TUNISIAN REPUBLIC
Ministry of Higher Education and Scientific Research
University of Carthage
National Institute of Applied Sciences and Technology

Personal Professional Project

as part of the

National Engineering Diploma
in Applied Sciences and Technology

Specialty: Computer Networks and Telecommunications Engineering

---

Optimizing Restricted Boltzmann Machines
via Quantum Annealing in High-Dimensional Hilbert Spaces

Prepared by

Iheb Gafsi
iheb.gafsi@insat.ucar.tn
National Institute of Applied Sciences
and Technology

Hanine Ben Amor
Hanine.benamor@insat.ucar.tn
National Institute of Applied Sciences
and Technology

Nermine Cheriaa
nermine.cheriaa@insat.ucar.tn
National Institute of Applied Sciences
and Technology

Defended on the 3rd June 2025 in front of a jury composed of:

Foulen el fouleni	President of the jury
Dr. Wided Miled Souid	Supervisor at INSAT

Acknowledgments

I. Gafsi gratefully acknowledges the support from the UTC Scholars, INSAT Fellows, and Google Quantum AI programs. H. Ben Amor is supported in part by the Mathematics and Computer Science Department at INSAT and C. Nermine gratefully acknowledge the support of the Ministry of Higher Education and Scientific Research through the Cooperative Agreement Research was sponsored by the INSAT laboratory and was accomplished under Cooperative Agreement. The authors are grateful to Wided Miled Souid for her valuable feedback on this work.

Abstract

The success of classical machine learning algorithms in the 20s of the 21st century has engendered the expansion of numerous fields. However, it also presented a critical limitation in the form of the curse of dimensionality and computational complexity. This has led to the scientific acceleration of quantum approaches due to their low complexity. In particular, in neuromorphic computing, inspired by the success of quantum Boltzmann distribution of a transverse field Ising Hamiltonian and rooted by the principles of quantum statistical mechanics, we propose a quantum restricted Boltzmann machine (QBM) that is able to create accurate quantum deep memory representations of data via quantum annealing and quantum Boltzmann sampling following a quasistatic evolution to a stable physical equilibrium. We circumvent the limitation of nontriviality due to the non-commutative nature of quantum mechanics. This allows us to obtain a quantum Boltzmann distribution that closely approximates the target data manifold even in regimes where classical samplers struggle. Additionally, we propose a set of practices to train an annealing-based QBM in high-dimensional Hilbert spaces with a plausible tradeoff between the data representability and computational cost.

Keywords: quantum annealing, boltzmann machines, sampling, high-dimensional Hilbert spaces.

