such as matrix product states (MPS) [88]. The key question is then whether the given quantum system can be efficiently represented by the MPS.

A quantum system that satisfies the *area law* (its entanglement entropy grows proportionally with the boundary area) has an efficient MPS representation. At the same time, a quantum system that satisfies the *volume law* (its entanglement entropy grows proportionally with the volume) cannot be efficiently represented by MPS [88].

12.2.4 Expressive power of PQC

In Chapter 5, we introduced the Restricted Boltzmann Machine (RBM) – a neural network operating on stochastic binary activation units, which is a natural classical counterpart of parameterised quantum circuits. We considered two types of RBM:

- a shallow two-layer network where the activation units in the visible layer are connected to the activation units in the hidden layer with no connections between the activation units within the same layer;
- a deeper multi-layer network of stacked RBMs where the hidden layer of the k-th RBM serves as the visible layer of the $(k+1)$-th RBM. Such stacked RBMs (trained sequentially) are called Deep Boltzmann Machines (DBMs).

It is also possible to impose further restrictions on the connections between the RBM layers. In *short-range* RBMs, we restrict the connectivity of the hidden layer activation units such that they are allowed to connect to the limited number of activation units in the visible layer that are in close proximity to each other (local connectivity) [84]. In *long-range* RBMs, we allow connections between the hidden layer activation units and the visible layer activation units that are not necessarily local.

It has been established by Deng, Li, and Sarma [85] that the entanglement entropy of all short-range RBM states satisfies an area law for arbitrary dimensions and bipartition geometry. For long-range RBM states, such states could exhibit volume law entanglement. Therefore long-range RBMs are capable of representing quantum states with large entanglement.

It is probably not surprising that a DBM would have even larger expressive power than a single RBM. However, using entanglement entropy as a measure of expressive power, Du, Hsieh, Liu, and Tao have proven in [88] that MPQC have strictly larger expressive power than DBM. The main result can be formulated as the following theorem:

Theorem 10 (Expressive Power Theorem)**.** *The expressive power of MPQC and TPQC with* $\mathcal{O}(poly(n))$ *single qubit gates and* CNOT *gates, and classical neural networks with* $\mathcal{O}(poly(n))$ *trainable parameters, where n refers to the number of qubits or visible units, can be ordered as*

$$MPQC > DBM > \textit{long-range RBM} > TPQC > \textit{short-range RBM}.$$

Theorem 10 provides a solid theoretical foundation for experimental works aimed at establishing quantum advantage of PQC-based QML models. The larger expressive power of PQCs in comparison with their classical counterparts prompted the development of many such models in recent years. For example, a hybrid quantum-classical approach,

suitable for NISQ devices and harnessing the greater expressive power of quantum entanglement, was proposed in [59]. It was shown through numerical simulations that the Quantum Long Short Term Memory (QLSTM) model learns faster than the equivalent classical LSTM with a similar number of network parameters. In addition, the convergence of QLSTM was shown to be more stable than that of its classical counterpart. A Quantum Convolutional Neural Network (QCNN) was proposed in [58] which, due to its larger expressive power, achieved greater test accuracy compared to classical CNNs. The source of expressive power was the replacement of the classical convolutional filters with quantum convolutional kernels based on variational quantum circuits.



Multi-layer parameterised quantum circuits such as QCBM have strictly more expressive power than classical models such as RBM when only a polynomial number of parameters is allowed. For systems that exhibit quantum supremacy, a classical model cannot learn to reproduce the statistics unless it uses exponentially scaling resources [29].

Summary

In this chapter, we learned where the power of parameterised quantum circuits comes from. We started with the observation that quantum neural networks enjoy strong regularisation inherent in their architecture. This is due to the fact that any PQC, however wide and deep, is a unitary linear operator.

Next, we considered the expressive power of parameterised quantum circuits and established the concept of the expressive power hierarchy. The main result (Theorem 10) supports the experimental findings

indicating the presence of the elements of quantum advantage in various QML models compatible with the main characteristics of NISQ devices.

In the next chapter, we will go deeper into the less explored territory of new quantum algorithms, an area of very active research.

Join our book's Discord space

Join our Discord community to meet like-minded people and learn alongside more than 2000 members at: https://packt.link/quantum



13
Looking Ahead

The first generation of quantum algorithms appeared in the 1990s when quantum computers existed only as a concept. On the one hand, the absence of actual quantum hardware was a huge disadvantage since it made direct experiments impossible; on the other hand, it stimulated theoretical research not inhibited by the limitations and constraints of the imperfect early quantum computers. Researchers focused on devising algorithms that would achieve quadratic or even exponential speedup, assuming that powerful, error-free quantum computers would be available one day. It was the time when Shor's prime factorisation algorithm [265] and Grover's search algorithm [117] were discovered. Incidentally, as the book was about to be released, Peter Shor was named one of the four recipients of the 2022 Breakthrough Prize in Fundamental Physics (along with C. H. Bennett, G. Brassard, and D. Deutsch) for their foundational work in quantum information. Many such algorithms, in turn, relied on basic building blocks such as Quantum Phase Estimation and Quantum Fourier Transform [278]. These algorithms played an important role in demonstrating the capabilities of universal gate model quantum computers – if only they were available!

A quarter of a century later we are facing a different problem: the development of practical quantum computing algorithms and techniques that would allow us to extract value from NISQ computers. While quantum computing hardware is improving at a breathtaking pace, it is still too far from a state where it can break RSA encryption. What are existing quantum computers capable of doing? What is their relative strength in comparison with classical computers? In this chapter we look at several new, NISQ-friendly algorithms that bring us one step closer to achieving the quantum advantage.

13.1 Quantum Kernels

We start with the popular classical kernel method and then describe its quantum counterpart based on parameterised quantum circuits.

13.1.1 Classical kernel method

A *kernel method* is the key element of a powerful classical supervised learning algorithm: Support Vector Machine (SVM). Unlike a feedforward-neural-network-based classifier whose objective is to minimise the classification error, the SVM's objective is to maximise the margin, defined as the distance between a separating hyperplane (decision boundary separating samples belonging to different classes) and the training samples that are closest to this hyperplane [243]. The samples that are closest to the separating hyperplane are called *support vectors*, thus giving its name to the algorithm.

The maximisation of the margins lowers the generalisation error and helps fight overfitting. This is a very important property but finding the separating hyperplane is not an easy task for non-linearly separable data. Fortunately, the kernel method allows us to overcome this difficulty, by creating non-linear combinations of the original features and projecting them onto a higher-dimensional space where the data samples become linearly separable.

Whereas an SVM with linearly separable data operates on the inner product $\langle \mathrm{x}^i, \mathrm{x}^j \rangle$ of the training samples, the generalised version to non-linearly separable data operates on the kernel function

$$k(\mathrm{x}^i, \mathrm{x}^j) := \phi(\mathrm{x}^i)^\top \phi(\mathrm{x}^j),$$

where $\varphi : \mathbb{R}^N \to \mathbb{R}^M$, with $M \gg N$, is the feature map that projects the N-dimensional feature $\mathrm{x} := (x_1, \ldots, x_N)$ onto the M-dimensional feature space. The inner product (13.1.1) would be computationally

expensive to calculate directly but the kernel function is computationally inexpensive – this is known as the *kernel trick*. The kernel function can be seen as a similarity function operating on a pair of samples. For example, the radial basis function

$$k(\mathrm{x}^i, \mathrm{x}^j) = \exp\left(-\frac{\|\mathrm{x}^i - \mathrm{x}^j\|^2}{2\sigma^2}\right),$$

translates the distance between samples x^i and x^j (defined on $[0,\infty)^N$) into a similarity score (defined on the interval [0,1]).

The right choice of kernel function can make the classification task much easier. However, some kernels may be hard to compute. This is where quantum computing may play an important role by providing efficient quantum circuits to compute them.

13.1.2 Quantum kernel method

Wang, Du, Luo, and Ta [298] have shown a close correspondence between classical and quantum kernels. The feature map $\varphi(\cdot)$ coincides with the preparation of a quantum state via a parameterised quantum circuit $\mathcal{U}(\cdot)$, which maps the input data sample into a high-dimensional Hilbert space described by n qubits:

$$\phi(\mathrm{x}) \to |\psi(\mathrm{x})\rangle = \mathcal{U}(\mathrm{x})\,|0\rangle^{\otimes n}.$$

The kernel function then coincides with applying measurements on the prepared quantum states:

$$k(\mathrm{x}^i, \mathrm{x}^j) \to \left|\langle\psi(\mathrm{x}^j)|\psi(\mathrm{x}^i)\rangle\right|^2,$$

and allows for more expressive models in comparison with the alternative

$$k(\mathrm{x}^i, \mathrm{x}^j) = \phi(\mathrm{x}^i)^\top \phi(\mathrm{x}^j) \to \left\langle \psi(\mathrm{x}^j) | \psi(\mathrm{x}^i) \right\rangle .$$

Huang *et al.* [143] argued that even though the kernel function (13.1.2) seems to be more natural, the quantum kernel (13.1.2) can learn arbitrarily deep quantum neural networks (deep PQC). This is a strong result, especially in combination with the hierarchy of expressive power of parameterised quantum circuits (Chapter 12, Equation (10)).

Havlíček *et al.* [129] described how a quantum computer can be used to estimate the kernel. The kernel entries are the *fidelities* between different feature vectors (analogous to similarity scores in classical kernel methods). Burnham, Cleve, Watrous, and R. de Wolf [50] and Cincio, Subaşi, Sornborger, and Coles [66] investigated various fidelity estimation methods such as quantum fingerprinting and machine learning approach (both relying on the application of a CSWAP gate implementing the swap test). However, by using the fact that the states in the feature space are not arbitrary, the overlap between the quantum states can be estimated from the transition probability:

$$\left| \left\langle \psi(\mathrm{x}^j) | \psi(\mathrm{x}^i) \right\rangle \right|^2 = | \left\langle \mathbf{0} \right| \mathcal{U}^\dagger(\mathrm{x}^j) \mathcal{U}(\mathrm{x}^i) \left| \mathbf{0} \right\rangle |^2,$$

where, for brevity, we used the notation $|\mathbf{0}\rangle := |0\rangle^{\otimes n}$. The first step is the application of a composition of two consecutive feature map circuits (representing the operators $\mathcal{U}(\mathrm{x}^i)$ and $\mathcal{U}^\dagger(\mathrm{x}^j)$) to the initial state $|\mathbf{0}\rangle$. The second step is the measurement of the final state in the computational basis K times and counting the number κ of all-zero strings $|\mathbf{0}\rangle$. The frequency κ/K of the all-zero string is the estimate of the transition probability (the "similarity score").

The rest of the supervising learning protocol is classical, allowing for the natural embedding of quantumly computed kernels into the overall

framework: the algorithm remains essentially classical with only the classically hard task outsourced to the quantum chip.

13.1.3 Quantum circuits for the feature maps

Figure 13.1 displays a schematic representation of the feature map circuit. In this example, we work with an 8-dimensional dataset with features encoded in the rotation angles such that there is a direct mapping of a sample $x^i := (x_1^i,\ldots,x_8^i)$ into the vector of adjustable circuit parameters $\theta^i := (\theta_1^i,\ldots,\theta_8^i)$. The first section of the circuit implements the operator $\mathcal{U}(x^i)$, creating an entangled state due to the layer of fixed two-qubit CZ gates, whereas the second section of the circuit implements $\mathcal{U}^\dagger(x^j)$. Here we use the fact that

$$R_X^\dagger(\theta) = R_X(-\theta), \quad R_Y^\dagger(\theta) = R_Y(-\theta), \quad CZ^\dagger = CZ.$$

It is easy to see that if the samples x^i and x^j are identical (so that $\theta^i = \theta^j$), then $\mathcal{U}(x^i)\mathcal{U}^\dagger(x^j) = \mathcal{I}$ and all K measurements will return the all-zero string $|\mathbf{0}\rangle$.

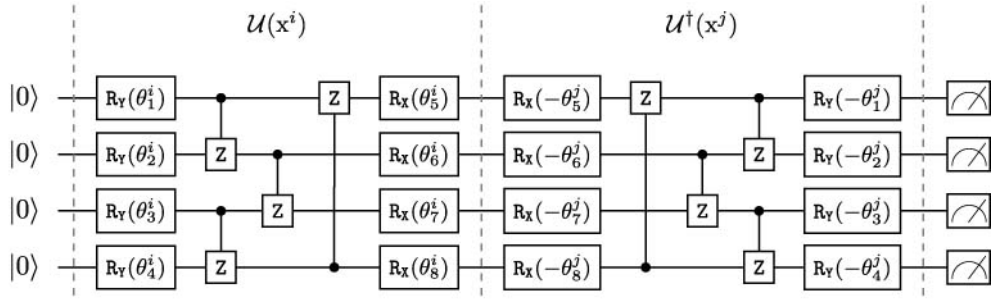


Figure 13.1: Schematic quantum kernel circuit.

The rest of the protocol is classical – the quantum computer is used to assist the classifier with the calculation of a kernel function that would not be feasible if only classical computational resources were available.

Let us now apply the quantum kernel method to the Australian Credit Approval dataset (introduced in Chapter 8) in order to estimate the

degree of similarity between samples drawn from the same class and samples drawn from two different classes. The ACA dataset consists of 690 samples, with 383 samples labelled as Class 0 and 307 samples labelled as Class 1, so the dataset is reasonably well balanced. Each sample consists of 14 features (continuous, integer, binary). In Chapter 8 we built a QNN classifier and tested its performance on the ACA dataset, employing the *angle encoding* scheme as explained in Section 7.2. We would like to build a feature map that is consistent with the angle encoding scheme and does not require the construction of a too deep PQC. In fact, we would like to build a feature map using the PQC that is as close as possible to the one shown in Figure 13.1. The proposed scheme can be embedded into all existing NISQ systems we considered earlier in this book. For example, we can use IBM's Melbourne system shown in Figure 13.2.

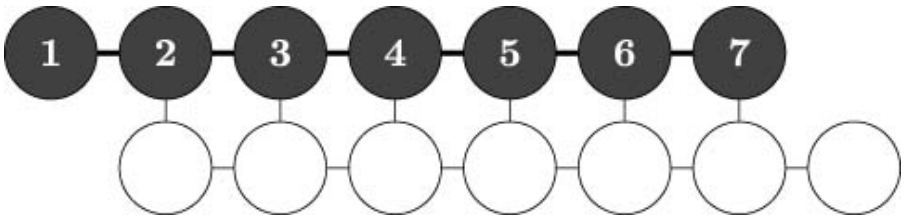


Figure 13.2: Embedding of the quantum kernel circuit into IBM's Melbourne system.

We know that 7 quantum registers (shown as shaded qubits connected by the thick lines in Figure 13.2) can encode a 14-feature data sample if we follow the angle encoding scheme. The corresponding circuit is shown in Figure 13.3. The linear sequential connectivity between the physical qubits makes the choice of the 2-qubit gates straightforward (and, in fact, similar to the one in Figure 13.1).

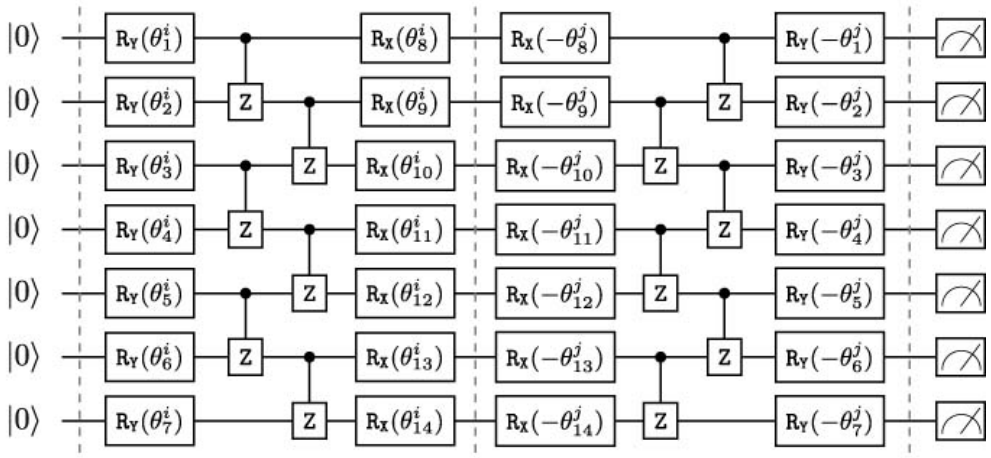


Figure 13.3: Quantum kernel circuit for the ACA dataset.

In the circuit shown in Figure 13.3, the angles θ^i and θ^j encode the data samples x^i and x^j, which can be drawn either from the same class or from two different classes. Running the circuit K times and calculating the number κ of all-zero bitstrings (after measurement) gives us the measure of similarity between samples x^i and x^j (estimated as the ratio κ/K). Figure 13.4 displays the mean values of the transition probabilities (similarity scores) obtained using the quantum kernel (13.1.2) by running the quantum circuit on the Qiskit simulator K = 10,000 times for each pair of data samples. The mean values were calculated across all possible pairs of samples from the corresponding classes.

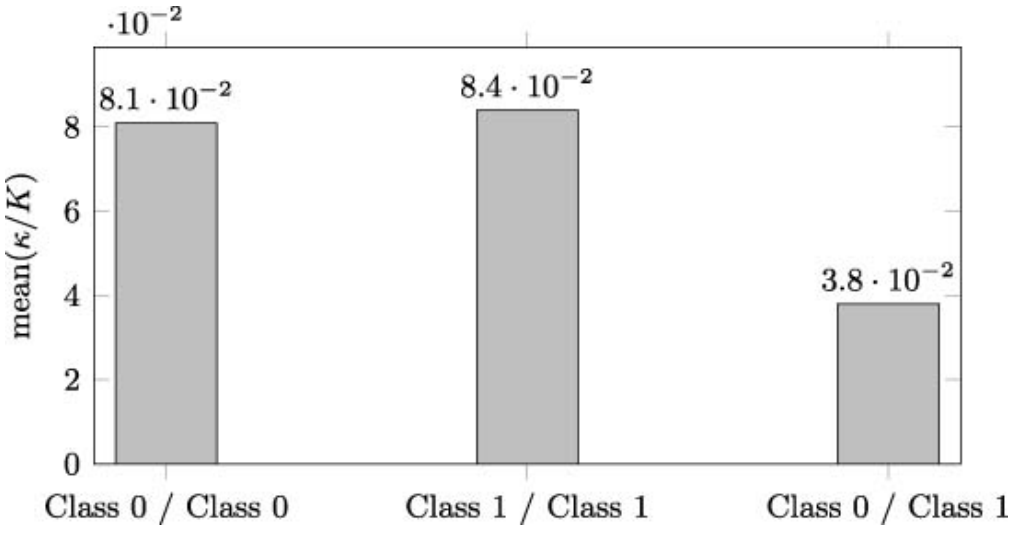


Figure 13.4: Mean values of the quantum kernel (13.1.2) for the ACA dataset.

As expected, samples drawn from the same class have, on average, significantly larger similarity scores given by the quantum kernel compared with samples drawn from two different classes.



Quantum kernels that can be efficiently calculated on quantum computers have the potential to improve the performance of hybrid quantum-classical machine learning models.

13.2 Quantum Generative Adversarial Networks

Generative Adversarial Networks (GANs) are powerful statistical techniques to generate (as much as needed) data close enough (in some sense) to given samples. They were introduced in [114] and originally tested on image data. Since then, they have seen wide applications in finance, for time series generation [301, 302], tuning of trading models [176], portfolio management [196], synthetic data generation [17], and diverse types of fraud detection [261]. The gist of it is to have a generator and a discriminator compete against each other in order to improve themselves: the generator improves by becoming better at generating good samples (i.e., close to real data) from random noise, whereas the discriminator improves by being able to recognise real data from "fake" (namely generated) data. Both the generator and the discriminator are usually built as neural networks with hyperparameters over which to optimise. Mathematically, given a generator $(\cdot,\theta)$ and a discriminator $(\cdot,\theta)$, where θ and θ represent the hyperparameters, the problem reads as follows:

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\theta}} \left\{ \mathbb{E}_{\mathrm{x}\sim\mathbb{P}_{\mathrm{data}}} \left[\log((\mathrm{x};\boldsymbol{\theta})\right] + \mathbb{E}_{\mathrm{z}\sim\mathbb{P}_{(\cdot,\boldsymbol{\theta})}} \left[\log\left(1-((\mathrm{z};\boldsymbol{\theta})\,;\boldsymbol{\theta})\right)\right] \right\},$$

where x $\sim \mathbb{P}_{\text{data}}$ means some sample x generated from the original dataset, whereas z $\sim \mathbb{P}$ refers to sample generated from the generator . We refer the interested reader to [95] for an overview of the advantages and the pitfalls of GANs in finance. Given this popularity and the existence of quantum neural networks (Chapter 8), it is thus natural to explore the question of whether GANs can be extended to the quantum world, and whether there is any advantage in doing so.

The main principles of Quantum Generative Adversarial Network (QGAN) – introduced simultaneously by Lloyd and Weedbrook [192] and by Dallaire-Demers and Killoran [77] – remain the same, relying on two actors, a generator and a discriminator, competing against each other. In [192], the authors translated the classical problem in the language of density matrices (described in Section 1.3.1): Given some data represented by a density matrix σ (not necessarily describing a pure state) and a generator generating some output density matrix ρ, the discriminator is tasked with identifying the true data from the fake one. More precisely, it makes a positive operator-valued measurement (Section 1.2.3) with outcomes T (for True) or (for False). The probability that the measurement yields a positive answer given the true data is

$$\mathbb{P}(\mathrm{T}|\sigma) = (\mathrm{T}\sigma),$$

while the probability that it yields a positive answer given generated data is

$$\mathbb{P}(\mathrm{T}|) = (\mathrm{T}\rho).$$

The adversarial game, similarly to the classical case, therefore reads

$$\min_{\mathrm{T}} \max \Big\{ (\mathrm{T}\rho) - (\mathrm{T}\sigma) \Big\}.$$

Note that both the set of positive measurement operators T (with 1-norm less than one) and the set of density matrices ρ are convex, ensuring that the optimisation problem (13.2) admits at least one optimum. However, these two sets are infinite dimensional, making the optimisation problem hard to solve. Following similar arguments, Dallaire-Demers and Killoran [77] further proposed to model both the generator and the discriminator as variational quantum circuits parameterised by some vector of parameters describing, for example, the rotation angles of all the gates. A natural question then is whether some optimal architecture of variational quantum circuit might exist. While there is no clear answer at this stage – as far as we know – recent developments have improved our understanding and the power of such circuits.

Starting from n qubits, a quantum generator : $\mathbb{C}^{2^n} \rightarrow \mathbb{C}^{2^n}$ takes the form of a multi-layer quantum neural network, for example of the following form:

$$:= \prod_{l=L}^{1} U_l(\theta_l).$$

For each layer $l \in \{1,\ldots,L\}$, the unitary gate $U_l(\theta_l)$ acts on all n qubits at the same time, and depends on a vector of parameters (or hyperparameters) θ_l. In order to avoid (too expensive) high-order qubit gates, entanglement takes the form of pairwise controlled unitary gates, and we therefore assume that, for each $l \in \{1,\ldots,L\}$, U_l is composed of one- or two-qubit gates only. One possible (though not the only one) way to parameterise U_l is with the following principles in mind:

- any one-qubit unitary gate can be decomposed into a sequence of three rotation gates R_Z, R_X and R_Y, as proved in [223, Theorem 4.1];

- following [256], *imprimitive* two-qubit gates (i.e., two-qubit gates that map product states to non-product states), together with one-qubit gates ensure quantum universality [47]. In particular the decomposition $R_X(\theta)Q(\varphi)$ is universal [47, Corollary 9.2], for $\theta,\varphi \in [0,2\pi)$, where

$$\mathtt{Q}(\phi) := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \mathrm{e}^{\mathrm{i}\phi} \end{bmatrix}.$$

The general form of the L-layer neural network is therefore (13.2), where each layer gate $U_l(\theta_l)$ takes the form

$$\mathtt{U}_l(\theta_l) = \left\{ \bigotimes_{i=1}^{n} \mathtt{R}_\mathtt{X}(\theta_{\mathrm{e}}^i) \mathtt{Q}^{1+(i \bmod n)}(\theta_{\mathrm{imp}}^i) \right\} \left\{ \left(\bigotimes_{i=1}^{n} \mathtt{R}_\mathtt{Z}(\theta_{\mathtt{Z},l}^i) \right) \left(\bigotimes_{i=1}^{n} \mathtt{R}_\mathtt{X}(\theta_{\mathtt{X},l}^i) \right) \left(\bigotimes_{i=1}^{n} \mathtt{R}_\mathtt{Y}(\theta_{\mathtt{Y},l}^i) \right) \right\},$$

where Q^i means that qubit i is the control qubit and the gate acts on qubit $(i + 1)$. Note that $1 + (i \bmod n) = 1 + i$ when $i \in \{1,\ldots,n-1\}$ and is equal to 1 when $i = n$. The total number of hyperparameters is therefore $5n$ per layer, thus $5nL$ in total. The discriminator itself may or may not be of quantum nature (following a construction similar to the generator), depending on the problem at hand (it is in [18] but not in [268] for example), and the nature of the problem – in particular the potential need to encode/decode data from quantum to classical (with a high cost) may influence this choice.

The finite-dimensional optimisation (13.2) is usually carried out via some gradient descent method; the gradients themselves are computed via separate quantum circuits in [77], or rather – more efficiently – using the parameter-shift rule, explained in Section 8.2.3 (see

also [257]), which allows for an exact computation of the gradient from the original circuit.

QGANs are a very new and active research area, and promise to be one where NISQ-based algorithms will be particularly fruitful. They are intimately linked with developments of QNNs as a whole, and current advances in the field relate to the following, which we encourage the reader to follow closely over the next few years:

- QGAN to generate probability distributions: we refer the interested reader to [18, 268, 314] for univariate distributions, mostly in the context of finance, and to [5, 312] for multivariate distributions;
- Quantum Convolutional Neural Networks: In [160], the authors show how to handle non-linearities in (quantum) deep neural networks; [69, 300] explain how reduce the number of required gates (equivalently, the number of rotation parameters) in the circuit, and [142] highlights the importance and sufficiency of two-qubit interactions, more amenable to NISQ devices;
- Quantum Wasserstein GAN: In [55] – mimicking recent results in classical Wasserstein GANs [13, 121] – the authors introduced a Wasserstein semimetric between quantum data, which they use to reduce the number of required quantum gates.

13.3 Bayesian Quantum Circuit

Parameterised quantum circuits can be used to construct a quantum state with desired properties and to modify it in a controlled way. Measuring the final state is then equivalent to drawing a sample from a probability distribution in a form of a bitstring. This is the key concept behind the Quantum Circuit Born Machine (QCBM) we considered in Chapter 9. The Bayesian Quantum Circuit (BQC) is another quantum generative machine learning model that extends the capabilities of

QCBM [88]. Unlike QCBM that operates only on *data* qubits encoding the desired probability distribution, BQC has additional *ancillary* qubits encoding the *prior* distribution. The BQC circuit is shown in Figure 13.5.

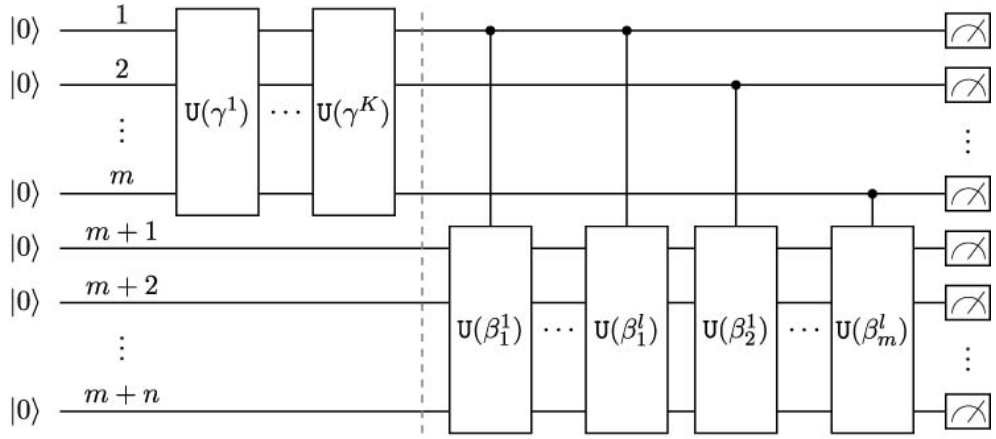


Figure 13.5: Schematic representation of BQC.

The first m quantum registers in the circuit are ancillary qubits. After applying K operator blocks $U(\gamma^i)_{i=1,\ldots,K}$, to the initial state $|0\rangle^{\otimes m}$, we construct the state $|\psi\rangle$,

$$|\psi\rangle = \prod_{i=1}^{K} U(\gamma^i)\ |0\rangle^{\otimes m},$$

and measuring it generates a sample from the prior distribution.

The next n quantum registers are data qubits. Quantum gates operating on them are conditional on the states of the ancillary qubits. Conditionally applying $l \times m$ operator blocks to n data qubits, we obtain a state that is conditional on $|\psi\rangle$. Measuring it will generate a sample from the conditional distribution, which is exactly what is needed to realise a Bayesian model. Bayesian modelling allows us to infer a *posterior* distribution over the parameters θ of the model given some observed data D using Bayes' theorem [57],

$$\mathbb{P}(\theta|D) = \frac{\mathbb{P}(D|\theta)\mathbb{P}(\theta)}{\mathbb{P}(D)} = \frac{\mathbb{P}(D|\theta)\mathbb{P}(\theta)}{\int \mathbb{P}(D|\theta)\mathbb{P}(\theta)\mathrm{d}\theta},$$

where $\mathbb{P}(D|\theta)$ is the likelihood, $\mathbb{P}(D)$ is the marginal likelihood or evidence, and $\mathbb{P}(\theta)$ is the prior. We obtain $\mathbb{P}(\theta)$ by repeatedly measuring the state $|\psi\rangle$ given by (13.3), $\mathbb{P}(D|\theta)$ by repeatedly measuring the final state after applying the conditional operators $U(\beta)$, and $\mathbb{P}(D)$ by repeatedly measuring the final state after applying the operators $U(\beta)$ unconditionally.

In the case of BQC, the prior is parameterised by the parameters $\gamma := (\gamma^1,\ldots,\gamma^K)$. The posterior can be used to model new unseen data, D^*, using the *posterior predictive* [105]:

$$\mathbb{P}(D^*|D) = \int \mathbb{P}(D^*|\theta)\mathbb{P}(\theta|D)\mathrm{d}\theta.$$

This integral averages predictions of all plausible models weighted by posterior probability and is called the *Bayesian model average*.

The BQC can be trained by minimising the maximum mean discrepancy cost function described in Chapter 9. In terms of expressive power, Du, Hsieh, Liu, and Tao [88] showed that a better expressive power of BQC is obtained in comparison with MPQC from a computational complexity perspective.

Bayesian networks can be used for financial asset price forecasting [21, 56], predicting dynamics of limit order book market [199], predicting corporate bankruptcy [52], and to model, analyse, and understand trading behaviour [282].



The Bayesian Quantum Circuit model extends the capabilities of parameterised quantum circuits trained as generative models (QCBM) through the addition of

ancillary quantum registers encoding the prior distribution. As a result, it achieves greater expressive power than MPQC.

13.4 Quantum Semidefinite Programming

In Semidefinite Programming (SDP), one optimises a linear function subject to the constraint that an affine combination of symmetric matrices is positive semidefinite. Such a constraint is non-linear and non-smooth, but convex, so semidefinite programs are convex optimisation problems. Semidefinite programming unifies several standard problems (e.g., linear and quadratic programming) and finds many applications in engineering and combinatorial optimisation [292]. Similarly to finding a quantum counterpart to the classical kernel method, we can specify a quantum version of the SDP.

13.4.1 Classical semidefinite programming

The SDP can be generally defined as the following optimisation problem:

$$\max_{X \in \mathcal{M}_N^+(\mathbb{R})} \operatorname{Tr}(CX), \quad \text{subject to} \quad \operatorname{Tr}(A_j X) \leq b_j, \quad \text{for all } j \in [\![M]\!],$$

where $[[M]] := \{1,\ldots,M\}$, $\mathcal{M}_n^+(\mathbb{R})$ denotes the set of positive semidefinite matrices of size $N \times N$. Here, Hermitian matrices $(A_j)_{j=1,\ldots,M}$ and C in $\mathcal{M}_N(\mathbb{R})$, and $(b_j)_{j\in[[M]]} \in \mathbb{R}^M$ are the inputs of the problem.

SDP can be applied to complex NP-hard optimisation problems [112], such as various portfolio optimisation problems. For example, it is typically an unrealistic assumption that the distribution of asset returns is known exactly. The necessary information may not be complete and

estimates are subject to estimation errors as well as modelling errors (e.g., an assumption of stationarity of the distributions).

13.4.2 Maximum risk analysis

The classical maximum risk analysis problem, assuming there is uncertainty in the estimate of the covariance matrix of asset returns, Σ, can be formulated as

$$\max_{\Sigma \in \mathcal{M}_N^+(\mathbb{R})} \mathrm{w}^\top \Sigma \mathrm{w}, \quad \text{subject to} \quad \Sigma_{ij}^L \leq \Sigma_{ij} \leq \Sigma_{ij}^U, \quad \text{for all } i,j \in [\![N]\!],$$

where w is the fixed vector of weights and Σ is the problem variable. For each $i,j \in [[N]]$, the matrices Σ_{ij}^{L} and Σ_{ij}^{U} are fixed constraints in $\mathcal{M}_N^+(\mathbb{R})$. The task is to establish the maximum possible portfolio risk for the known asset allocation, given uncertainty in the estimate of covariance matrix of asset returns. The problem can be expressed as the following SDP [229]:

$$\max_{\Sigma \in \mathcal{M}_N^+(\mathbb{R})} \mathrm{Tr}\left(\mathrm{w}^\top \Sigma \mathrm{w}\right),$$
$$\text{subject to} \quad \begin{cases} \mathrm{Tr}(-E_{ij}\Sigma) \leq -\Sigma_{ij}^L, \\ \mathrm{Tr}(E_{ij}\Sigma) \leq \Sigma_{ij}^U, \end{cases} \quad \text{for all } (i,j) \in [\![N]\!] \times [\![N]\!],$$

where we denote $(E_{ij})_{\alpha\beta} := \delta_{i\alpha}\delta_{j\beta}$. The maximum risk analysis problem can be expressed in the same form with different risk measures such as VaR or Expected Shortfall.

13.4.3 Robust portfolio construction

The robust portfolio construction problem aims at finding an asset allocation method that would achieve the minimum *estimation error* in the suggested asset allocation weights. This problem has been addressed in [194] using Monte Carlo simulations to determine the

most robust asset allocation method with respect to small changes in the input covariance matrix for the given portfolio.

In the most general case, it can be formulated as the Min-Max problem

$$\min_{\mathrm{w}\in\mathcal{W}} \max_{\Sigma\in\mathcal{S}} \mathrm{w}^{\top}\Sigma\mathrm{w},$$

with

$$\begin{aligned}\mathcal{S} &:= \left\{\Sigma \in \mathcal{M}_{\mathrm{N}}^{+}(\mathbb{R}) : \Sigma_{\mathrm{ij}}^{\mathrm{L}} \leq \Sigma_{\mathrm{ij}} \leq \Sigma_{\mathrm{ij}}^{\mathrm{U}}, \text{ for all } i,j \in [[N]]\right\},\\ \mathcal{W} &:= \left\{\mathrm{w} \in \mathbb{R}^{N} \ : \ \mathbf{1}^{\top}\mathrm{w} = 1, \quad \mu^{\top}\mathrm{w} \geq R_{\min}\right\},\end{aligned}$$

where w is the vector of weights, μ is the vector of expected asset returns, and Σ is the covariance matrix of asset returns.