Contents

Acknowledgments

Introduction **1**

1 Literature review **3**

1 Quantum Information 4

1.1 Quantum Algorithms and Quantum Advantage 5

1.2 Quantum Systems for Data Encoding 6

1.2.1 Qubits 6

1.2.2 Superposition 8

1.2.3 Single Qubit Superposition 9

1.2.4 Multi-Qubit Superposition 10

1.2.4.1 Two-Qubit Case 10

1.2.4.2 Classical vs. Quantum Approaches 10

1.2.4.3 Tensor-Product Structure 11

1.2.4.4 Scaling Implications 11

1.2.5 Encoding Bits into Qubits 12

1.2.6 Qudits 12

1.2.7 Qumodes 13

1.2.8 Operations in The Frame of Qumodes 13

1.2.9 Embedding into QUBO Problems 14

1.3 Processing Quantum Information with The Circuital Model . . . 14

1.3.1 Quantum Gates 15

1.3.1.1 Single-Qubit Gates 15

1.3.1.2 Multi-Qubit Gates 16

1.3.2 Quantum Circuits . . . 16
1.3.3 Quantum Correlations . . . 18
1.3.4 Quantum Entanglement . . . 18
1.3.5 Heisenberg's Uncertainty Principle . . . 19
1.3.6 High-Dimensional Hilbert Spaces . . . 20
1.4 Processing Quantum Information with the Adiabatic Model . . . 20
1.4.1 Quantum Annealing . . . 20
1.4.2 Applicable Problems . . . 21
1.4.3 Double Well Potential . . . 21
1.4.4 The Hamiltonian and the Eigenspectrum . . . 22
1.4.5 Annealing in Low-Energy States . . . 22
1.4.6 Evolution of Energy States . . . 23
2 Quantum Machine Learning . . . 24
2.1 Emulation of Quantum Computing Resources by HPC . . . 25
2.2 Generalization of Neural Networks in Quantum Circuits . . . 25
2.2.1 Variational approach . . . 26
2.2.2 Neurons into Qubits . . . 26
2.2.2.1 1-to1 Encoding . . . 26
2.2.3 n-to-2^n (Superposition Coefficients) Encoding . . . 27
2.3 Quantum Supervised Learning . . . 27
2.3.1 Quantum FFNN for binary decisions: . . . 27
2.4 Quantum Approximate Optimization Algorithm . . . 28
2.4.1 NP-Hard Problems . . . 29
2.4.2 The MaxCut problem . . . 29
2.4.3 QAOA Formulation . . . 30

2 Work Methodology 32

1 Restricted Boltzmann machines and Deep Belief Network . . . 35
1.1 Probabilistic graphical model and Markov random field . . . 36
1.2 Gibbs Sampling . . . 36
1.3 Statistical Physics and Ising Model . . . 37
1.3.1 Boltzmann-Gibbs Distribution . . . 38

1.4 Hopfield Network . . . 39
1.5 Structure of Restricted Boltzmann Machine . . . 40
1.6 Conditional Distributions . . . 41
1.7 Sampling Hidden and Visible Variables . . . 42
1.8 Training Restricted Boltzmann Machine by Maximum Likelihood Estimation . . . 42
1.9 Contrastive Divergence . . . 43
1.10 Conditional Restricted Boltzmann Machine . . . 43
1.11 Deep Belief Network . . . 43
2 Quantum Approach in High-Dimensional Hilbert Spaces . . . 45
2.1 Quantum Boltzmann machine . . . 45
2.1.1 Bound-Based QBM . . . 48
2.1.2 Restricted QBM . . . 50
2.2 QBM with Quantum Annealing Processor in High-Dimensional Hilbert Spaces . . . 51
2.2.1 Quasistatic Evolution and Freeze-Out Process . . . 51
2.2.2 Hilbert Spaces Mathematical Framework . . . 53
2.2.3 High-Dimensional Hilbert Spaces Intrinsic Properties . . . 53
2.2.4 High-Dimensional Hilbert Spaces in QBM . . . 53
3 Conclusion . . . 54

3 Experiments and Results 56

1 Datasets . . . 56
1.1 AudioMNIST . . . 57
1.2 MNIST . . . 58
1.3 Synthesized Quantum Data . . . 59
1.3.1 Reproducibility . . . 62
1.3.2 Discretization and Quantization . . . 62
1.3.3 Binarization . . . 62
1.3.4 Data Reconstructibility (Undiscretization) . . . 62
1.3.5 Mini-Batching . . . 63
1.3.6 Monte Carlo Sampling `n samples` per Iteration . . . 63

1.3.7 Learning-Rate Scheduling and Exponential Decay 63
2 Experimental setup 63
2.1 QC Emulation 64
2.2 QPU setup 64
3 Results 65
3.1 Restricted Boltzmann Machines and Variational AutoEncoder . . 66
3.2 Quantum Distribution Preservation 67
3.3 Quantum Annealing on Quantum Distributions 71
3.4 The Effect of the Dimension of the Hilbert Space and Quantum Noise 74

Conclusion and perspectives **78**

A Appendix A: Quantum Technical Concepts and Definitions **89**
1 Hilbert Space 89
2 State Space and State Vector 90
3 Schrödinger's Equation 90
4 Hermitian Matrix/Operator 90
4.1 Hamiltonian 90
5 Observable 91
6 Heisenberg's Uncertainty Principle 91

B Appendix B: Mathematical Framework **92**

List of Figures

1.1 (a) Classical bit and (b) quantum bit (qubit) with α and β as complex respective amplitudes of the state-vector. creating a superposition state bound by the Max-Born rule $|\alpha|^2 + |\beta|^2 = 1$, which satisfies the completeness equation in which states have all probabilities summing up to 1 [Nielsen and Chuang, 2010]. 7