The following theorem (first proven by von Neumann [297] in 1928) establishes the equivalence of the Min-Max and Max-Min optimisation problems [288]:

Theorem 11 (Minimax Theorem)**.** *Let $\mathcal{X} \subset \mathbb{R}^{\mathrm{n}}$ and $\mathcal{Y} \subset \mathbb{R}^{\mathrm{m}}$ be compact convex sets. If the function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is continuous and concave in x for fixed y and continuous and convex in y for fixed x, then*

$$\min_{y\in\mathcal{Y}} \max_{x\in\mathcal{X}} f(x,y) = \max_{x\in\mathcal{X}} \min_{y\in\mathcal{Y}} f(x,y).$$

Therefore, in general, the Min-Max robust portfolio construction problem (which is convex in w and concave in Σ) is equivalent to the Max-Min problem and can be expressed for the constraints above as an SDP in all variables [229].

13.4.4 Quantum semidefinite programming

The key idea behind Quantum Semidefinite Programming (QSDP) is based on the observation that a normalised positive semidefinite matrix can be naturally represented as a quantum state. Operations on quantum states can sometimes be computationally cheaper to perform on a quantum computer than classical matrix operations. This idea prompted the development of quantum algorithms for SDPs [42].

Consider the SDP (13.4.1) and let $\varepsilon > 0$ be small. An algorithm is called an *ε-approximate quantum SDP oracle* [290] if for all inputs $g \in \mathbb{R}$ and $\zeta \in (0,1)$, it finds, with success probability $1-\zeta$, a vector $y \in \mathbb{R}^{M+1}$ and a real number z such that for the density matrix

$$\rho = \frac{\exp\left(-\sum_{j=1}^{M} y_j \mathrm{A}_j + y_0 \mathrm{C}\right)}{\mathrm{Tr}\left(\exp\left(-\sum_{j=1}^{M} y_j \mathrm{A}_j + y_0 \mathrm{C}\right)\right)},$$

we have that $z\rho$ is an ε-feasible solution with objective value at least $g - \varepsilon$, that is

$$\begin{cases} \mathrm{Tr}(z\rho \mathrm{A}_j) \leq b_j + \varepsilon, & \text{for all } j \in [\![M]\!], \\ \mathrm{Tr}(z\rho \mathrm{C}) \geq g - \varepsilon, \end{cases}$$

or concludes that no such z and y exist even if we set $\varepsilon = 0$.

A general QSDP-solver for sparse matrices was implemented by Brandão and Svore [42] using the Arora-Kale framework [14]. They observed that the density matrix ρ in (13.4.4) is in fact a $\log(N)$-qubit Gibbs state and can be efficiently prepared as a quantum state on a quantum computer.

The reader should already be familiar with the Gibbs state (Gibbs distribution) in the form

$$\rho = \frac{e^{-\beta \mathcal{H}}}{\mathrm{Tr}(e^{-\beta \mathcal{H}})},$$

where $\mathcal{H}$ is the problem Hamiltonian and $\mathrm{Tr}(\exp(-\beta\mathcal{H}))$ is the partition function. The Gibbs (Boltzmann) sampling and the Gibbs (Boltzmann) distribution were discussed in Chapter 5 (in (5.4.1) and (5.4.1)). The form of the partition function in (13.4.4) should not be confusing. Recall from (10.1), that since the Hamiltonian is a Hermitian operator, its spectral decomposition yields the representation

$$\mathcal{H} = \sum_i E_i \left|\psi_i\right\rangle \left\langle\psi_i\right|,$$

which gives the following expression for the Gibbs state:

$$\rho = \frac{e^{-\beta \mathcal{H}}}{Z} = \frac{1}{Z} \sum_i e^{-\beta E_i} \left|\psi_i\right\rangle \left\langle\psi_i\right|,$$

where the partition function Z is given by

$$Z = \mathrm{Tr}\left(e^{-\beta \mathcal{H}}\right) = \sum_i e^{-\beta E_i}.$$

QSDP gives a square-root unconditional speedup over any classical method for solving SDPs both in N and M [42].



Quantum Semidefinite Programming is yet another example where quantum speedup can be achieved since operations on quantum states performed on a quantum computer are less computationally expensive than the corresponding matrix operations on a classical computer.

We would like to finish this chapter (and the book!) with a glance beyond the capabilities of the NISQ computers. The last section presents several important algorithms which, one day, will become the main building blocks of many quantum computing applications.

13.5 Beyond NISQ

In this section, we start with describing the workhorse of many important quantum algorithms, the Quantum Fourier Transform (QFT), before moving to its flagship application, the Quantum Phase Estimation (QPE), and then discussing the possibility of achieving quantum speedup with the Quantum Monte Carlo (QMC) and the Quantum Linear Solver (QLS) algorithms.

13.5.1 Quantum Fourier Transform

In the classical setting, the discrete Fourier transform maps a vector $x := (x_0,\ldots,x_{2^n-1}) \in \mathbb{C}^{2^n}$ to a vector $y := (y_0,\ldots,y_{2^n-1}) \in \mathbb{C}^{2^n}$, the components of which read

$$y_k = \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} \exp\left(\frac{2\pi \mathrm{i} jk}{2^n}\right) x_j, \quad \text{for each } k = 0,\ldots,2^n-1.$$

Similarly, the quantum Fourier transform is the linear map

$$|k\rangle \longmapsto \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} \exp\left(\frac{2\pi \mathrm{i} kj}{2^n}\right) |j\rangle\,,$$

and the operator

$${}_{\mathrm{q}}\mathcal{F} := \frac{1}{\sqrt{2^n}} \sum_{k,j=0}^{2^n-1} \exp\left(\frac{2\pi \mathrm{i} kj}{2^n}\right) |j\rangle\langle k|$$

represents the Fourier transform matrix which is unitary as $_q\mathcal{F}\,_q\mathcal{F}^\dagger = \mathcal{I}$. In an n-qubit system with basis $(|0\rangle,\ldots,|2^n-1\rangle)$, for a given state $|j\rangle$, we use the binary representation

$$j := \overline{j_1 \cdots j_n},$$

with $(j_1,\ldots,j_n) \in \{0,1\}^n$ so that $|j\rangle = |j_1 \cdots j_n\rangle = |j_1\rangle \otimes \ldots \otimes |j_n\rangle$. Likewise, the notation $0.j_1 j_2 \ldots j_n$ represents the binary fraction $\sum_{i=1}^{n} 2^{-i} j_i$. Elementary algebra (see [223, Section 5.1] for details) then yields

$$\begin{aligned} {}_q\mathcal{F}\,|j\rangle = & \frac{1}{\sqrt{2^n}} \left(|0\rangle + e^{2\pi i\, \overline{0.j_n}}\,|1\rangle\right) \otimes \left(|0\rangle + e^{2\pi i\, \overline{0.j_{n-1} j_n}}\,|1\rangle\right) \otimes \cdots \\ & \cdots \otimes \left(|0\rangle + e^{2\pi i\, \overline{0.j_1 \ldots j_n}}\,|1\rangle\right). \end{aligned}$$

13.5.2 Quantum Phase Estimation

The goal of QPE is to estimate the unknown phase $\phi \in [0,1)$ for a given unitary operator $\mathcal{U}$ with an eigenvector $|u\rangle$ and eigenvalue $\exp(2\pi i\phi)$. Consider a register of size m and define

$$b^* := \sup_{j \le 2^m \varphi} \left\{ j = 2^m\, \overline{0.j_1 \cdots j_m} \right\}.$$

Thus with $b^* = b_1 \cdots b_m$, we obtain that $2^{-m} b^* = 0.b_1 \cdots b_m$ is the best m-bit approximation of ϕ from below. The QPE procedure uses two registers, with the first containing m qubits initially in the state $|0\rangle$. Selecting m relies on the number of digits of accuracy for the estimate of ϕ, and the probability with which we wish to obtain a successful phase estimation procedure.

QPE allows us to implement a measurement for any Hermitian operator. Note that we always measure individual qubits. If we want to measure a more complex observable, we can use a QPE that implements the von Neumann's measurement scheme [212]. The

routine prepares an eigenstate of the Hermitian operator in one register and stores the corresponding eigenvalue in a second register.

Up to a SWAP transformation, the quantum phase circuit [223, Section 5.2] gives the output

$$|\psi\rangle = \frac{1}{\sqrt{2^m}} \left(|0\rangle + e^{2\pi i \overline{0.\varphi_m}} |1\rangle \right) \otimes \left(|0\rangle + e^{2\pi i \overline{0.\varphi_{m-1}\varphi_m}} |1\rangle \right) \otimes \cdots$$
$$\cdots \otimes \left(|0\rangle + e^{2\pi i \overline{0.\varphi_1 \ldots \varphi_m}} |1\rangle \right),$$

which is exactly equal to the QFT for the state $|2^m\varphi\rangle = |\varphi_1\varphi_2\cdots\varphi_m\rangle$ as in (13.5.1), and therefore $|\psi\rangle = q\mathcal{F}|2^m\varphi\rangle$. Since the QFT is a unitary transformation, we can inverse the process to retrieve $|2^m\varphi\rangle$. Algorithm 10 below provides pseudocode for the QPE procedure, and we refer the interested reader to [223, Chapter 5.2] for detailed explanations.

Algorithm 10: Quantum Phase Estimation

Input:

Unitary matrix (gate) $\mathtt{U}$ with $\mathtt{U}\,|\mathrm{u}\rangle = \mathrm{e}^{2\pi \mathrm{i}\varphi}\,|\mathrm{u}\rangle$;

m ancilla qubits initialised at $|0\rangle$.

1: Prepare the initial state with $|0\rangle^{\otimes m}$ being the m-qubit ancilla register and $|\mathrm{u}\rangle$ being the n-qubit eigenstate register.

2: Map to

$$\frac{1}{\sqrt{2^m}} \sum_{j=0}^{2^m-1} |j\rangle\,|\mathrm{u}\rangle$$

with Hadamard gates applied to the ancilla register.

3: Map to

$$\frac{1}{\sqrt{2^m}} \sum_{j=0}^{2^m-1} |j\rangle\,\mathtt{U}^j\,|\mathrm{u}\rangle = \frac{1}{\sqrt{2^m}} \sum_{j=0}^{2^m-1} |j\rangle\,\mathrm{e}^{2\pi \mathrm{i} j\varphi}\,|\mathrm{u}\rangle$$

with Controlled $\mathtt{U}^j$ gates applied to the eigenstate register.

4: Compute

$$|\widetilde{\varphi}\rangle\,|\mathrm{u}\rangle$$

using the inverse QFT, where $\widetilde{\varphi}$ is an m-qubit approximation of φ.

5: Measure to deduce $\widetilde{\varphi}$.

Result: Phase estimate $\widetilde{\varphi}$.

13.5.3 Monte Carlo speedup

Leveraging on the speedup provided by the Quantum Phase Estimation, Montanaro [216] devised a Monte Carlo scheme providing quantum speedup compared to the classical one.

Classical Monte Carlo

Monte Carlo techniques represent a wide array of methods to simulate statistics of random processes. We refer the interested reader to the excellent monograph [111] for a full description and analysis. Consider a one-dimensional random variable X and a function $\varphi : \mathbb{R} \to [0,1]$ such that both $\mathfrak{p} := \mathbb{E}[\varphi(X)]$ and $\sigma^2 := \mathbb{V}[\varphi(X)]$ are well defined. By the Central Limit Theorem, given an iid collection of random variables $(X_1,\ldots,X_N)$ distributed as X, then

$$\sqrt{N}\frac{\widehat{\mathfrak{p}}_N - \mathfrak{p}}{\sigma}$$

converges to a centered Gaussian with unit variance $\mathcal{N}(0,1)$ as N tends to infinity, where $\mathfrak{p}_N := \frac{1}{N}\sum_{i=1}^{N} X_i$ is the empirical mean. This implies that, for any $\varepsilon > 0$, we can estimate

$$\mathbb{P}\left(\left|\widehat{\mathfrak{p}}_N - \mathfrak{p}\right| \leq \varepsilon\right) = \mathbb{P}\left(|\mathcal{N}(0,1)| \leq \frac{\varepsilon\sqrt{N}}{\sigma}\right),$$

so that, for any $z > 0$ and $\delta \in (0,1)$, in order to get an estimate of the form $\mathbb{P}\left(\left|\widehat{\mathfrak{p}}_N - \mathfrak{p}\right| \leq z\right) = 1 - \delta$, we need $N = \mathcal{O}(1/\varepsilon^2)$ samples.

Quantum Monte Carlo

Consider now an operator $\mathcal{A}$ of the form

$$\mathcal{A}\,|0\rangle^{\otimes n} = \sum_{x\in\{0,1\}^k} \alpha_x\,|\psi_x\rangle\,|x\rangle\,,$$

for some $k \leq n$, where each $|\psi_x\rangle$ is a quantum state with $n-k$ qubits and $|x\rangle$ a quantum state with k qubits, and $\alpha_x \in \mathbb{C}$ is some amplitude, the meaning of which will be made clear below. We simply assume that $\{|\psi_x\rangle\}_{x\in\{0,1\}^k}$ forms an orthogonal family and are in fact "garbage qubits", i.e., qubits that are, for example, used as controlled to build the solution vector $|x\rangle$ from the data. Given the encoded data $|x\rangle$, assume further the existence of the operator $\mathcal{W}$:

$$\mathcal{W}\,|\mathrm{x}\rangle\,|0\rangle = |\mathrm{x}\rangle\left(\sqrt{1-\phi(\mathrm{x})}\,|0\rangle + \sqrt{\phi(\mathrm{x})}\,|1\rangle\right).$$

This can be achieved for example by using the following lemma.

Lemma 9 (Conditional Rotation. Theorem 3.5 in [184]). *Given a quantum state* $|\psi_a\rangle$*, encoding* $a \in [-1,-1]$ *in q qubits, there exists a quantum circuit performing the unitary mapping* $|\psi_a\rangle|0\rangle \longmapsto |\psi_a\rangle\left(a|0\rangle + \sqrt{1-a^2}|1\rangle\right)$.

Consider now the operator $\mathcal{M}$:

$$\mathcal{M} := \left(\mathcal{I}^{n-k} \otimes \mathcal{W}\right)\left(\mathcal{A} \otimes \mathcal{I}\right),$$

where $\mathcal{I}^{n-k}$ means the identity operator acting on $n-k$ qubits, so that

$\|\psi\rangle$	$:= \mathcal{M}\|0\rangle^{\otimes(n+1)}$		
	$= \left(\mathcal{I}^{n-k} \otimes \mathcal{W}\right)\left(\sum_{\mathrm{x}\in\{0,1\}^k} \alpha_{\mathrm{x}}\,|\psi_{\mathrm{x}}\rangle\,|\mathrm{x}\rangle\right)|0\rangle$		
	$= \sum_{\mathrm{x}\in\{0,1\}^k} \alpha_{\mathrm{x}} \left(\mathcal{I}^{n-k} \otimes \mathcal{W}\right)|\psi_{\mathrm{x}}\rangle|\mathrm{x}\rangle|0\rangle$		
	$= \sum_{\mathrm{x}\in\{0,1\}^k} \alpha_{\mathrm{x}} |\psi_{\mathrm{x}}\rangle|\mathrm{x}\rangle\left(\sqrt{1-\phi(\mathrm{x})}\,|0\rangle + \sqrt{\phi(\mathrm{x})}\,|1\rangle\right)$		
	$=: |\Psi_B\rangle|0\rangle + |\Psi_G\rangle|1\rangle,$	(13.5.1)	

where $|\Psi_B\rangle$,

$$|\Psi_B\rangle := \sum_{\mathrm{x}\in\{0,1\}^k} \alpha_{\mathrm{x}}\sqrt{1-\phi(\mathrm{x})}\,|\psi_{\mathrm{x}}\rangle\,|\mathrm{x}\rangle\,,$$

stands for the 'bad' state, and $|\Psi_G\rangle$,

$$|\Psi_G\rangle := \sum_{\mathrm{x}\in\{0,1\}^k} \alpha_{\mathrm{x}}\sqrt{\phi(\mathrm{x})}\,|\psi_{\mathrm{x}}\rangle\,|\mathrm{x}\rangle\,,$$

stands for the 'good' state.

Consider now the projector $\mathcal{P} := \mathcal{I}^{n}|1\rangle\langle 1|$ and measure the probability of the last qubit of $|\psi\rangle$ to be in state $|1\rangle$, namely

$$\begin{aligned}
\langle\psi|\mathcal{P}^{\dagger}\mathcal{P}|\psi\rangle &= \langle\psi|\mathcal{P}|\psi\rangle \\
&= \Big(\langle 0|\langle\Psi_B| + \langle 1|\langle\Psi_G|\Big)\mathcal{P}\Big(|\Psi_B\rangle|0\rangle + |\Psi_G\rangle|1\rangle\Big) \\
&= \Big(\langle 0|\langle\Psi_B| + \langle 1|\langle\Psi_G|\Big)\Big(|\Psi_B\rangle|1\rangle\langle 1|0\rangle + |\Psi_G\rangle|1\rangle\langle 1|1\rangle\Big) \\
&= \Big(\langle 0|\langle\Psi_B| + \langle 1|\langle\Psi_G|\Big)|\Psi_G\rangle|1\rangle \\
&= \langle 0|\langle\Psi_B|\Psi_G\rangle|1\rangle + \langle 1|\langle\Psi_G|\Psi_G\rangle|1\rangle \\
&= \langle\Psi_B|\Psi_G\rangle\langle 0|1\rangle + \langle\Psi_G|\Psi_G\rangle\langle 1|1\rangle = |\Psi_G|^2.
\end{aligned}$$

Now, since the family $\{|\psi_x\rangle\}_x$ is orthogonal, it is easy to see from (13.5.3) that

$$\begin{aligned}
|\Psi_G|^2 &= \langle\Psi_G|\Psi_G\rangle \\
&= \left(\sum_{x\in\{0,1\}^k} \alpha_x^* \sqrt{\phi(x)}\,\langle x|\,\langle\psi_x|\right)\left(\sum_{y\in\{0,1\}^k} \alpha_y \sqrt{\phi(y)}\,|\psi_y\rangle\,|y\rangle\right) \\
&= \sum_{x,y\in\{0,1\}^k} \alpha_x^* \alpha_y \sqrt{\phi(x)}\sqrt{\phi(y)}\langle x|\langle\psi_x|\psi_y\rangle|y\rangle \\
&= \sum_{x\in\{0,1\}^k} |\alpha_x|^2 \varphi(x),
\end{aligned}$$

which corresponds precisely to the expectation $\mathbb{E}[\varphi(X)]$ where the random variable X is discretised over the set with labels $\{0,1\}^k$, and where each $|\alpha_x|^2$ corresponds to the discrete probability of X being in x.

In order to retrieve the expectation we are after, we therefore simply need to run the circuit corresponding to $\mathcal{M}$, measure the output in the

computational basis, and determine the probability of observing the state $|1\rangle$.

QMC speedup

The actual speedup of QMC resides in a subtle application of the *Amplitude Estimation* theorem and the *Powering Lemma*, which we present now.

Theorem 12 (Amplitude Estimation. Theorem 12 in [43]). *Assume that we have access to a quantum unitary operator* $\mathcal{U}$ *such that* $\mathcal{U}|0\rangle = \sqrt{1-\mathfrak{p}}\,|\Psi_B\rangle|0\rangle + \sqrt{\mathfrak{p}}\,|\Psi_G\rangle|1\rangle$, *for some states* $|\Psi_B\rangle, |\Psi_G\rangle$. *Then, for any* $N \in \mathbb{N}$, *the amplitude estimation algorithms outputs the estimate* $\mathfrak{p}$ *such that*

$$\left|\widehat{\mathfrak{p}} - \mathfrak{p}\right| \leq 2\pi \frac{\sqrt{\mathfrak{p}(1-\mathfrak{p})}}{N} + \frac{\pi^2}{N^2}$$

with probability at least $8/\pi^2$. *To achieve this takes exactly* N *iterations.*

Lemma 10 (Powering Lemma. Lemma 6.1 in [150]). *Let* $\mathfrak{p}$ *be a quantity to estimate and* $\mathcal{U}$ *an algorithm that output* $\mathfrak{p}$ *such that* $\left|\widehat{\mathfrak{p}} - \mathfrak{p}\right| \leq \varepsilon$ *except with probability smaller than* $1/2$. *Then, for any* $\delta \in (0,1)$, *it suffices to repeat* $\mathcal{U}$ *about* $\mathcal{O}(\log(1/\delta))$ *times and to take the median to obtain* $\left|\widehat{\mathfrak{p}} - \mathfrak{p}\right| \leq \varepsilon$ *with probability at least* $1-\delta$.

In light of (13.5.1), the Amplitude Estimation theorem, combined with the Powering Lemma, shows that in order to obtain an estimate of the empirical mean

$$\langle\psi|\,\mathcal{P}^\dagger\mathcal{P}\,|\psi\rangle = |\Psi_G|^2$$

with probability at least $1-\delta$ (for any $\delta \in (0,1)$), i.e.,

$$\mathbb{P}\left(\left|\widehat{\mathfrak{p}} - \mathfrak{p}\right| \leq \varepsilon\right) \geq 1-\delta,$$

it suffices to apply the operators $\mathcal{M}$ and $\mathcal{P}$ about $\mathcal{O}(N \log(1/\delta))$ times, with

$$\varepsilon = 2\pi \frac{\sqrt{\mathfrak{p}(1-\mathfrak{p})}}{N},$$

so that, for any fixed $\delta \in (0,1)$, the computational cost is of order $\mathcal{O}(1/\varepsilon)$, achieving quadratic speedup compared to classical Monte Carlo.

13.5.4 Quantum Linear Solver

Harrow, Hassidim and Lloyd [126] devised a quantum algorithm to solve linear systems, beating classical computation times. Linear systems are ubiquitous in applications, and many aspects of quantitative finance rely on being able to solve such (low- or high-dimensional) systems. We highlight below two key examples of fundamental importance in finance: solving Partial Differential Equations (PDEs) and portfolio optimisation.

Theoretical aspects

The problem can be stated as follows: given a matrix $A \in \mathcal{M}_N(\mathbb{C})$ and a vector $b \in \mathbb{C}^N$, find the vector $x \in \mathbb{C}^N$ such that

$$Ax = b.$$

In order for the algorithm to work, the matrix A needs to be Hermitian. If A is not so, we can nevertheless consider the augmented system

$$\begin{pmatrix} \mathbf{0}_{N,N} & A \\ A^\dagger & \mathbf{0}_{N,N} \end{pmatrix} \begin{pmatrix} \mathbf{0}_{N,1} \\ x \end{pmatrix} = \begin{pmatrix} b \\ \mathbf{0}_{N,1} \end{pmatrix},$$

similarly to the Hamiltonian embedding in Section 7.6. We assume from now now that A is indeed Hermitian. The first step of the

algorithm is to assume that the vector b can be encoded into a quantum state $|\mathrm{b}\rangle$ and to then rewrite (13.5.4) as

$$\mathrm{A}\,|\mathrm{x}\rangle = |\mathrm{b}\rangle\,,$$

where we now look for the solution, not as an element of $\mathbb{C}^{\mathrm{N}}$, but as a quantum state.

Since A is Hermitian, it admits the spectral decomposition (Section 1.1.5)

$$\mathrm{A} = \sum_{j=0}^{N-1} \lambda_j\, |\phi_j\rangle\, \langle\phi_j|\,,$$

where $\lambda_0,\ldots,\lambda_{\mathrm{N}-1}$ are its (not necessarily distinct) strictly positive eigenvalues with corresponding eigenstates $|\phi_0\rangle,\ldots,|\phi_{N-1}\rangle$, and we immediately obtain that its inverse reads

$$\mathrm{A}^{-1} = \sum_{j=0}^{N-1} \frac{1}{\lambda_j}\, |\phi_j\rangle\, \langle\phi_j|\,.$$

We can also decompose $|\mathrm{b}\rangle$ into the $(|\phi_j\rangle)_{\mathrm{j}=0,\ldots,\mathrm{N}-1}$ basis as

$$|\mathrm{b}\rangle = \sum_{j=0}^{N-1} b_i\, |\phi_j\rangle\,,$$

and therefore the solution to (13.5.4) reads

$$|\mathrm{x}\rangle = \mathrm{A}^{-1}\,|\mathrm{b}\rangle = \sum_{j=0}^{N-1} \frac{b_j}{\lambda_j}\, |\phi_j\rangle\,.$$

The goal of the QLS algorithm is thus to construct such a state, and we summarise it as Algorithm 11 below. Note that, since A is Hermitian then, for any $t \in \mathbb{R}$, $\mathrm{U} := \exp(\mathrm{i}\mathrm{A}t)$ is unitary with decomposition

$$\mathtt{U} = \sum_{j=0}^{N-1} \mathrm{e}^{\mathrm{i}\lambda_j t} \left|\phi_j\right\rangle \left\langle\phi_j\right| .$$

In total, the QLS algorithm requires $n_l + n_b + 1$ qubits, where n_l is the number of qubits used to encode the n_l-bit binary representation of $(\lambda_j)_{j=0,\ldots,N-1}$ and n_b is the number of qubits used to convert b into $|b\rangle$ (and also the number of qubits to write the solution state).

In terms of computation time, Harrow, Hassidim, and Lloyd showed that the stated runtime is of order poly(log(N),κ) assuming that A is sparse with condition number κ, which yields an exponential speedup compared to the classical $\mathcal{O}(N\sqrt{\kappa})$ runtime.

Algorithm 11: HHL Quantum Linear Solver

Input: Hermitian matrix A and $n_l + n_b + 1$ qubits initialised at $|0\rangle^{\otimes n_l} |0\rangle^{\otimes n_b} |0\rangle$.

1: Load the data b into $|\mathrm{b}\rangle$ using n_b qubits (with $N = 2^{n_b}$).

2: Apply QPE with $\mathtt{U} := \exp(\mathrm{iA}t)$, after which the quantum state of the register is

$$\sum_{j=0}^{N-1} b_j \, |\lambda_j\rangle_{n_l} \, |\phi_j\rangle_{n_b} \, |0\rangle \, .$$

3: Rotate the ancillary qubit $|0\rangle$ controlled by $|\lambda_j\rangle_{n_l}$ to obtain

$$\sum_{j=0}^{N-1} b_j \, |\lambda_j\rangle_{n_l} \, |\phi_j\rangle_{n_b} \left(\sqrt{1 - \frac{C^2}{\lambda_j^2}} \, |0\rangle + \frac{C}{\lambda_j} \, |1\rangle \right),$$

for some normalising constant C (with $|C| < \min_j \lambda_j$).

4: Apply the inverse QPE to obtain

$$\sum_{j=0}^{N-1} b_j \, |0\rangle_{n_l} \, |\phi_j\rangle_{n_b} \left(\sqrt{1 - \frac{C^2}{\lambda_j^2}} \, |0\rangle + \frac{C}{\lambda_j} \, |1\rangle \right).$$

5: Measure the ancillary qubit in the computational basis. If the outcome is $|1\rangle$, the register is in the post-measurement state

$$C \sum_{j=0}^{N-1} \frac{b_i}{\lambda_i} \, |0\rangle_{n_l} \, |\phi_j\rangle_{n_b} \, ,$$

which up to a normalisation factor corresponds to the solution.

Result: Solution $|\mathrm{x}\rangle$:

$$|\mathrm{x}\rangle = \mathrm{A}^{-1} \, |\mathrm{b}\rangle = \sum_{j=0}^{N-1} \frac{b_j}{\lambda_j} \, |\phi_j\rangle \, .$$

Solving PDEs

One important example is finite-difference schemes for partial differential equations; standard tools can be consulted in [269] for example, and specific applications to finance can be found in [89].

Consider for example the Black-Scholes parabolic PDE:

$$\partial_t V_t + rS\partial_S V_t + \frac{\sigma^2}{2} S^2 \partial^2_{SS} V_t = rV_t,$$

with boundary condition $V_T(S)$ (for instance for a European call option with maturity $T > 0$ and strike $K > 0$, we have $V_T(S) = (S_T - K)_+ := \max(S_T - K, 0)$). Before trying to solve it, it is standard to simplify it. Let $\tau := T - t$ and define $g_\tau(S) := V_t(S)$, then $\partial_t V_t(S) = -\partial_\tau g_\tau(S)$ and hence

$$-\partial_\tau g_\tau + rS\partial_S g_\tau + \frac{\sigma^2}{2} S^2 \partial^2_{SS} g_\tau = r g_\tau,$$

with boundary condition $g_0(S)$. Introduce now $f_\tau(S) := e^{r\tau} g_\tau(S)$, so that

$$-\partial_\tau f_\tau + rS\partial_S f_\tau + \frac{\sigma^2}{2} S^2 \partial^2_{SS} f_\tau = 0,$$

with boundary condition $f_0(S)$. The transformation $x := \log(S)$ and the map $\psi_\tau(x) := f_\tau(S)$ yield, after simplifications,

$$-\partial_\tau \psi_\tau + \left(r - \frac{\sigma^2}{2}\right) \partial_x \psi_\tau + \frac{\sigma^2}{2} \partial^2_{xx} \psi_\tau = 0,$$

with boundary condition $\psi_0(x)$. Finally, setting φ_τ via $\psi_\tau(x) =: e^{\alpha x + \beta \tau} \varphi_\tau(x)$ with

$$\alpha := -\frac{1}{\sigma^2}\left(r - \frac{\sigma^2}{2}\right) \quad \text{and} \quad \beta := -\frac{1}{2\sigma^2}\left(r - \frac{\sigma^2}{2}\right)^2,$$

implies that equation (13.5.4) becomes the heat equation

$$\partial_\tau \phi_\tau(x) = \frac{\sigma^2}{2} \partial^2_{xx} \phi_\tau(x),$$

for all $x \in \mathbb{R}$ with (Dirichlet) boundary condition $\varphi_0(x) = e^{-\alpha x}\psi_0(x)$.

We now discretise this PDE using an explicit scheme, where the time derivative ∂_τ is evaluated by forward difference while the space derivative ∂_{xx} is approximated with a central difference scheme (implicit schemes or more general θ-schemes follow a similar logic). We consider (13.5.4) for $\tau > 0$ and x in some interval $[x_L, x_U] \in \mathbb{R}$, with (Dirichlet) boundary conditions $\varphi(0,x) = f(x)$ (payoff at maturity), $\varphi(\tau, x_L) = f_L(\tau)$, and $\varphi(\tau, x_U) = f_U(\tau)$.

We start by constructing the time-space grid for the approximation scheme. For two integers m and n, we consider a uniform grid, i.e., we split the space axis into m intervals and the time axis into n intervals, and we denote $\mathcal{V} := \{0,1,\ldots,n\}$ and $\mathcal{W} := \{0,1,\ldots,m\}$. This means that each point on the grid has coordinates $(i\delta_T, x_L + j\delta_x)$ for $i \in \mathcal{V}$ and $j \in \mathcal{W}$, where

$$\delta_T := \frac{T}{n} \quad \text{and} \quad \delta_x := \frac{x_U - x_L}{m}.$$

At each node, we let $\varphi_{i,j} := \varphi(i\delta_T, x_L + j\delta_x)$ denote the value of the function u. Note in particular that the boundary conditions imply

$$\phi_{0,j} = f(x_L + j\delta_x), \quad \phi_{i,0} = f_L(i\delta_T), \quad \phi_{i,m} = f_U(i\delta_T).$$

More precisely we consider the following approximations

$$\partial_\tau \varphi(\tau,x) = \frac{\phi(\tau + \delta_T, x) - \phi(\tau, x)}{\delta_T} + \mathcal{O}(\delta_T),$$

$$\partial_{xx} \varphi(\tau,x) = \frac{\phi(\tau, x + \delta_x) - 2\phi(\tau, x) + \phi(\tau, x - \delta_x)}{\delta_x^2} + \mathcal{O}(\delta_x^2).$$

Ignoring the terms in δ_T and δ_x^2, the heat equation at the node $(i\delta_T, x_L + j\delta_x)$ becomes

$$\frac{\phi_{i+1,j} - \phi_{i,j}}{\delta_T} + \mathcal{O}(\delta_T) = \frac{\sigma^2}{2} \frac{\phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1}}{\delta_x^2} + \mathcal{O}(\delta_x^2),$$

which we can rewrite

$$\phi_{i+1,j} = \frac{\delta_T}{\delta_x^2} \frac{\sigma^2}{2} \phi_{i,j+1} + \left(1 - \frac{\delta_T}{\delta_x^2} \sigma^2\right) \phi_{i,j} + \frac{\delta_T}{\delta_x^2} \frac{\sigma^2}{2} \phi_{i,j-1},$$

for all $i = 0,\ldots,n-1$, $j = 1,\ldots,m-1$. To rewrite this in matrix form, introduce for each $i = 0,\ldots,n$, $_i \in \mathbb{R}^{m-1}$, $B_i \in \mathbb{R}^{m-1}$ and the matrix $A \in \mathcal{M}_{m-1}(\mathbb{R})$ by

$$\begin{aligned} {}_i &:= (\phi_{i,1}, \ldots, \phi_{i,m-1})^\top, \\ B_i &:= (\phi_{i,0}, 0, \ldots, 0, \phi_{i,m})^\top, \\ A &:= T_{m-1}\left(1 - \alpha\sigma^2, \frac{\alpha\sigma^2}{2}, \frac{\alpha\sigma^2}{2}\right), \end{aligned}$$

where

$$\alpha := \frac{\delta_T}{\delta_x^2}$$

and where $T_{m-1}(\cdot)$ denotes the tridiagonal matrix of dimension $(m-1) \times (m-1)$.

The recursion (13.5.4) thus becomes

$$_{i+1} = A_i + \frac{\alpha\sigma^2}{2} B_i, \quad \text{for each } i = 0, \ldots, n-1,$$

with the time boundary condition

$$_0 = (\phi_{0,1}, \ldots, \phi_{0,m-1})^\top = (f(x_L + \delta_x), \ldots, f(x_L + (m-1)\delta_x))^\top.$$

Leaving the boundary term B_i aside, the recursion (13.5.4) thus is exactly of the form (13.5.4), and can therefore be tackled using the HHL algorithm.

This is the obvious first step to investigate the use of HHL-type algorithms in quantitative finance, and further developments have already been proposed in [104, 108, 188, 310], with or without finance applications in mind.

Application to portfolio optimisation

The second immediate application of QLS in finance is for portfolio optimisation. Indeed, the standard Markowitz-type problem of the form (3.3) in Section 3.3 is readily formulated (at least for weights in {0,1}) as a linear problem, the constraints only increasing the dimension as Lagrange multipliers. We shall not dive into the details here as this is a rather novel development with huge potential but limited results so far, and instead refer the reader to [306, 187] for promising implementations and details.

Summary

In this chapter, we introduced several new promising quantum algorithms. First, we learned about quantum kernels, which can replace classical kernels in hybrid quantum-classical protocols.

Next, we introduced the Bayesian quantum circuit model that expands the concept of a Bayesian neural network to parameterised quantum circuits. BQC is a promising generative model with larger expressive power than QCBM/MPQC (covered in Chapters 9 and 12).

Then we looked at quantum SDP and its potential to outperform classical SDP. This is a topic of active research.

Finally, we covered several important quantum algorithms that rely on the existence of a quantum computing hardware with characteristics that exceed the capabilities of currently available NISQ computers. However, the very presence of these algorithms and their potential to achieve quadratic or even exponential speedup provides strong motivation for the rapid development of quantum computers.

This chapter completes the book. Looking ahead, we see a bright future for quantum computing. In the update to their quantum computing development roadmap [145], IBM outlined an exciting vision with the goal to build quantum-centric supercomputers. The latter will incorporate quantum processors, classical processors, quantum communication networks, and classical networks. The immediate deliverables are expected to be the 433-qubit *Osprey* processor (expected to be released in 2022) and the 1,121-qubit *Condor* processor (expected to be released in 2023). The next step will be to develop ways to link processors together into a modular system capable of scaling without physical limitations.