1.2 Bloch sphere representation of a qubit. It defines the geometrical boundaries of the qubit states. 7

1.3 A few examples for single qubit Superposition cases. 10

1.4 Examples of two-qubit probabilistic superpositions: (a) Equal probabilities (25% for $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$); (b) 50% probability for $|00\rangle$ and 50% for $|10\rangle$; (c) 50% probability for $|01\rangle$ and 50% for $|11\rangle$; (d) 25% probability for $|00\rangle$, 50% for $|01\rangle$, 25% for $|10\rangle$. 11

1.5 A simple visual model for the Bell states for two qubits in a non-local interaction 19

1.6 Annealing process's energy diagram showing the raising of the energy barrier for a single qubit, resulting in a 50/50 probability of ending in a classical state of 0 or 1. 22

1.7 Energy spectrum showing ground state and excited states during annealing. 23

1.8 Annealing schedule showing $A(s)$ (tunneling energy) and $B(s)$ (problem energy) as functions of anneal fraction s. At $s = 0$, $A(s) \gg B(s)$, and at $s = 1$, $A(s) \ll B(s)$. 24

1.9 Classification of quantum machine learning algorithms. 25

2.1 General, restricted, and deep Boltzmann machine architectures. The visible units are represented by circles, while the hidden units are represented by squares. The connections between the visible and hidden units are represented by lines. 33

2.2 Pre-training a deep belief network by considering every pair of layers as an RBM. 44

2.3 Quantum annealing vs classical thermal annealing. The quantum annealer follows a quasistatic evolution until the dynamics become too slow to maintain equilibrium. The system then deviates from the equilibrium distribution and soon after the dynamics freeze. 51

3.1 The logarithmic KL divergence between the validation data and the model reconstructions for the CRBM (blue) and the VAE (green). The CRBM converges quickly to a lower KL divergence within the first 10 iterations and maintains stability with slight fluctuations, while the VAE exhibits persistently high and noisy KL divergence throughout their performance differences. 66

3.2 CRBM Reconstruction vs VAE Reconstruction 67

3.3 A visualisation of 10 RBM weights 67

3.4 The logarithmic KL divergence between the empirical training distribution and the model samples for the simulated QBM (blue) and the classical RBM (green). The QBM converges rapidly to a lower KL divergence while maintaining of fluctuating KL in contrast, indicating better 69

3.5 The Reconstruction Error per iteration for the simulated QBM (blue) and the classical RBM (green). The simulated QRBM is 6 times more efficient than the RBM. 70

3.6 The wall-clock training time per iteration for the simulated QBM (black) and the classical RBM (red). The CRBM is approximately 4 times slower than the simulated QBM. 71

3.7 The KL divergence between the empirical training distribution and the model samples for the simulated QBM (blue) and the annealer-based QBM (orange). The annealer-based QBM converges to a lower KL divergence, indicating a closer fit to the data distribution, but is noisier across epochs. 72

3.8 The wall-clock training time per iteration for the simulated QBM (blue) and the annealer-based QBM (orange). The annealer is between 150 and 450 times slower per iteration than the simulator, reflecting both the communication overhead and the limited bandwidth of current annealing hardware. 73

3.9 The effective temperature parameter during training for the simulated QBM (blue) and the annealer-based QBM (orange). The temperature curve of the annealer-based training remains higher throughout all iterations, consistent with the fact that the device operates at a fixed physical temperature and samples from a frozen, non-equilibrium distribution. . . 74

3.10 The effect of the dimension of the Hilbert space and quantum noise on the performance of the QBM. The QBM is able to learn complex distributions even in high-dimensional Hilbert spaces, but the performance degrades as the noise level increases. 75

List of Tables

1.1 Quantification of speed-ups for given machine learning subroutines. The table is taken from [Biamonte et al., 2017] and some of the algorithms will be further discussed rigorously in the next section. 6

List of acronyms

- **ANNS** Approximate Nearest Neighbor Search
- **bQBM** Bound-based Quantum Boltzmann Machine
- **BQP** Bounded-Error Quantum Polynomial Time
- **CD** Contrastive Divergence
- **CRBM** Conditional Restricted Boltzmann Machine
- **CV** Continuous Variables
- **DBNs** Deep Belief Networks
- **HNNs** Hopfield Neural Networks
- **MLE** Maximum Likelihood Estimation
- **MLP** Multi-Layer Perceptron
- **MCMC** Markov Chain Monte Carlo
- **MRF** Markov Random Field
- **NLL** Negative Log Likelihood
- **PCA** Principal Component Analysis
- **PGM** Probabilistic Graphical Model
- **QAOA** Quantum Approximate Optimization Algorithm
- **QBM** Quantum Boltzmann Machine
- **QML** Quantum Machine Learning
- **QNN** Quantum Neural Network
- **qSVM** Quantum Support Vector Machine
- **RBM** Restricted Boltzmann Machine

- **RL** Reinforcement Learning
- **VQC** Variational Quantum Circuit

Introduction

In statistical mechanics and mathematics, a Boltzmann distribution is a probability measure that attributes the probability that a system will be in a certain state as a function of that state's energy and the temperature of the system. It appears in statistical mechanics when considering closed systems of fixed composition that are in thermal equilibrium with respect to energy exchange. The most general case is the probability distribution for the canonical ensemble. This energy-based distribution was found to be useful for modeling physical systems statistically. One of these systems was the Ising model, which modeled interacting particles with binary spins [Ising, 1925b]. Hence, the Hopfield network was proposed, which modeled the Ising model in a network for modeling memory [Hopfield, 1982, Little, 1974b]. Inspired by this network, which was itself inspired by the physical Ising model, Hinton et. al proposed Boltzmann Machine (RBM) [Ackley et al., 1985, Hinton and Sejnowski, 1983b]. A Boltzmann machine (BM) features weighted connections both between its two neuron layers and among neurons within each layer, whereas a restricted Boltzmann machine (RBM) forbids any intra-layer links. In both BM and RBM, one layer serves as the data input and the other as its learned representation or embedding. These architectures can be viewed as special cases of the Ising model, with their coupling weights learned during training, and likewise as Hopfield networks whose weights are fitted by maximum-likelihood estimation rather than by the classical Hebbian update rule [Cooper, 1994]. The Hebbian learning method, used in Hopfield network, was very weak and could not generalize well to unseen data. Therefore, backpropagation [Rumelhart et al., 1986b] was proposed for training neural networks. Backpropagation was gradient descent plus the chain rule technique that was further developed over time for more complex update rules. However, researchers soon found out that neural networks cannot get deep in their number of layers. This is due to the vanishing gradient problem [Boser et al., 1992].

Hopfield networks introduced a revolution in the field of neural networks and inspired the development of deep learning, which, in turn, led to the technological revolution as we know it today and serves as the core of most advanced machine learning algorithms such as Large Language Models (LLMs). However, we reached a bottleneck in the field of deep learning with the computational complexity curse hindering the way to develop responsible and efficient solutions that serve humanity as a whole. Quantum computing, meanwhile, has been a hot topic in the scientific community for the last decade and has been the area of intense research and development. The possibility of using quantum computation for machine learning has been considered theoretically for both gate model [Lloyd et al., 2013, Rebentrost et al., 2014] and quantum annealing [Neven et al., 2008]. It has become possible to test machine learning ideas with an actual quantum hardware.

In this work, we reinforce the idea of using a quantum probabilistic model for machine learning based on Boltzmann distribution of a quantum Hamiltonian, therefore, a Quantum Boltzmann Machine (QBM). We circumvent the utilization of classical-quantum hybridization of Boltzmann distributions, and we discuss the best practices to train an annealing-based QBM in High-Dimensional Hilbert spaces. We also show that the QBM can be used to solve the problem of quantum state tomography, which is a fundamental problem in quantum information theory. We demonstrate the effectiveness of our approach on several benchmark datasets.

1

Literature review

Introduction

Machine Learning (ML) systems are well-established tools for identifying patterns in data and generalizing complex, nonlinear problems, and these systems have found applications in various domains, including computer vision, healthcare, finance, and the automotive industry. However, ML practitioners must navigate the substantial computing resources required to run large ML models, despite employing numerous hardware-aware optimizations such as compression and approximations Capra et al. [2018], Hanif et al. [2018, 2022], Leon et al. [2023], Marchisio et al. [2019]. According to Moore's law, the number of transistors on an integrated circuit doubles approximately every two years; however, this trend has reached its limits with current high-end CPUs and GPUs Gustafson [2011]. This physical saturation restricts computational power, leading to delays in processing, development, and scientific discovery within the ML community. For instance, training Large Language Models with hundreds of billions of parameters and trillions of tokens is extremely compute-intensive OpenAI [2023].