The modular, multi-chip scaling technology is also envisaged by Rigetti. Rigetti anticipates the launch of their next generation single-chip 84-qubit quantum computer in 2023 and 336-qubit multi-chip processor later in 2023. The 336-qubit multi-chip processor is expected to combine the anticipated improvements of the 84-qubit processor with the modular, multi-chip scaling technology of Rigetti's Aspen-M machine. These machines are expected to deliver increased performance across the key dimensions of speed, scale, and fidelity [246].

We also expect to see significant progress in the trapped ion space. IonQ announced several major breakthroughs that may have a major

impact on the way quantum algorithms are designed and run on trapped ion quantum computing hardware. This includes, for example, a new family of *n*-qubit gates, such as the *n*-qubit Toffoli gate, which flips a select qubit if and only if all the other qubits are in a particular state. Unlike standard two-qubit quantum computing gates, the *n*-qubit Toffoli gate acts on many qubits at once, leading to more efficient operations [146].

Quantum annealing is going from strength to strength. In a recent white paper [39], D-Wave introduced the new *Zephyr* graph with better connectivity than its predecessors, *Chimera* and *Pegasus*. Plans are in place for a 7,000-qubit chip based on *Zephyr*, scheduled to be available in 2023-2024 [93]. Early benchmarks with smaller-scale prototype systems consisting of 500+ qubits have demonstrated more compact embedding, lower error rates, improved solution quality, and an increased probability of finding optimal solutions [210].

But, ultimately, it is up to the users to try and test various hardware and software solutions on a variety of use cases. We encourage our readers to experiment and apply the methods of quantum computing to their own spheres of interest and discover new quantum algorithms and applications. This is an exciting journey and a great opportunity to participate in the collective effort of achieving quantum advantage for the benefits of wider society.

Bibliography

[1] A. Abbas, D. Sutter, C. Zoufal, A. Lucchi, A. Figalli and S. Woerner. The power of quantum neural networks. *Nature Computational Science*, `1`, 2021.

[2] D.M. Abrams, N. Didier, B.R. Johnson, M.P. da Silva and C.A. Ryan. Implementation of the XY interaction family with calibration of a single pulse. *Nature Electronics*, `3`, 2020.

[3] D.H. Ackley, G.E. Hinton and T.J. Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive Science*, `9`(1), 1985.

[4] S. Adachi and M. Henderson. Application of quantum annealing to training of deep neural networks. `arXiv:1510.06356`, 2015.

[5] G. Agliardi and E. Prati. Optimal tuning of quantum generative adversarial networks for multivariate distribution loading. *Quantum Reports*, `4`(1), 2022.

[6] A. Agresti. Categorical Data Analysis. Wiley, 3rd Edition, 2013.

[7] D. Aharonov, W. van Dam, J. Kempe, Z. Landau, S. Lloyd and O. Regev. Adiabatic quantum computation is equivalent to standard quantum computation. *SIAM Review*, `50`(4), 2008.

[8] D. Aharonov and L. Zhou. Hamiltonian sparsification and gap-simulations. `arXiv:1804.11084`, 2018.

[9] O. Akbilgic, H. Bozdogan and M.E. Balaban. Istanbul Stock Exchange Dataset. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml. UC Irvine, School of Information and Computer Science, 2013.

[10] O. Akbilgic, H. Bozdogan and M.E. Balaban. A novel Hybrid RBF Neural Networks model as a forecaster. *Statistics and Computing*, 24(3), 2013.

[11] M. Amin and M. Steininger. Adiabatic quantum computation with superconducting qubits. US Patent US7135701B2, 2006.

[12] K.P. Anagnostopoulos and G. Mamanis. A portfolio optimization model with three objectives and discrete variables. *Computers & Operations Research*, 37(7), 2010.

[13] M. Arjovsky, S. Chintala and L. Bottou. Wasserstein generative adversarial networks. International conference on machine learning, 2017.

[14] S. Arora and S. Kale. A combinatorial, primal-dual approach to semidefinite programs. *Journal of the ACM*, 63(2), 2016.

[15] S. Arunachalam, V. Gheorghiu, T. Jochym-O'Connor, M. Mosca and P.V. Srinivasan. On the robustness of bucket brigade quantum RAM. *New Journal of Physics*, 17(12), 2015.

[16] F. Arute, K. Arya, R. Babbush, D. Bacon, J.C. Bardin, R. Barends, R. Biswas, S. Boixo, F.G.S.L. Brandao, D.A. Buell, B. Burkett, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, A. Fowler, C. Gidney, M. Giustina, R. Graff, K. Guerin, S. Habegger, M.P. Harrigan, M.J. Hartmann,

A. Ho, M. Hoffmann, T. Huang, T.S. Humble, S.V. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, J. Kelly, P.V. Klimov, S. Knysh, A. Korotkov, F. Kostritsa, D. Landhuis, M. Lindmark, E. Lucero, D. Lyakh, S. Mandrà, J.R. McClean, M. McEwen, A. Megrant, X. Mi, K. Michielsen, M. Mohseni, J. Mutus, O. Naaman, M. Neeley, C. Neill, M.Y. Niu, E. Ostby, A. Petukhov, J.C. Platt, C. Quintana, E.G. Rieffel, P. Roushan, N.C. Rubin, D. Sank, K.J. Satzinger, V. Smelyanskiy, K.J. Sung, M.D. Trevithick, A. Vainsencher, B. Villalonga, T. White, Z.J. Yao, P. Yeh, A. Zalcman, H. Neven and J.M. Martinis. Quantum supremacy using a programmable superconducting processor. *Nature*, `574`, 2019.

[17] S.A. Assefa, D. Dervovic, M. Mahfouz, R.E. Tillman, P. Reddy and M. Veloso. Generating synthetic data in finance: opportunities, challenges and pitfalls. Proceedings of the First ACM International Conference on AI in Finance, 2020.

[18] A. Assouel, A. Jacquier and A. Kondratyev. A quantum generative adversarial network for distributions. *Quantum Machine Intelligence*, `4`(2), 2022.

[19] J.E. Avron and A. Elgart. Adiabatic theorem without a gap condition. *Communications In Mathematical Physics*, `203`(445), 1999.

[20] J.E. Avron, R. Seiler and L.G. Yaffe. Adiabatic theorems and applications to the quantum Hall effect. *Communications In Mathematical Physics*, `110`(1), 1987.

[21] A. Back and W. Keith. Bayesian Neural Networks for Financial Asset Forecasting. Degree Project in Mathematics, KTH Royal Institute of Technology, School of Engineering Sciences, 2019.

[22] P. Baldi and R. Vershynin The capacity of feedforward neural networks. *Neural networks*, `116`, 2019.

[23] V. Bapst, L. Foini, F. Krzakala, G. Semerjian and F. Zamponi. The quantum adiabatic algorithm applied to random optimization problems: the quantum spin glass perspective. *Physics Reports*, `523`(127), 2013.

[24] F. Barahona. On the computational complexity of Ising spin glass models. *Journal of Physics A*, `15`(10), 1982.

[25] B. Barak, A. Moitra, R. O'Donnell, P. Raghavendra, O. Regev, D. Steurer, L. Trevisan, A. Vijayaraghavan, D. Witmer and J. Wright. Beating the random assignment on constraint satisfaction problems of bounded degree. Approximation, Randomization, and Combinatorial Optimization, Algorithms and Techniques (APPROX/RANDOM 2015), *Leibniz International Proceedings in Informatics*, `40`, 2015.

[26] P. Barkoutsos, G. Nannicini, A. Robert, I. Tavernelli and S. Woerner. Improving Variational Quantum Optimization using CVaR. *Quantum*, `4`, 2020.

[27] A.G. Baydin, B.A. Pearlmutter, A.A. Radul and J.M. Siskind. Automatic differentiation in machine learning: a survey. *Journal of Machine Learning Research*, `18`, 2018.

[28] M. Benedetti, D. Garcia-Pintos, O. Perdomo, V. Leyton-Ortega, Y. Nam and A. Perdomo-Ortiz. A generative modeling approach for benchmarking and training shallow quantum circuits. *Quantum Information*, `5`(45), 2019.

[29] M. Benedetti, E. Lloyd, S. Sack and M. Fiorentini. Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, 4(4), 2019.

[30] M. Benedetti, B. Coyle, M. Fiorentini, M. Lubasch and M. Rosenkranz. Variational Inference with a Quantum Computer. *Physical Review Applied*, 16(044057), 2021.

[31] J. Berkson. Application of the Logistic Function to Bio-Assay. *Journal of the American Statistical Association*, 39(227), 1944.

[32] C. Bernhardt. Quantum Computing for Everyone. MIT Press, 2019.

[33] M.V. Berry. Quantal phase factors accompanying adiabatic changes. *Proceedings of the Royal Society of London A*, 392(45), 1964.

[34] D. Berry, G. Ahokas, R. Cleve and B.C. Barry. Efficient quantum algorithms for simulating sparse Hamiltonians. *Communications in Mathematical Physics*, 270(2), 2007.

[35] J.D. Biamonte and P.J. Love. Realizable Hamiltonians for universal adiabatic quantum computers. *Physical Review A*, 78(1), 2008.

[36] A. Billionnet and B. Jaumard. A decomposition method for minimizing quadratic pseudo-Boolean functions. *Operations Research Letters*, 8(3), 1989.

[37] C. Bishop. Pattern Recognition and Machine Learning. Springer, 2006.

[38] S. Boixo, V.N. Smelyanskiy, A. Shabani, S. V. Isakov, M. Dykman, V.S. Denchev, M.H. Amin, A.Y. Smirnov, M. Mohseni

and H. Neven. Computational multiqubit tunnelling in programmable quantum annealers. *Nature Communications*, `7`(10327), 2016.

[39] K. Boothby, A.D. King and J. Raymond. Zephyr Topology of D-Wave Quantum Processors. D-Wave Technical Report, 2021.

[40] V. Bornemann. Homogenization in Time of Singular Perturbed Mechanical Systems. *Lecture Notes in Mathematics*, `1687`, Springer, 1998.

[41] M. Born and V. Fock. Beweis des Adiabatensatzes. *Zeitschrift für Physik*, `51`(3), 1928.

[42] F.G.S.L. Brandão and K.M. Svore. Quantum speed-ups for semidefinite programming. 58th Annual Symposium on Foundations of Computer Science, IEEE, 2017.

[43] G. Brassard, P. Hoyer, M. Mosca and A. Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, `305`, 2002.

[44] S. Bravyi, A. Kliesch, R. Koenig and E. Tang. Obstacles to Variational Quantum Optimization from Symmetry Protection. *Physical Review Letters*, `125`(260505), 2020.

[45] S. Bravyi, D.P. DiVincenzo, R.I. Oliveira and B.M. Terhal. The complexity of stoquastic local Hamiltonian problems. *Quantum Information & Computation*, `8`(5), 2008.

[46] C.D. Bruzewicz, J. Chiaverini, R. McConnell and J.M. Sage. Trapped-ion quantum computing: progress and challenges. *Applied Physics Reviews*, `6`(2), 2019.

[47] J.-L. Brylinski and R. Brylinski. Universal quantum gates. Mathematics of Quantum Computation. Chapman and Hall/CRC, 2002.

[48] H. Bühler, B. Horvath, T. Lyons, I. Perez Arribaz and B. Wood. A data-driven market simulator of financial time series for small data environments. arXiv:2006.14498, 2020.

[49] H. Bühler, B. Horvath, T. Lyons, I. Perez Arribaz and B. Wood. Generating financial markets with signatures. *Risk*, 34(6), 2021.

[50] H. Buhrman, R. Cleve, J. Watrous and R. de Wolf. Quantum fingerprinting. *Physical Review Letters*, 87(167902), 2001.

[51] E. Campbell, A. Khurana and A. Montanaro. Applying quantum algorithms to constraint satisfaction problems. *Quantum*, 3, 2019.

[52] Y. Cao, X. Liu, J. Zhai and S. Hua. A two-stage Bayesian network model for corporate bankruptcy prediction. *International Journal of Finance & Economics*, 27(1), 2022.

[53] M.A. Carreira-Perpiñán and G.E. Hinton. On contrastive divergence learning. *AISTATS*, 2005.

[54] M. Cerezo, A. Sone, T. Volkoff, L. Cincio and P.J. Coles. Cost function dependent barren plateaus in shallow parametrized quantum circuits. *Nature Communications*, 12, 2021.

[55] S. Chakrabarti, H. Yiming, T. Li, S. Feizi and X. Wu, Xiaodi. Quantum Wasserstein generative adversarial networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[56] R. Chandra and Y. He. Bayesian neural networks for stock price forecasting before and during COVID-19 pandemic. *PLoS ONE*, `16`(7), 2021.

[57] D.T. Chang. Bayesian Neural Networks: Essentials. `arXiv:2106.13594`, 2021.

[58] S.Y.-C. Chen, T.-C. Wei, C. Zhang, H. Yu and S. Yoo. Quantum Convolutional Neural Networks for High Energy Physics Data Analysis. *Physical Review Research*, `4`(1), 2022.

[59] S.Y.-C. Chen, S. Yoo and Y.-L.L. Fang. Quantum Long Short-Term Memory. IEEE International Conference on Acoustics, Speech and Signal Processing, 2022.

[60] S. Cheng, J. Chen and L. Wang. Information perspective to probabilistic modeling: Boltzmann machines versus Born machines. *Entropy*, `20`(583), 2018.

[61] A.M. Childs, E. Farhi and J. Preskill. Robustness of adiabatic quantum computation. *Physical Review A*, `65`, 2001.

[62] K. Cho, A. Ilin and T. Raiko. Improved learning of Gaussian-Bernoulli restricted Boltzmann machines. Proceedings of the 20th International Conference on Artificial Neural Networks, 2011.

[63] V. Choi. Minor-embedding in adiabatic quantum computation: I. The parameter setting problem. *Quantum Information Processing*, `7`(5), 2008.

[64] V. Choi. Minor-embedding in adiabatic quantum computation: II. Minor-universal graph design. *Quantum Information Processing*, `10`(3), 2011.

[65] C. Ciliberto, M. Herbster, A.D. Ialongo, M. Pontil, A. Rocchetto, S. Severini and L. Wossnig. Quantum machine learning: a classical perspective. *Proceedings of the Royal Society A*, `474`(2209), 2018.

[66] L. Cincio, Y. Subaşi, A.T. Sornborger and P.J. Coles. Learning the quantum algorithm for state overlap. *New Journal of Physics*, `20`(11), 2018.

[67] J.I. Cirac, R. Blatt, A.S. Parkins and P. Zoller. Preparation of Fock states by observation of quantum jumps in an ion trap. *Physical Review Letters*, `70`(6-8), 1993.

[68] J.I. Cirac and P. Zoller. Quantum Computations with Cold Trapped Ions. *Physical Review Letters*, `74`(20), 1995.

[69] I. Cong, S. Choi and M.D. Lukin. Quantum convolutional neural networks. *Nature Physics*, `15`(12), 2019.

[70] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, `20`(3), 1995.

[71] L. Coslovich, R. Pesenti and W. Ukovich. Large-scale set partitioning problems: Some real-world instances hide a beneficial structure. *Technological and Economic Development of Economy*, `12`(1), 2006.

[72] B. Coyle, M. Henderson, J. Chan Jin Le, N. Kumar, M. Paini and E. Kashefi. Quantum versus classical generative modelling in Finance. *Quantum Science and Technology*, `6`(2), 2021.

[73] B. Coyle, D. Mills, V. Danos and E. Kashefi. The Born supremacy: Quantum advantage and training of an Ising Born

machine. *Quantum Information*, 6(1), 2020.

[74] A. Crespi, R. Ramponi, R. Osellame, L. Sansoni, I. Bongioanni, F. Sciarrino, G. Vallone and P. Mataloni. Integrated photonic quantum gates for polarization qubits. *Nature Communications*, 1570, 2011.

[75] G. Cybenko. Approximation by Superpositions of a Sigmoidal Function. *Mathematics of Control, Signals and Systems*, 2, 1989.

[76] E.D. Dahl. Programming with D-Wave: map coloring problem. D-Wave White Paper, 2013.

[77] . P.-L. Dallaire-Demers and N. Killoran. Quantum generative adversarial networks. *Physical Review A*, 98(1), 2018.

[78] G.B. Dantzig and J.H. Ramser. The truck dispatching problem. *Management Science*, 6(1), 1959.

[79] S. Darolles and C. Gouriéroux. Conditionally Fitted Sharpe Performance with an Application to Hedge Fund Rating. *Journal of Banking & Finance*, 34(3), 2010.

[80] A. Dawar. Quantum Computing. https://www.cl.cam.ac.uk/teaching/1819/QuantComp/notes18.pdf

[81] B.S. Dees, L. Stanković, A.G. Constantinides and D.P. Mandic. Portfolio Cuts: A Graph-Theoretic Framework to Diversification. International Conference on Acoustics, Speech and Signal Processing, 2020.

[82] V. DeMiguel, L. Garlappi and R. Uppal. Optimal Versus Naive Diversification: How Inefficient is the 1/*N* Portfolio Strategy? *The Review of Financial Studies*, 22(5), 2009.

[83] V.S. Denchev, S. Boixo, S.V. Isakov, N. Ding, R. Babbush, V. Smelyanskiy, J. Martinis and H. Neven. What is the computational value of finite-range tunneling? *Physical Review X*, 6(3), 2016.

[84] D.-L. Deng, X. Li and S.D. Sarma. Machine Learning Topological States. *Physical Review B*, 96(195145), 2017.

[85] D.-L. Deng, X. Li and S.D. Sarma. Quantum Entanglement in Neural Network States. *Physical Review X*, 7(021021), 2017.

[86] P.A.M. Dirac. The Principles of Quantum Mechanics. Oxford University Press, 1930.

[87] D.P. DiVincenzo. The physical implementation of quantum computation. *Fortschritte der Physik*, 48(9–11), 2000.

[88] Y. Du, M.-H. Hsieh, T. Liu and D. Tao. Expressive power of parameterized quantum circuits. *Physical Review Research*, 2(033125), 2020.

[89] D.J. Duffy. Finite Difference methods in financial engineering: a Partial Differential Equation approach. John Wiley & Sons, 2013.

[90] D-Wave Systems. Practical quantum computing. D-Wave Technology Overview, 2020.

[91] D-Wave Systems. D-Wave QPU Architecture: Topologies. docs.dwavesys.com/docs/latest/c_gs_4.html, 2021.

[92] D-Wave Systems. Advantage Performance Update. www.dwavesys.com/solutions-and-products/systems, 2021.

[93] D-Wave Systems. Ahead of the Game: D-Wave Delivers Prototype of Next-Generation Advantage2 Annealing Quantum Computer. www.dwavesys.com/company/newsroom/press-release, 2022.

[94] G.K. Dziugaite and D.M. Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. arXiv:1703.11008, 2017.

[95] F. Eckerli and J. Osterrieder. Generative Adversarial Networks in finance: an overview. arXiv:2106.06364, 2021.

[96] E. Farhi, J. Goldstone and S. Gutmann. A Quantum approximate optimisation algorithm. arXiv:1411.4028, 2014.

[97] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren and D. Preda. A quantum adiabatic evolution algorithm applied to random instances of an NP-Complete problem. *Science*, 292(5516), 2001.

[98] E. Farhi, J. Goldstone, S. Gutmann and M. Sipser. Quantum computation by adiabatic evolution. arXiv:0001106, 2000.

[99] E. Farhi and A.W. Harrow. Quantum Supremacy through the Quantum Approximate Optimization Algorithm. arXiv:1602.07674, 2016.

[100] E. Farhi and H. Neven. Classification with quantum neural networks on near-term processors. arXiv:1802.06002, 2018.

[101] R. Feynman, R. Leighton and M. Sands. The Schrödinger equation in a classical context: a seminar on superconductivity. The Feynman Lectures on Physics, The Definitive Edition, 2006.

[102] A. Fischer and C. Igel. An introduction to restricted Boltzmann machines. Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications. Lecture Notes in Computer Science, 7441, Springer, 2012.

[103] A. Fischer and C. Igel. Training restricted Boltzmann machines: an introduction. *Pattern Recognition*, 47(1), 2014.

[104] F. Fontanela, A. Jacquier and M. Oumgari. A quantum algorithm for linear PDEs arising in finance. *SIAM Journal on Financial Mathematics*, 12(4), 2021.

[105] V. Fortuin. Priors in Bayesian deep learning: a review. International Statistical Review, 2022.

[106] M. Frank. Foundations of generalized reversible computing. International Conference on Reversible Computation, 2017.

[107] Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1), 1997.

[108] P. García-Molina, J. Rodríguez-Mediavilla and J.J. García-Ripoll. Solving partial differential equations in quantum computers. arXiv:2104.02668, 2021.

[109] A. Gilyén, S. Lloyd and E. Tang. Quantum-inspired low-rank stochastic regression with logarithmic dependence on the

dimension. arXiv:1811.04909, 2018.

[110] V. Giovannetti, S. Lloyd and L. Maccone. Quantum random access memory. *Physical Review Letters*, 100(16), 2008.

[111] P. Glasserman. Monte Carlo Methods in Financial Engineering. Stochastic Modelling and Applied Probability, 53, Springer, 2003.

[112] M.X. Goemans. Semidefinite programming in combinatorial optimization. *Mathematical Programming*, 79, 1997.

[113] I. Goodfellow, Y. Bengio and A. Courville. Deep Learning. MIT Press, 2016.

[114] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio. Generative Adversarial Nets. NIPS Proceedings, 2014.

[115] H. Gouk. Regularisation of neural networks by enforcing Lipschitz continuity. *Machine Learning*, 110(2), 2021.

[116] E. Grant, T.S. Humble and B. Stump. Benchmarking Quantum Annealing Controls with Portfolio Optimization. *Physical Review Applied*, 15(1), 2021.

[117] L.K. Grover. A fast quantum mechanical algorithm for database search. Proceedings of the twenty-eighth annual ACM symposium on Theory of computing. ACM, 1996.

[118] L.K. Grover and T. Rudolph. Creating superpositions that correspond to efficiently integrable probability distributions. arXiv:0208112, 2002.

[119] M. Grundmann. Quantum devices of reduced dimensionality. Encyclopedia of Condensed Matter Physics, Elsevier, 2005.

[120] G.G. Guerreschi. Solving quadratic unconstrained binary optimization with divide-and-conquer and quantum algorithms. arXiv:2101.07813, 2021.

[121] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin and A.C. Courville. Improved training of Wasserstein GANs. *Advances in neural information processing systems*, 30, 2017.

[122] S. Hadfield, Z. Wang, B. O'Gorman, E.G. Rieffel, D. Venturelli and R. Biswas. From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz. *Algorithms*, 12(2), 2019.

[123] B.C. Hall. Quantum Theory for Mathematicians. *Graduate Texts in Mathematics*, 267, Springer, 2013.

[124] R. Hamerly, T. Inagaki, P.L. McMahon, D. Venturelli, A. Marandi, E. Ng, C. Langrock, K. Inaba, T. Honjo, K. Enbutsu, T.Umeki, R. Kasahara, S. Utsunomiya, S. Kako, K. Kawarabayashi, R.L. Byer, M.M. Fejer, H. Mabuchi, D. Englund, E. Rieffel, H. Takesue and Y. Yamamoto. Experimental investigation of performance differences between coherent Ising machines and a quantum annealer. *Science Advances*, 5(5), 2019.

[125] J.M. Hammersley and P. Clifford. Markov fields on finite graphs and lattices. Unpublished, 1971.

[126] A.W. Harrow, A. Hassidim and S. Lloyd. Quantum algorithm for linear systems of equations. *Physical Review Letters*,

`103`(150502), 2009.

[127] R. Hassan, B. Cohanim, O. de Weck and G. Venter. A comparison of particle swarm optimization and the genetic algorithm. 46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, 2005.

[128] M.B. Hastings. Classical and Quantum Bounded Depth Approximation Algorithms. *Quantum Information & Computation*, `19`(13-14), 2019.

[129] V. Havlíček, A.D. Córcoles, K. Temme, A.W. Harrow, A. Kandala, J.M. Chow and J.M. Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, `567`, 2019.

[130] J. He and L. Kang. On the convergence rates of genetic algorithms. *Theoretical Computer Science*, `229`, 1999.

[131] L.-P. Henry, S. Thabet, C. Dalyac and L. Henriet. Quantum evolution kernel: Machine learning on graphs with programmable arrays of qubits. *Physical Review A*, `104`(3), 2021.

[132] A.D. Hill, M.J. Hodson, N. Didier and M.J. Reagor. Realization of arbitrary doubly-controlled quantum phase gates. `arXiv:2108.01652`, 2021.

[133] G. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, `14`(8), 2002.

[134] G. Hinton. A practical guide to training restricted Boltzmann machines. Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, `7700`, Springer, 2012.

[135] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, `313`, 2006.

[136] T.K. Ho. Random Decision Forests. Proceedings of the 3rd International Conference on Document Analysis and Recognition, 1995.

[137] C.A.R. Hoare. Algorithm 64: Quicksort. *Communications of the ACM*, `4`(7), 1961.

[138] M. Hodson, B. Ruck, H. Ong, D. Garvin and S. Dulman. Portfolio rebalancing experiments using the Quantum Alternating Operator Ansatz. `arXiv:1911.05296`, 2019.

[139] Z. Holmes, K. Sharma, M. Cerezo and P.J. Coles. Connecting ansatz expressibility to gradient magnitudes and barren plateaus. *PRX Quantum*, `3`(1), 2022.

[140] F.-Y. Hong, Y. Xiang, Z.-Y. Zhu, L.-Z. Jiang and L.-N. Wu. Robust quantum random access memory. *Physical Review A*, `86`(1), 2012.

[141] K. Hornik, M. Stinchcombe and H. White. Universal Approximation of an Unknown Mapping and Its Derivatives Using Multilayer Feedforward Networks. *Neural Networks*, `3`(5), 1990.

[142] T. Hur, L. Kim and D.K. Park. Quantum convolutional neural network for classical data classification. *Quantum Machine Intelligence*, `4`(1), 2022.

[143] H.Y. Huang, M. Broughton, M. Mohseni, R. Babbush, S. Boixo, H. Neven and J.R. McClean. Power of data in quantum

machine learning. *Nature Communications*, 12(2631), 2021.

[144] W. Huggins, P. Patel, K.B. Whaley and E.M. Stoudenmire. Towards quantum machine learning with tensor networks. *Quantum Science and Technology*, 4(2), 2019.

[145] IBM Quantum Roadmap: Expanding the IBM Quantum roadmap to anticipate the future of quantum-centric supercomputing. https://research.ibm.com/blog/ibm-quantum-roadmap-2025, 2022.

[146] IonQ: Duke University and IonQ Develop New Quantum Computing Gate, Only Possible on IonQ and Duke Systems. https://investors.ionq.com/news/news-details/2022/Duke-University-and-IonQ-Develop-New-Quantum-Computing-Gate-Only-Possible-on-IonQ-and-Duke-Systems/default.aspx, 2022.

[147] E. Ising. Beitrag zur Theorie des Ferromagnetismus. *Zeitschrift für Physik*, 31(1), 1925.

[148] B. Jackson, J.D. Scargle, D. Barnes, S. Arabhi, A. Alt, P. Gioumousis, E. Gwin, P. Sangtrakulcharoen, L. Tan and T.T. Tsai. An Algorithm for Optimal Partitioning of Data on an Interval. *IEEE Signal Processing Letters*, 12(2), 2005.

[149] S. Jansen, R. Seiler and M.-B. Ruskai. Bounds for the adiabatic approximation with applications to quantum computation. *Journal of Mathematical Physics*, 48(102111), 2007.

[150] M.R. Jerrum, L.G. Valiant and V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43, 1986.

[151] M. Johnson, M.H.S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A.J. Berkley, J. Johansson, P. Bunyk, E.M. Chapple, C. Enderud, J.P. Hilton, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Oh, I. Perminov, C. Rich, M.C. Thom, E. Tolkacheva, C.J.S. Truncik, S. Uchaikin, J. Wang, B. Wilson and G. Rose. Quantum annealing with manufactured spins. *Nature*, `473`(7346), 2011.

[152] T. Kadowaki and H. Nishimori. Quantum annealing in the transverse Ising model. *Physical Review E*, `58`(5), 1998.

[153] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J.M. Chow and J.M. Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, `549`, 2017.

[154] H. Karimi and G. Rosenberg. Boosting quantum annealer performance via sample persistence. *Quantum Information Processing*, `16`(7), 2017.

[155] R. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*. Plenum Press, 1972.

[156] T. Kato. On the adiabatic theorem of quantum mechanics. *Journal of the Physical Society of Japan*, `5`(6), 1950.

[157] P. Kaye, R. Laflamme, Raymond and M. Mosca. An introduction to quantum computing. OUP, 2007.

[158] H. Kellerer, R. Mansini and M.G. Speranza. Selecting portfolios with fixed costs and minimum transaction lots. *Annals of Operations Research*, `99`(1-4), 2000.

[159] H. Kellerer and U. Pferschy. Knapsack problems. Springer, 2004.

[160] I. Kerenidis, J. Landman and A. Prakash. Quantum algorithms for deep convolutional neural networks. arXiv:1911.01117, 2019.

[161] D. Khachatryan. Variational Quantum Eigensolver. https://github.com/DavitKhach/quantum-algorithms-tutorials, 2020.

[162] A. Khoshaman, W. Vinci, B. Denis, E. Andriyash, H. Sadeghi and M.H. Amin. Quantum variational autoencoder. *Quantum Science and Technology*, 4(1), 2019.

[163] M. Kim, K. Kim, J. Hwang, E.-G. Moon and J. Ahn. Rydberg quantum wires for maximum independent set problems. *Nature Physics*, 18, 2022.

[164] M. Kjaergaard, M.E. Schwartz, J. Braumüller, P. Krantz, J. I-Jan Wang, S. Gustavsson and W.D. Oliver. Superconducting qubits: current state of play. *Annual Review of Condensed Matter Physics*, 11, 2020.

[165] E. Knill, R. Laflamme and G.J. Milburn. A scheme for efficient quantum computation with linear optics. *Nature*, 409, 2001.

[166] D. Knuth. The Art of Computer Programming, Volume 3 / Sorting and Searching. Addison-Wesley, 2nd Edition, 1998.

[167] D. Koller and N. Friedman. Probabilistic graphical models. MIT Press, 2009.

[168] I. Kolotouros and P. Wallden. An evolving objective function for improved variational quantum optimisation. *Physical Review Research*, `4`(2), 2022.

[169] A. Kondratyev. Curve dynamics with artificial neural networks. *Risk*, `31`(6), 2018.

[170] A. Kondratyev. Non-differentiable learning of quantum circuit Born machine with genetic algorithm. *Wilmott*, `2021`(114), 2021.

[171] A. Kondratyev. Quantum Machine Learning. Presentation at the Quant Insights Conference. SSRN:3865344, 2021.

[172] A. Kondratyev and G. Giorgidze. Evolutionary algos for optimising MVA. *Risk*, `30`(12), 2017.

[173] A. Kondratyev and C. Schwarz. The market generator. *Risk*, `33`(2), 2020.

[174] A. Kondratyev, C. Schwarz and B. Horvath. The data anonymiser. *Risk*, `33`(8), 2020.

[175] A. Kondratyev and D. Venturelli. Beyond Markowitz with quantum annealing. *Risk*, `32`(6), 2019.

[176] A. Koshiyama, N. Firoozye and P. Treleaven. Generative adversarial networks for financial trading strategies fine-tuning and combination. *Quantitative Finance*, `21`(5), 2021.

[177] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson and W.D. Oliver. A quantum engineer's guide to superconducting qubits. *Applied Physics Reviews*, `6`(2), 2019.

[178] A. Krizhevsky. Learning multiple layers of features from tiny images. University of Toronto, Technical Report, 2009.

[179] S. Kshatriya and P.K. Prasanna. Genetic algorithm-based portfolio optimization with higher moments in global stock markets. *Journal of Risk*, `20`(4), 2018.

[180] D. Kudrow, K. Bier, Z. Deng, D. Franklin, Y. Tomita, K.R. Brown and F.T. Chong. Quantum rotations: a case study in static and dynamic machine-code generation for quantum computers. International Symposium on Computer Architecture, 2013.

[181] L. Lamata, U. Alvarez-Rodriguez, J.D. Martín-Guerrero, M. Sanz and E. Solano. Quantum autoencoders via quantum adders with genetic algorithms. *Quantum Science and Technology*, `4`(1), 2018.

[182] L.D. Landau and E.M. Lifshitz. Quantum Mechanics. Non-Relativistic Theory. Pergamon Press, 1965.

[183] R. Landauer. Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development*, `5`(3), 1961.

[184] J. Landman. Quantum Algorithms for Unsupervised Machine Learning and Neural Networks. arXiv:2111.03598, 2021.

[185] C. Li. A Gentle Introduction to Gradient Boosting. Northeastern University, College of Computer and Information Science – Tutorial. http://www.chengli.io/tutorials/gradient_boosting.pdf

[186] R.Y. Li, R. Di Felice, R. Rohs and D.A. Lidar. Quantum annealing versus classical machine learning applied to a simplified computational biology problem. *Quantum Information*, `4`(14), 2018.

[187] Q. Li, H. Wu, W. Qian, X. Li, Q. Zhu and S. Yang. Portfolio Optimization Based on Quantum HHL Algorithm. International Conference on Artificial Intelligence and Security, 2022.

[188] N. Linden, A. Montanaro and C. Shao. Quantum vs. classical algorithms for solving the heat equation. *Communications in Mathematical Physics*, 2022.

[189] J.-G. Liu and L. Wang. Differentiable learning of quantum Circuit Born Machine. *Physical Review A*, `98`(062324), 2018.

[190] L. Liu, S. Yang and D. Wang. Particle swarm optimization with composite particles in dynamic environments. *IEEE Transations on Systems, Man, and Cybernetics, Part B: Cybernetics*, `40`(6), 2010.

[191] S. Lloyd. Universal quantum simulators. *Science*, `273`(5278), 1996.

[192] S. Lloyd and C. Weedbrook. Quantum generative adversarial learning. *Physical Review Letters*, `121`(4), 2018.

[193] M. López de Prado. Advances in Financial Machine Learning. Wiley, 2018.

[194] M. López de Prado. A Robust Estimator of the Efficient Frontier. SSRN:3469961, 2019.

[195] R. Loredo. Learn Quantum Computing with Python and IBM Quantum Experience. Packt, 2020.

[196] J. Lu, and S. Yi. Autoencoding Conditional GAN for Portfolio Allocation Diversification. arXiv:2207.05701, 2022.

[197] A. Lucas. Ising formulations of many NP problems. *Frontiers in Physics*, 2(5), 2014.

[198] X.-Z. Luo, J.-G. Liu, P. Zhang and L. Wang. Yao.jl: Extensible, efficient framework for quantum algorithm design. *Quantum*, 4, 2020.

[199] M. Magris, M. Shabani and A. Iosifidis. Bayesian Bilinear Neural Network for Predicting the Mid-price Dynamics in Limit-Order Book Markets. arXiv:2203.03613, 2022.

[200] F. Mallet, F.R. Ong, A. Palacios-Laloy, F. Nguyen, P. Bertet, D. Vion and D. Esteve. Single-shot qubit readout in circuit quantum electrodynamics. *Nature Physics*, 5, 2009.

[201] S. Mandrà and H.G. Katzgraber. A deceptive step towards quantum speedup detection. *Quantum Science and Technology*, 3(4), 2018.

[202] H. Markowitz. Portfolio selection. *Journal of Finance*, 7(1), 1952.

[203] J. Marshall, D. Venturelli, I.Hen and E.G. Rieffel. The power of pausing: advancing understanding of thermalization in experimental quantum annealers. *Physical Review Applied*, 11(044083), 2019.

[204] J.M. Martinis. Superconducting phase qubits. *Quantum Information Processing*, 8, 2009.

[205] M. Marzec. Portfolio optimization: applications in quantum computing. Handbook of high-frequency trading and modeling in Finance. John Wiley & Sons, 2016.