To overcome these physical limits and support further discoveries, there is an urgent need to explore new technological avenues of hardware systems that can enhance the computational efficiency for solving real-world problems by closely simulating and comprehending them. One of the most promising solutions to this bottleneck is Quantum Computing (QC). Initially proposed by Feynman Feynman [1982] and further developed by Preskill Preskill [2018, 2022], QC systems exploit quantum mechanical phenomena to significantly improve performance and information processing compared to classical systems. Unlike classical systems that struggle to encompass natural processes, quantum computers operate on similar principles to those found in nature. This dissimilarity suggests that QC could help us better understand natural phenomena and, moreover, solve problems more cost-efficiently through optimizations, time savings, and environmentally friendly designs.

The Quantum Machine Learning (QML) paradigm represents an excellent opportunity for researchers and industry to achieve remarkable discoveries and design efficient solutions for complex real-world problems. QML systems, driven towards practicality and improved performance over classical systems, open new avenues for the community to discover, build, and align their designs across different levels of the quantum stack Schuld and Killoran [2022]. In this literature review, we systematically examine the foundations of quantum computation, data encoding strategies, and state-of-the-art QML architectures, highlighting both theoretical advances and practical implementations.

1 Quantum Information

Quantum Information Theory brings together ideas from Classical Information Theory, Quantum Mechanics, and Computer Science. Theorems and techniques from various branches of Mathematics and Mathematical Physics, in particular Group Theory, Probability Theory, and Quantum Statistical Physics, find applications in this field. While Classical Information Theory [Shannon, 1948] is the mathematical theory of information-processing tasks such as storage and transmission, Quantum Information Theory is the study of how such tasks can be accomplished using quantum mechanical systems.

1.1 Quantum Algorithms and Quantum Advantage

The quantum Turing machines [Deutsch, 1985] defined the computational classes in which quantum algorithms belong instead of the conventional ones. In the complexity theory, for both classical and quantum computation, the runtime of an algorithm is measured in terms of a number of elementary operations N involved [Montanaro, 2016]. These operations match the application of native gates on the hardware for gate-based architectures in the circuital models. Thus, the same problem can be mapped as nondeterministic polynomial time (NP) but not polynomial time (P) for classical machines while it can be, if defined on the Hilbert spaces on which quantum Turing machines rely, of class bounded error quantum polynomial time (BQP). Jager and Kerms proved that there exists a feature map and a quantum kernel that make variational quantum classifiers (VQC) and quantum kernel support vector machines (qSVM) efficient solvers for any BQP problem. Thus, with the adequate quantum approach, classical NP problems can be solved exponentially faster. Different degrees of speed-up have been defined by a panel of experts [Rønnow et al., 2014], which can be summarized as follows:

- *Provable quantum speed-up*: there is a proof that there can be no classical algorithm that performs as well or better than the quantum algorithm, like Grover's algorithm.
- *Strong quantum speed-up*: the quantum algorithm performs better than the best possible classical algorithm for a specific problem, like Shor's algorithm for prime number factorization.
- *Common quantum speed-up*: the quantum algorithm performs better than the best available classical algorithm, like the quantum approximate optimization algorithm (QAOA) for the MaxCut problem.
- *Potential quantum speed-up*: there is no consensus about which is the best available classical algorithm. It can be defined as a comparison with an arbitrary classical algorithm.
- *Limited quantum speed-up*: it refers to the benchmark of two corresponding algorithms, like classical and quantum annealing.

In the literature, there are numerous ways to implement quantum machine learning, among them, three approaches are common. First, by direct speed-up of machine learning techniques, using algebra-related algorithms like the HHL algorithm is superior [Bia-

monte et al., 2017]. Secondly, by implementing variational quantum circuits and finally multilayer perceptrons (MLP). Thirdly, explicit models rely on a parametric definition of unitary operators $\hat{U}(\theta)$, where θ is the set of parameters. Such models, therefore, can be easily encoded into any variational quantum circuit. The QAOQ algorithm is an example of an explicit model.

Methods	qRAM	Amplitude Amplification	HHL	Adiabatic	Speed-up
Bayesian inference	no	yes	yes	no	$O(\sqrt{N})$
Online perceptron	optional	yes	no	no	$O(\sqrt{N})$
Least-squares fitting	yes	yes	yes	no	$O(\log(N))$
Qantum PCA	optional	no	yes		$O(\log(N))$
Quantum SVM	yes	no	yes		$O(\log(N))$
Quantum RL	no	yes	no	no	$O(\sqrt{N})$
Classical BM	optional	yes/no	optional/no	no/yes	$O(\sqrt{N})$
Quantum BM	no	optional/no	no	no/yes	$O(\log(N))$

Table 1.1: Quantification of speed-ups for given machine learning subroutines. The table is taken from [Biamonte et al., 2017] and some of the algorithms will be further discussed rigorously in the next section.

1.2 Quantum Systems for Data Encoding

1.2.1 Qubits

The bit forms the fundamental processing unit of classical computing systems. As shown in Figure 1.1, a bit can hold one of the two mutually exclusive values, i.e., either 0 or 1. Where, on the other hand, a *quantum bit* or a *qubit* exists in a continuum of states (basis states). Understanding the qubit as an abstract mathematical object is crucial for the development of quantum circuits. Although the physical realization of qubits is important, the fundamental properties and specifications of qubits are dictated by quantum mechanical principles and remain consistent regardless of their physical form. The qubit is the fundamental unit of information encoded by a quantum computer and can be represented as a two-level quantum system in superposition of two basis states, which allows it to embody a range of probabilities for being in either state, reflecting its inherent quantum nature. This abstract representation of a qubit is consistent across any physical realization of the qubit, whether it is implemented using superconducting circuits, trapped ions, or any other technology. It is represented through the basis states $|1\rangle$ and $|0\rangle$, followed by the final probabilistic distribution over possible measured state outcomes as solutions to the problem [Marinescu, 2011].

(a) Classical Bit (b) Quantum Bit: Qubit

Figure 1.1: (a) Classical bit and (b) quantum bit (qubit) with α and β as complex respective amplitudes of the state-vector. creating a superposition state bound by the Max-Born rule $|\alpha|^2 + |\beta|^2 = 1$, which satisfies the completeness equation in which states have all probabilities summing up to 1 [Nielsen and Chuang, 2010].

The Bloch sphere is an intuitive visual representation of a two-level (two states per qubit) single qubit system. As shown in 1.2, it represents the pure and mixed states of a qubit. The pure states are distributed over the surface of the sphere, and all the internal states within the sphere are referred to as mixed-qubit states. It has three axes with their respective basis states on opposite ends.

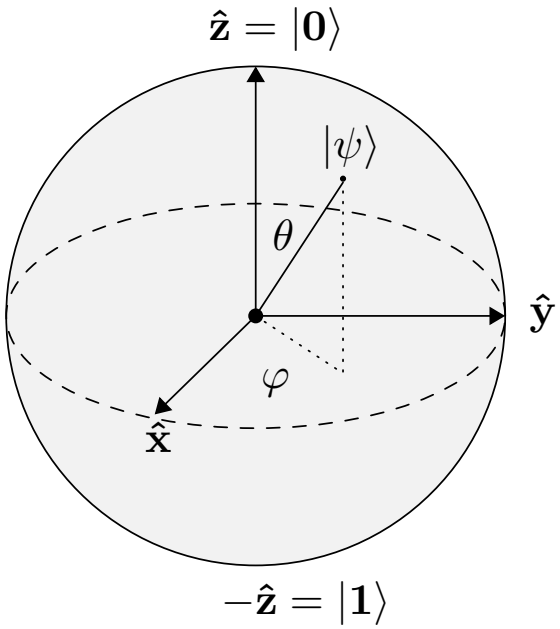

Figure 1.2: Bloch sphere representation of a qubit. It defines the geometrical boundaries of the qubit states.