[206] K.-P. Marzlin and B.C. Sanders. Inconsistency in the application of the adiabatic theorem. *Physical Review Letters*, 93(160408), 2004.

[207] J.R. McClean, S. Boixo, V.N. Smelyanskiy, R. Babbush and H. Neven. Barren plateaus in quantum neural network training landscapes. *Nature Communications*, 9(4812), 2018.

[208] D. McClure and J. Gambetta. Quantum computation center opens. https://www.ibm.com/blogs/research/2019/09/quantum-computation-center, 2019.

[209] C. McGeoch. Adiabatic quantum computation and quantum annealing: theory and practice. Synthesis Lectures on Quantum Computing, Morgan & Claypool, 2014.

[210] C. McGeoch, P. Farre and K. Boothby. The D-Wave Advantage2 Prototype. D-Wave Technical Report, 2022.

[211] C. McGeoch and C. Wang. Experimental evaluation of an adiabiatic quantum system for combinatorial optimization. Proceedings of the ACM International Conference on Computing Frontiers, CF, 2013.

[212] P.A. Mello. The von Neumann Model of Measurement in Quantum Mechanics. Conference Proceedings, 1575(1), American Institute of Physics, 2014.

[213] K. Mitarai, M. Negoro, M. Kitagawa and K. Fujii. Quantum circuit learning. *Physical Review A*, 98(032309), 2018.

[214] M. Mitchell. An introduction to genetic algorithms. MIT Press, 1998.

[215] N. Moll, P. Barkoutsos, L.S. Bishop, J.M. Chow, A. Cross, D.J. Egger, S. Filipp, A. Fuhrer, J.M. Gambetta, M. Ganzhorn, A. Kandala, A. Mezzacapo, P. Müller, W. Riess, G. Salis, J. Smolin, I. Tavernelli and K. Temme. Quantum optimization using variational algorithms on near-term quantum devices. *Quantum Science and Technology*, 3(030503), 2018.

[216] A. Montanaro. Quantum speedup of Monte Carlo methods. *Proceedings of the Royal Society A*, 471(2181), 2015.

[217] M. Morini. One more model risk when using Gaussian copula for risk management. SSRN:1520670, 2009.

[218] A. Mott, J. Job, J.R. Vlimant, D. Lidar and M. Spiropulu. Solving a Higgs optimization problem with quantum annealing for machine learning. *Nature*, 550, 2017.

[219] W.C. Myrvold. On the Relation of the Laws of Thermodynamics to Statistical Mechanics. Preprint. http://philsci-archive.pitt.edu/id/eprint/19361, 2021.

[220] H. Naomichi and M. Suzuki. Finding exponential product formulas of higher orders. *Quantum annealing and other*

optimization methods. Springer, 2005.

[221] J. Nemirovsky and Y. Sagi. Fast universal two-qubit gate for neutral fermionic atoms in optical tweezers. *Physical Review Research*, `3`(013113), 2021.

[222] N.H. Nguyen, E.C. Behrman and J.E. Steck. Quantum learning with noise and decoherence: a robust quantum neural network. *Quantum Machine Intelligence*, `2`(1), 2020.

[223] M.A. Nielsen and I.S. Chuang. Quantum Computation and Quantum Information. Cambridge University Press, 10th Anniversary Edition, 2010.

[224] S.M. Nikolskii. Approximation of functions of several variables and imbedding theorems. Springer, 1975.

[225] J.L. O'Brien. Optical quantum computing. *Science*, `318`(5856), 2007.

[226] B. Øksendal. Stochastic differential equations. Springer, 5th Edition, 2000.

[227] G. Ortiz, J.E. Gubernatis, E. Knill and R. Laflamme. Quantum algorithms for fermionic simulations. *Physical Review A*, `64`(022319), 2001.

[228] R. Orús, S. Mugel and E. Lizaso. Quantum computing for finance: overview and prospects. *Reviews in Physics*, `4`, 2019.

[229] M. Paini. Quantum Finance: The Road to Business Applications. Presentation at the London Quantum Computing Meetup. `Quantum Finance`, 2018.

[230] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and É. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, `12`, 2011.

[231] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik and J.L. O'Brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, `5213`, 2014.

[232] E. Peters, J. Caldeira, A. Ho, S. Leichenauer, M. Mohseni, H. Neven, P. Spentzouris, D. Strain and G.N. Perdue. Machine learning of high dimensional data on a noisy quantum processor. *Quantum Information*, `7`(1), 2021.

[233] R.C. Pfaffenberger and J.H. Patterson. Statistical methods for business and economics. Irwin, 3rd Edition, 1987.

[234] F. Phillipson and H.S. Bhatia. Portfolio Optimisation Using the D-Wave Quantum Annealer. In: M. Paszynski, D. Kranzlmüller, V.V. Krzhizhanovskaya, J.J. Dongarra, and P.M.A Sloot (eds). Computational Science – ICCS 2021. Lecture Notes in Computer Science, `12747`, Springer, 2021.

[235] B. Pokharel, Z.G. Izquierdo, P.A. Lott, E. Strbac, K. Osiewalski, E. Papathanasiou, A. Kondratyev, D. Venturelli and E. Rieffel. Inter-generational comparison of D-Wave quantum annealers in solving hard scheduling problems. `arXiv:2112.00727`, 2021.

[236] R. Poli. Analysis of the publications on the applications of particle swarm optimisation. *Journal of Artificial Evolution and Applications*, 2008.

[237] J. Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, `2`, 2018.

[238] Qiskit: An open-source framework for quantum computing. https://qiskit.org, 2019.

[239] Qiskit: Circuit Library – RZ Gate. https://qiskit.org/documentation/stubs/qiskit.circuit.library.RZGate.html, 2022.

[240] Qiskit: Circuit Library – RZX Gate. https://qiskit.org/documentation/stubs/qiskit.circuit.library.RZXGate.html, 2022.

[241] J.R. Quinlan. Australian Credit Approval Dataset. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml. UC Irvine, School of Information and Computer Science, 1987.

[242] J.R. Quinlan. Simplifying Decision Trees. *International Journal of Man-Machine Studies*, `27`(3), 1987.

[243] S. Raschka and V. Mirjalili. Python machine learning. Packt, 3rd Edition, 2019.

[244] S.E. Rasmussen and N.T. Zinner. Simple implementation of high fidelity controlled-iSWAP gates and quantum circuit exponentiation of non-Hermitian gates. *Physical Review Research*, `2`(033097), 2020.

[245] P. Rebentrost and S. Lloyd. Quantum computational finance: quantum algorithm for portfolio optimization. arXiv:1811.03975, 2018.

[246] Rigetti Quantum Roadmap: Rigetti Computing Reports First Quarter 2022 Financial Results and Provides Business Update. https://investors.rigetti.com/node/7371/pdf, 2022.

[247] J.J. Rissanen. Fisher information and stochastic complexity. *IEEE Transactions on Information Theory*, 42(1), 1996.

[248] A. Robert, P.K. Barkoutsos, S. Woerner and I. Tavernelli. Resource-efficient quantum algorithm for protein folding. *Quantum Information*, 7(38), 2021.

[249] R. Robinett. Quantum mechanics: classical results, modern systems, and visualized examples. Oxford University Press, 2nd Edition, 2006.

[250] T.F. Rønnow, Z. Wang, J. Job, S. Boixo, S.V. Isakov, D. Wecker, J.M. Martinis, D.A. Lidar and M. Troyer. Defining and detecting quantum speedup. *Science*, 345(6195), 2014.

[251] G. Rosenberg, P. Haghnegahdar, P. Goddard, P. Carr, K. Wu and M. López De Prado. Solving the optimal trading trajectory problem using a quantum annealer. *IEEE Journal of Selected Topics in Signal Processing*, 10(6), 2016.

[252] R. Salakhutdinov, A. Mnih and G. Hinton. Restricted Boltzmann machines for collaborative filtering. Proceedings of the 24th International Conference on Machine Learning, 2007.

[253] G.E. Santoro, R. Martonak, E. Tosatti and R. Car. Theory of quantum annealing of an Ising spin glass. *Science*, `295`(2427), 2002.

[254] V. Schmitt. Design, fabrication and test of a four superconducting quantum-bit processor. PhD Thesis, Université Pierre et Marie Curie, Physics, 2015.

[255] A. Schrijver. Theory of integer and linear programming. Wiley, 1998.

[256] M. Schuld, A. Bocharov, K. Svore and N. Wiebe. Circuit-centric quantum classifiers. *Physical Review A*, `101`(3), 2020.

[257] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac and N. Killoran. Evaluating analytic gradients on quantum hardware. *Physical Review A*, `99`(3), 2019.

[258] F. Schwabl. Quantum mechanics. Springer, 4th Edition, 2007.

[259] A. Selby. D-Wave: comment on comparison with classical computers, 2013.

[260] S. Sethi. Optimal control theory. Applications to management science and economics. Springer, 2019.

[261] A. Sethia, R. Patel and P. Raut. Data augmentation using generative models for credit card fraud detection. 4th International Conference on Computing Communication and Automation, 2018.

[262] A.D. Shapiro. King-Rook vs. King-Pawn Chess Dataset. UCI Machine Learning Repository.

http://archive.ics.uci.edu/ml. UC Irvine, School of Information and Computer Science, 1987.

[263] A.D. Shapiro. Structured Induction in Expert Systems. Addison-Wesley, 1987.

[264] R.R. Sharapov and A.V. Lapshin. Convergence of genetic algorithms. *Pattern Recognition and Image Analysis*, 16(3), 2006.

[265] P.W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5), 1997.

[266] S. Sim, P.D. Johnson and A. Aspuru-Guzik. Expressibility and Entangling Capability of Parameterized Quantum Circuits for Hybrid Quantum-Classical Algorithms. *Advanced Quantum Technologies*, 2(12), 2019.

[267] B. Simon. Holonomy, the Quantum Adiabatic Theorem, and Berry's Phase. *Physical Review Letters*, 51(2167), 1983.

[268] H. Situ, Z. He, Y. Wang, L. Li and S. Zheng. Quantum generative adversarial network for generating discrete distribution. *Information Sciences*, 538, 2020.

[269] G.D. Smith. Numerical solution of partial differential equations: finite difference methods. Oxford University Press, 1985.

[270] P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. Parallel Distributed Processing Explorations in the Microstructure of Cognition, 1: Foundations. MIT Press, 1986.

[271] K. Sörensen. Clustering in Financial Markets: A Network Theory Approach. Master of Science Thesis, Royal Institute of Technology, Stockholm, 2014.

[272] M. Stechły. Variational Quantum Eigensolver explained. https://www.mustythoughts.com/variational-quantum-eigensolver-explained, 2020.

[273] M. Steffen, W. van Dam, T. Hogg, G. Breyta and I. Chuang. Experimental implementation of an adiabatic quantum optimization algorithm. *Physical Review Letters*, 90(6), 2003.

[274] S.A. Stein, B. Baheri, D. Chen, Y. Mao, Q. Guan, A. Li, B. Fang and S. Xu. QuGAN: A Generative Adversarial Network Through Quantum States. arXiv:2010.09036, 2020.

[275] F.W. Strauch, P.R. Johnson, A.J. Dragt, C.J. Lobb, J.R. Anderson and F.C. Wellstood. Quantum logic gates for coupled superconducting phase qubits. *Physical Review Letters*, 91(167005), 2003.

[276] M. Suchara. Quantum Algorithms and Their Applications. CQE-Protiviti Design Thinking Workshop, 2021.

[277] Y. Sung, L. Ding, J. Braumüller, A. Vepsäläinen, B. Kannan, M. Kjaergaard, A. Greene, G.O. Samach, C. McNally, D. Kim, A. Melville, B.M. Niedzielski, M.E. Schwartz, J.L. Yoder, T.P. Orlando, S. Gustavsson and W.D. Oliver. Realization of high-fidelity CZ and ZZ-free iSWAP gates with a tunable coupler. *Physical Review X*, 11(021058), 2021.

[278] R.S. Sutor. Dancing with Qubits: How quantum computing works and how it can change the world. Packt, 2019.

[279] M. Suzuki. General Decomposition Theory of Ordered Exponentials. *Proceedings of Japan Academy*, 69(B), 1993.

[280] G.W. Taylor, G.E. Hinton and S.T. Roweis. Two distributed-state models for generating high-dimensional time series. *Journal of Machine Learning Research*, 12(28), 2011.

[281] M. Tegmark. Life 3.0. Being Human in the age of artificial intelligence. Allen Lane, Penguin Random House, 2017.

[282] J.L. Ticknor. A Bayesian regularized artificial neural network for stock market forecasting. *Expert Systems with Applications*, 40(14), 2013.

[283] L. Tomawski, I. Mrózb and Z. Kukułac. From Thomson Formula to Resonant Equivalent Diagrams. *Acta Physica Polonica A*, 139(3), 2021.

[284] T. Toffoli. Reversible Computing. International Colloquium on Automata, Languages, and Programming. ICALP 1980: Automata, Languages and Programming, 1980.

[285] T.T. Tran, M. Do, E.G. Rieffel, J. Frank, Z. Wang, B. O'Gorman, D. Venturelli and J.C. Beck. A hybrid quantum-classical approach to solving scheduling problems. The 9th Annual Symposium on Combinatorial Search, 2016.

[286] H.F. Trotter. On the product of semi-groups of operators. *Proceedings of the American Mathematical Society*, 10(4), 1959.

[287] C.A. Trugenberger. Probabilistic quantum memories. *Physical Review Letters*, 87(067901), 2001.

[288] H. Tuy. Minimax Theorems Revisited. *Acta Mathematica Vietnamica*, 29(3), 2004.

[289] F. Vaezi, S.J. Sadjadi and A. Makui. A portfolio selection model based on the knapsack problem under uncertainty. *PLoS ONE*, 14(5), 2019.

[290] J. van Apeldoorn and A. Gilyén. Improvements in quantum SDP-solving with applications. arXiv:1804.05058, 2018.

[291] W. van Dam, M. Mosca and U. Vazirani. How powerful is adiabatic quantum computation? Proceedings of Symposium on the Foundations of Computer Science, 2001.

[292] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1), 1996.

[293] V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2), 1971.

[294] D. Ventura and T. Martinez. Quantum associative memory. *Information Sciences*, 124(1), 2000.

[295] D. Venturelli, D.J.J. Marchand and G. Rojo. Quantum annealing implementation of job-shop scheduling. arXiv:1506.08479, 2015.

[296] D. Venturelli and A. Kondratyev. Reverse quantum annealing approach to portfolio optimization problems. *Quantum Machine Intelligence*, 1(3), 2019.

[297] J. von Neumann. Zur Theorie der Gesellschaftsspiele. *Mathematische Annalen*, 100: 295-320, 1928.

[298] X. Wang, Y. Du, Y. Luo and D. Tao. Towards understanding the power of quantum kernels in the NISQ era. *Quantum*, 5, 2021.

[299] S. Wang, E. Fontana, M. Cerezo, K. Sharma, A. Sone, L. Cincio and P.J. Coles. Noise-induced barren plateaus in variational quantum algorithms. *Nature Communications*, 12(1), 2021.

[300] S. Wei, ShiJie, Y. Chen, Z. Zhou and G. Long. A quantum convolutional neural network on NISQ devices. *AAPPS Bulletin*, 32(1), 2022.

[301] M. Wiese, B. Wood, A. Pachoud, R. Korn, H. Buehler, P. Murray and L. Bai. Multi-asset spot and option market simulation. arXiv:2112.06823, 2021.

[302] M. Wiese, R. Knobloch, R. Korn and P. Kretschmer. Quant GANs: deep generation of financial time series. *Quantitative Finance*, 20(9), 2020.

[303] J.W.J. Williams. Algorithm 232 — Heapsort. *Communications of the ACM*, 7(6), 1964.

[304] J. Wurtz and P.J. Love. Counterdiabaticity and the quantum approximate optimization algorithm. *Quantum*, 6, 2022.

[305] N. Xu, J. Zhu, D. Lu, X. Zhou, X. Peng and J. Du. Quantum factorization of 143 on a dipolar-coupling nuclear magnetic resonance system. *Physical Review Letters*, 108(13), 2012.

[306] R. Yalovetzky, P. Minssen, D. Herman and M. Pistoia. NISQ-HHL: Portfolio optimization for near-term quantum hardware. arXiv:2110.15958, 2021.

[307] I.C. Yeh and C.H. Lien. Default of Credit Card Clients Dataset. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml. UC Irvine, School of Information and Computer Science, 2009.

[308] I.C. Yeh and C.H. Lien. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications*, 36(2), 2009.

[309] K. Yosida. Functional analysis. Springer, 1965.

[310] T. Zhao, C. Sun, A. Cohen, J. Stokes and S. Veerapaneni. Quantum-inspired variational algorithms for partial differential equations: Application to financial derivative pricing. arXiv:2207.10838, 2022.

[311] X. Zhou, S. Li and Y. Feng. Quantum circuit transformation based on simulated annealing and heuristic search. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(12), 2020.

[312] E. Zhu, S. Johri, D. Bacon, M. Esencan, J. Kim, M. Muir, Mark N. Murgai, J. Nguyen, N. Pisenti, A. Schouela, K. Sosnova and K. Wright. Generative quantum learning of joint probability distribution functions. *Bulletin of the American Physical Society*, 2022.

[313] D. Zhu, N.M. Linke, M. Benedetti, K.A. Landsman, N.H. Nguyen, C.H. Alderete, A. Perdomo-Ortiz, N. Korda, A. Garfoot,

C. Brecque, L. Egan, O. Perdomo and C. Monroe. Training of quantum circuits on a hybrid quantum computer. *Science Advances*, 5(10), 2019.

[314] C. Zoufal, A. Lucchi and S. Woerner. Quantum generative adversarial networks for learning and loading random distributions. *Quantum Information*, 5(103), 2019.