Mathematically, the qubit is a quantum state of the system, which is defined by a vector $|\psi\rangle$ in a Hilbert space $\mathcal{H} = \mathbb{C}^2$. It is the transposition of the classical bit, but instead of having discrete values, the values are transposed ina vector state [Kopczyk, 2018]:

$$0 \to |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad 1 \to |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \Rightarrow |\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \alpha|0\rangle + \beta|1\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle \tag{1.1}$$

The $|0\rangle, |1\rangle$ family of vectors forms an orthonormal basis in the Hilbert space $\mathcal{H}$, therefore, any qubit can be a set of linear combination as in the equation 1.1. Thus, $\langle\psi|\psi\rangle = 1$, which is the normalization condition of the qubit. As, in the rightmost part of the equation, θ and ϕ are the polar and azimuthal angles of the Bloch sphere, respectively. The α and β coefficients represent the probability for the state to be found either in the $|0\rangle$ or $|1\rangle$ state, respectively.

All the logic operations on a single qubit are implemented by unitary operators $\hat{U}$, which in turn, transform such states from $\mathcal{H}$ to $\mathcal{H}$: $\hat{U}|\psi\rangle \rightarrow |\psi'\rangle$. The single-qubit operators $\hat{U}$ are given by the unitary group $SU(2)$ to preserve the normalization condition of the qubit. There are several most common single-qubit gates, which are the Hadamard gate H, the Pauli gates X, Y, and Z, and the phase gate S that we will define in detail in section 1.3.2.

The Hadamard gate $\hat{H}$ is an isomorphism on $\mathbb{C}^2$, a rotation of π around the Y axis of the Bloch sphere, and it is used to create superposition states. The Pauli gates are rotations of π around the X, Y, and Z axes of the Bloch sphere, respectively. The phase gate is a rotation of $\pi/2$ around the Z axis of the Bloch sphere.

1.2.2 Superposition

The superposition principle is one of the most intriguing and counterintuitive aspects of quantum mechanics. It states that a quantum system can exist in multiple states simultaneously until it is measured. An example of such, is the photons of light that can be polarized in two orthogonal states, horizontal and vertical, or in a superposition of both states. By passing the photon through a polarizer, it can be measured in either state with a certain probability. This is known as the wave-particle duality of light, where the photon behaves as both a wave and a particle.

The *Wave-Particle Duality* theory posits that waves can exhibit particle-like properties and vice versa, in contrast to classical Newtonian physics, where particles and waves are distinct entities [Loscri et al., 2024]. Quantum mechanics describes the behavior of particles at the atomic and subatomic levels, and it is consistent with all the experiments and observations that are made, like the double-slit experiment.

The space in which these states exist is called *Hilbert space*, a mathematical framework that allows the description of quantum states as vectors, and in which the state-vector

formalism lies. The *Max-Born rule* provides a probabilistic framework for quantum mechanics, it states that the probability density of finding a quantum system in a given state is proportional to the square of the amplitude of the system's wave function at that state. Similarly, the energy levels of free electrons in an atom illustrate superposition. An electron can exist in a superposition of the ground state $|0\rangle$ and an excited state $|1\rangle$, with possible intermediate states. In this sense, we can eventually measure qubits to fall into bits, section 1.2.5 will discuss in detail the encoding of bits into qubits.

1.2.3 Single Qubit Superposition

A single qubit exists in a superposition of the basis states $|0\rangle$ and $|1\rangle$ simultaneously. Quantum physics uses state vectors

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{1.2}$$

to mathematically describe the state of a single qubit system, where α and β are complex amplitudes representing the contribution of each basis state. These amplitudes determine the probability of the qubit collapsing to $|0\rangle$ or $|1\rangle$ when measured, with:

$$P(|0\rangle) = |\alpha|^2 \quad \text{and} \quad P(|1\rangle) = |\beta|^2 \tag{1.3}$$

These amplitudes must satisfy the normalization condition according to the Born rule:

$$|\alpha|^2 + |\beta|^2 = 1 \tag{1.4}$$

ensuring the total probability sums to unity.

various superposition states are possible for a single qubit system, resulting in probabilities. To use these probabilities, we should satisfy the property of Max-Born, and to meet these conditions, the amplitude values of the state vector must be normalized.

The figure 1.3 shows representative examples of single-qubit superposition states. Each block corresponds to an independent quantum system, illustrating the probabilistic outcomes of measurements from amplitude values α and β of respective superpositions through gradient shading.

Figure 1.3: A few examples for single qubit Superposition cases.