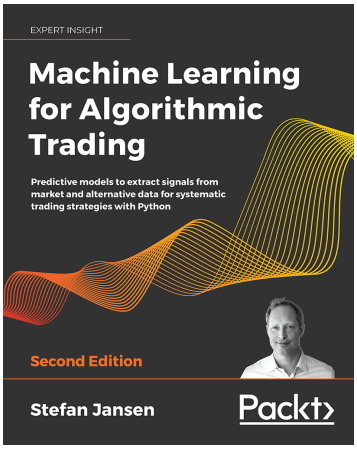
Join our book's Discord space

Join our Discord community to meet like-minded people and learn alongside more than 2000 members at: https://packt.link/quantum



Other Books You Might Enjoy

If you enjoyed this book, you may be interested in these other books by Packt:



Machine Learning for Algorithmic Trading - Second Edition

Stefan Jansen

ISBN: 978-1-83921-771-5

- Leverage market, fundamental, and alternative text and image data
- Research and evaluate alpha factors using statistics, Alphalens, and SHAP values
- Implement machine learning techniques to solve investment and trading problems
- Backtest and evaluate trading strategies based on machine learning using Zipline and Backtrader
- Optimize portfolio risk and performance analysis using pandas, NumPy, and pyfolio

- Create a pairs trading strategy based on cointegration for US equities and ETFs
- Train a gradient boosting model to predict intraday returns using AlgoSeek's high-quality trades and quotes data



Learn Quantum Computing with Python and IBM Quantum Experience

Robert Loredo

ISBN: 978-1-83898-100-6

- Explore quantum computational principles such as superposition and quantum entanglement
- Become familiar with the contents and layout of the IBM Quantum Experience
- Understand quantum gates and how they operate on qubits
- Discover the quantum information science kit and its elements such as Terra and Aer
- Get to grips with quantum algorithms such as Bell State, Deutsch-Jozsa, Grover's algorithm, and Shor's algorithm
- How to create and visualize a quantum circuit



Dancing with Qubits

Dr. Robert S. Sutor

ISBN: 978-1-83882-736-6

- See how quantum computing works, delve into the math behind it, what makes it different, and why it is so powerful with this quantum computing textbook
- Discover the complex, mind-bending mechanics that underpin quantum systems
- Understand the necessary concepts behind classical and quantum computing
- Refresh and extend your grasp of essential mathematics, computing, and quantum theory
- Explore the main applications of quantum computing to the fields of scientific computing, AI, and elsewhere
- Examine a detailed overview of qubits, quantum circuits, and quantum algorithm

Packt is searching for authors like you

If you're interested in becoming an author for Packt, please visit authors.packtpub.com and apply today. We have worked with thousands of developers and tech professionals, just like you, to help them share their insight with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Share Your Thoughts

Now you've finished *Quantum Machine Learning and Optimisation in Finance*, we'd love to hear your thoughts! If you purchased the book from Amazon, please click here to go straight to the Amazon review page for this book and share your feedback or leave a review on the site that you purchased it from.

Your review is important to us and the tech community and will help us make sure we're delivering excellent quality content.



www.packt.com

Subscribe to our online digital library for full access to over 7,000 books and videos, as well as industry leading tools to help you plan your personal development and advance your career. For more information, please visit our website.

Why subscribe?

- Spend less time learning and more time coding with practical eBooks and Videos from over 4,000 industry professionals
- Improve your learning with Skill Plans built especially for you
- Get a free eBook or video every month
- Fully searchable for easy access to vital information
- Copy and paste, print, and bookmark content

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at packt.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at customercare@packtpub.com for more details.

At www.packt.com, you can also read a collection of free technical articles, sign up for a range of free

Download a free PDF copy of this book

Thanks for purchasing this book!

Do you like to read on the go but are unable to carry your print books everywhere? Is your eBook purchase not compatible with the device of your choice?

Don’t worry, now with every Packt book you get a DRM-free PDF version of that book at no cost.

Read anywhere, any place, on any device. Search, copy, and paste code from your favorite technical books directly into your

application.

The perks don't stop there, you can get exclusive access to discounts, newsletters, and great free content in your inbox daily.

Follow these simple steps to get the benefits:

1. Scan the QR code or visit the link below:

   https://packt.link/free-ebook/9781801813570

2. Submit your proof of purchase
3. That's it! We'll send your free PDF and other benefits to your email directly

Index

A

activation functions 192

activation unit 186

AdaBoost 167

addition operator
  building from NAND gates 272

Adiabatic Quantum Computing (AQC) 88
  implementation 111
  power 96
  principles 92
  universality 124

adjustable one-qubit gates 295

adjustable two-qubit gates 304

amplitude estimation 679

analytic gradient approach 414

ancillary qubits 389, 651

AND gate 268

ansatz 416, 439, 531, 534

Artificial Neural Network (ANN) 186

artificial neuron 186

B

backpropagation 190

bagging 446

Bayesian Quantum Circuit (BQC) 650

Bell circuit 318

Bell states 318

binary digit (bit) 267, 43

binary integer linear programming problem 137

Black-Scholes parabolic PDE 689

Bloch sphere 287

Boltzmann distribution 223, 209

Boltzmann sampling 240

bootstrap 446

C

Chimera graph 251, 699, 120

classical benchmarking 444, 484, 154, 186

client clustering 578

client segmentation 578

CMOS logic 274

coherence 360

completeness condition 52

complexity classes 88

computation 273, 286, 50

computational basis 283, 53

Conditional Rotation Lemma 674

confusion matrix 183

controlled gates 297

cost function 430, 472, 480, 116, 130

CR gate 352

CX (CNOT) gate 299
  decomposition 323, 327, 352

CY gate 298

CZ (CPHASE) gate 301

CZ (CPHASE) gate
decomposition 323

D

D-Wave quantum annealers
architecture 120
inter-generational comparison 113

data encoding
amplitude encoding 382
angle encoding 374

data encoding
binary inputs into basis states 385
Hamiltonian encoding 397
superposition encoding 389

dataset
Australian Credit Approval (ACA) 439, 607, 639
Default of Credit Card Clients (DCCC) 177
King+Rook vs. King+Pawn 259
Stock Exchange 234, 491

Deep Boltzmann Machine (DBM)
expressive power 627
pipeline 259
training 257

discrete portfolio optimisation
solving with VQE 550

D-Wave quantum annealers 699, 116

decision tree 195

Deep Boltzmann Machine (DBM) 256

density matrix 618, 66

density operator 65

Dirac notations 36

discrete portfolio optimisation 148

DiVincenzo criteria 329

E

ensemble learning
majority voting 449
quantum boosting 454, 168

entropy
Gibbs (statistical mechanics) 311, 617
Shannon (information) 311, 617, 197
von Neumann (entanglement) 618, 624

expectation value
calculation on a quantum computer 511

effective dimension 617

elementary memory cell 276

energy function 209

energy-based models 210

ensemble learning 449

entanglement 314, 351, 61

error correction 361

expectation value 508, 59

external couplers 121

F

feature map 474, 638

feedforward neural network 188

fidelity 360, 637

finite difference scheme 411

Fisher information 617

forward quantum annealing 142

Fund of Funds portfolio manager 150

G

Generative Adversarial Network (GAN) 490, 643

Genetic Algorithm (GA)
QCBM training algorithm 477

graph theory
directed graph 206
undirected graph 206

generative models 255

Genetic Algorithm (GA) 476, 154

Gibbs distribution 209

Gibbs state 663

global phase 326, 328

gradient boosting 200

graph partitioning problem 134

graph theory 206

H

Hamiltonian
final 567, 117
initial 567, 118
Ising 248, 117
longitudinal field 107, 120
mixing 572
phase 572
transverse field 107, 120

H (Hadamard) gate 293, 57

Hamiltonian simulation 397, 25

Hamiltonian 47

Hammersley-Clifford Theorem 207

Hamming distance 109

Harrow-Hassidim-Lloyd (HHL) 683

Hermitian conjugate 3

Hermitian operator 9, 60

I

I gate 289

IBM Melbourne system graph 464

impurity
Gini 197

IBM Rochester system graph 466

impurity 196

information gain 195

inner product 5

internal couplers 121

Ising problem 131, 175

iSWAP gate 303

J

Josephson junction 347

K

k-step Contrastive Divergence (CD) algorithm 232

kernel function 475, 633

kernel trick 633

knapsack with integer weights problem 139

Kolmogorov-Smirnov (KS) 496

L

linear classifier 445

linear operator 8

Lipschitz constant 603

logic gates 268

logical qubits 122

logistic regression 445

M

majority voting 446, 449, 123

market generator 489

Markovian property 207

Matrix Product State (MPS) 624

Max-Cut problem 576

measurement operator 291, 52

measurement process 57

metaheuristic 105

Minimax Theorem 659

mixed state 618, 72

Monte Carlo
  classical 672
  quantum speedup 679
  quantum 673

Multilayer PQC (MPQC)
  expressive power 627

Modern Portfolio Theory (MPT) 148

Multi-Layer Perceptron (MLP) 444, 608, 188, 200

Multilayer PQC (MPQC) 613

N

NAND gate 268

natural unit of information (nat) 312

NP-complete problem 89, 116, 149

NP-hard problem 90, 116, 164

NISQ QPU
  connectivity 436

NISQ QPU
  Max-Cut problem, embedding on 592
  QCBM, embedding on 464, 465, 436, 436, 455

number partitioning problem 133

O

one-qubit quantum logic gates 289

OR gate 268

outer product 37

parameter shift rule 419

P

Parameterised Quantum Circuit (PQC) 369

Partial Differential Equation (PDE) 683

partial trace 620

Parameterised Quantum Circuit (PQC)
expressive power 610, 625
schematic training process 374

Particle Swarm Optimisation (PSO)
QNN training algorithm 429

postulates of quantum mechanics
composite system 60
dynamics 47
measurement 51
observable 58
statics 43

Particle Swarm Optimisation (PSO) 425

Pegasus graph 251, 699, 120

perceptron 188

phase shift gates 295

phonon 357

photonic qubits 353

physical qubits 121

polar coordinates 31

postulates of quantum mechanics 42

Powering Lemma 679

probability amplitude 280, 45

projection operator 7, 53

pure state 618, 67

Q

QBoost algorithm 169

Qiskit 362

QBoost algorithm
applications in finance 176
QUBO to Ising transformation 173

Quadratic Unconstrained Binary Optimisation (QUBO)
Ising transformation 131, 132

Quantum Boltzmann Machine (QBM)
mapping into hardware graph 244
quantum annealing and Boltzmann sampling 239

Quantum Circuit Born Machine (QCBM)
differentiable learning 466
embedding 463
non-differentiable learning 476

Quantum Neural Network (QNN)
embedding 436
training with gradient descent 410
training with PSO 425

Quadratic Unconstrained Binary Optimisation (QUBO) 129

Quantum Adiabatic Theorem 96

quantum advantage 494, 628, 699

Quantum Anharmonic Oscillator (QAO) 346

Quantum Annealing for Machine Learning (QAML) 169

quantum annealing 239, 107, 111

Quantum Approximate Optimisation Algorithm (QAOA) 563

quantum binary digit (qubit) 279, 44

Quantum Boltzmann Machine (QBM) 205

quantum boosting 454, 169

Quantum Circuit Born Machine (QCBM) 459

quantum circuits 291

Quantum Convolutional Neural Network (QCNN) 628, 650

Quantum Fourier Transform (QFT) 665

Quantum Generative Adversarial Network (QGAN) 644

quantum hardware 360

Quantum Harmonic Oscillator (QHO) 332

quantum kernel 474, 634

Quantum Linear Solver (QLS) 683

quantum logic gates 279, 46, 50

Quantum Long Short Term Memory (QLSTM) 628

Quantum Neural Network (QNN) 405

quantum operators 37

Quantum Phase Estimation (QPE) 668

quantum registers 291

Quantum Semidefinite Programming (QSDP) 660

quantum simulators 360

Quantum Software Development Kit (QSDK) 363

quantum speedup 679, 161

quantum state vector 43, 63

quantum state 43, 63

quantum tunnelling 108, 110

Quantum Variational Autoencoder (QVA) 255

R

radial basis function 633

Random Access Memory (RAM) 276

random forest 445

reduced density matrix 620

regularisation 602

relay logic 274

resistor-transistor logic 274

Restricted Boltzmann Machine (RBM)
  expressive power 627
  samples generation 234
  training with CD algorithm 230

Restricted Boltzmann Machine (RBM) 210

reverse quantum annealing 146

reversible computing 310

Rigetti Aspen system graph 437

rotation matrices 27

S

S gate 295

Schrödinger equation 338, 397, 564, 47

Schur Decomposition 19

search algorithms 87

Semidefinite Programming (SDP) 655

Sharpe ratio 152

simulated annealing 105

Singular Value Decomposition (SVD) 17

Spectral Theorem 18

superconducting qubits 332

superposition 314, 44

supervised learning 189

Semidefinite Programming (SDP)
  maximum risk analysis 656
  robust portfolio construction 657

statistical ensemble
  of quantum states 65, 77

SWAP gate
  decomposition 323, 325

Support Vector Machine (SVM) 444, 632

Suzuki-Trotter expansion 570

SWAP gate 302

synthetic data generation 234, 489

T

T gate 295

Tensor Network PQC (TPQC)
expressive power 627

Toffoli gate
decomposition 308

Tensor Network PQC (TPQC) 615

tensor product 40

thermal annealing 110

Thompson formula 345

time evolution, QAOA 564

Time-to-Solution (TTS) 113, 161

Toffoli gate 305

transition probability 637

transverse field coefficient 107

trapped ion qubits 356

truth table 268

Turing machine 89

two-qubit quantum logic gates 297

U

unit cell 121

unitary operator 49

universal logic gate 270

V

Vapnik-Chervonenkis dimension 617

Variational Quantum Eigensolver (VQE) 505

volatile memory 276

X

X (NOT) gate 289

XOR gate 268

XY gate 304

Y

Y gate 289

Z

Z (PHASE) gate 289

Zephyr graph 699

ZX gate 352

Quantum Machine Learning and Optimisation in Finance

- Quantum Machine Learning and Optimisation in Finance
  - Foreword
  - Endorsements
  - Contributors
    - About the authors
    - About the reviewer
    - Join our book’s Discord space
  - Table of Contents
    - Standard notations
    - Standard abbreviations
  - Preface
    - Introduction
    - Who this book is for
    - What this book covers
    - To get the most out of this book
    - Conventions used
    - Get in touch
    - Share your thoughts
    - Download a free PDF copy of this book
  - Chapter 1: The Principles of Quantum Mechanics
    - 1.1 Linear Algebra for Quantum Mechanics
    - 1.2 Postulates of Quantum Mechanics

1.3 Pure and Mixed States
Summary
Join our book’s Discord space

Part I: Analog Quantum Computing – Quantum Annealing

Chapter 2: Adiabatic Quantum Computing

2.1 Complexity of Computational Problems
2.2 Principles of Adiabatic Quantum Computing
2.3 Implementations of AQC
2.4 Universality of AQC
Summary
Join our book’s Discord space

Chapter 3: Quadratic Unconstrained Binary Optimisation

3.1 Principles of Quadratic Unconstrained Binary Optimisation
3.2 Forward and Reverse Quantum Annealing
3.3 Discrete Portfolio Optimisation
Summary
Join our book’s Discord space

Chapter 4: Quantum Boosting

4.1 Quantum Annealing for Machine Learning
4.2 QBoost Applications in Finance
4.3 Classical Benchmarks
Summary
Join our book’s Discord space

Chapter 5: Quantum Boltzmann Machine

5.1 From Graph Theory to Boltzmann Machines
5.2 Restricted Boltzmann Machine
5.3 Training and Running RBM
5.4 Quantum Annealing and Boltzmann Sampling
5.5 Deep Boltzmann Machine
Summary
Join our book’s Discord space

Part II: Gate Model Quantum Computing

Chapter 6: Qubits and Quantum Logic Gates

6.1 Binary Digit (Bit) and Logic Gates
6.2 Physical Realisations of Classical Bits and Logic Gates
6.3 Quantum Binary Digit (Qubit) and Quantum Logic Gates
6.4 Reversible Computing
6.5 Entanglement
6.6 Quantum Gate Decompositions
6.7 Physical Realisations of Qubits and Quantum Gates
6.8 Quantum Hardware and Simulators
Summary
Join our book’s Discord space

Chapter 7: Parameterised Quantum Circuits and Data Encoding

7.1 Parameterised Quantum Circuits
7.2 Angle Encoding
7.3 Amplitude Encoding
7.4 Binary Inputs into Basis States

7.5 Superposition Encoding
7.6 Hamiltonian Simulation
Summary
Join our book’s Discord space

Chapter 8: Quantum Neural Network

8.1 Quantum Neural Networks
8.2 Training QNN with Gradient Descent
8.3 Training QNN with Particle Swarm Optimisation
8.4 QNN Embedding on NISQ QPU
8.5 QNN Trained as a Classifier
8.6 Classical Benchmarks
8.7 Improving Performance with Ensemble Learning
Summary
Join our book’s Discord space

Chapter 9: Quantum Circuit Born Machine

9.1 Constructing QCBM
9.2 Differentiable Learning of QCBM
9.3 Non-Differentiable Learning of QCBM
9.4 Classical Benchmark
9.5 QCBM as a Market Generator
Summary
Join our book’s Discord space

Chapter 10: Variational Quantum Eigensolver

10.1 The Variational Approach
10.2 Calculating Expectations on a Quantum Computer

10.3 Constructing the PQC
10.4 Running the PQC
10.5 Discrete Portfolio Optimisation with VQE
Summary
Join our book’s Discord space

Chapter 11: Quantum Approximate Optimisation Algorithm

11.1 Time Evolution
11.2 The Suzuki-Trotter Expansion
11.3 The Algorithm Specification
11.4 The Max-Cut Problem
Summary

Chapter 12: The Power of Parameterised Quantum Circuits

12.1 Strong Regularisation
12.2 Expressive Power
Summary
Join our book’s Discord space

Chapter 13: Looking Ahead

13.1 Quantum Kernels
13.2 Quantum Generative Adversarial Networks
13.3 Bayesian Quantum Circuit
13.4 Quantum Semidefinite Programming
13.5 Beyond NISQ
Summary

Bibliography

Join our book’s Discord space

Other Books You Might Enjoy

Packt is searching for authors like you

Share Your Thoughts

Why subscribe?

Download a free PDF copy of this book

Index