1.2.4 Multi-Qubit Superposition

The principles of superposition extend naturally to multi-qubit systems. For n qubits, the state is represented as:

$$|\psi\rangle = \sum_{i=0}^{2^n-1} c_i |i\rangle, \quad \text{where } |i\rangle = \underbrace{|0\rangle \otimes |0\rangle \otimes \cdots \otimes |0\rangle}_{n \text{ times}}. \tag{1.5}$$

Here, $c_i \in \mathbb{C}$ are complex amplitudes, and the system resides in a Hilbert space of dimension 2^n. This exponential scaling enables quantum parallelism but imposes significant challenges in classical simulation and resource management (discussed in Section 1.2.4.4).

1.2.4.1 Two-Qubit Case

A two-qubit system generalizes single-qubit superposition to four basis states:

$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \eta|11\rangle, \tag{1.6}$$

with the normalization constraint:

$$|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\eta|^2 = 1 \quad \text{(Born rule)}. \tag{1.7}$$

Figure 1.4 illustrates key examples of these probabilistic states.

1.2.4.2 Classical vs. Quantum Approaches

Consider locating an object among positions $A(00)$, $B(01)$, $C(10)$, $D(11)$ (figure 1.4).

- **Classical**: Sequentially checks states with deterministic outcomes (e.g., 100% confidence in position B)

- **Quantum**: Evaluates all states simultaneously via superposition, yielding probabilistic distributions (e.g., 60% B, 20% C, 10% A/D) reflecting pre-measurement amplitudes

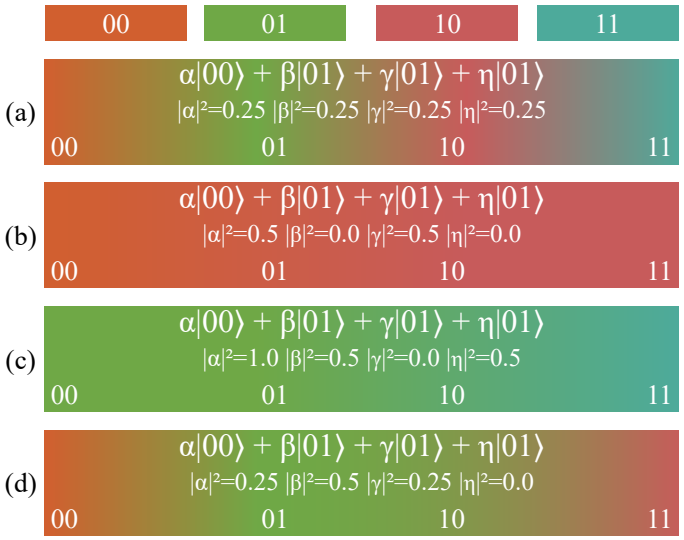

Figure 1.4: Examples of two-qubit probabilistic superpositions: (a) Equal probabilities (25% for $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$); (b) 50% probability for $|00\rangle$ and 50% for $|10\rangle$; (c) 50% probability for $|01\rangle$ and 50% for $|11\rangle$; (d) 25% probability for $|00\rangle$, 50% for $|01\rangle$, 25% for $|10\rangle$.

1.2.4.3 Tensor-Product Structure

Two qubits occupy the Hilbert space:

$$\mathcal{H}_2 \otimes \mathcal{H}_2 \cong \mathbb{C}^4. \tag{1.8}$$

For n qubits:

$$\underbrace{\mathcal{H}_2 \otimes \cdots \otimes \mathcal{H}_2}_{n \text{ times}} \cong \mathbb{C}^{2^n}, \tag{1.9}$$

with the all-zero basis state written compactly as $|0\rangle^{\otimes n} = |00\cdots0\rangle$.

1.2.4.4 Scaling Implications

Classical simulation becomes infeasible as:

$$\text{Memory Cost} \propto 2^n, \quad \text{Computation Cost} \propto (2^n)^2 \tag{1.10}$$

Doubling n squares both memory and computational requirements.

The 2^n-dimensional Hilbert space enables:

- Exponential state parallelism as it allows exponentially many states in superposition.
- Potential quantum speedups

But introduces critical challenges:

- **Measurement Collapse**: Only probabilistic outcomes ($|c_i|^2$) are observable
- **Resource Constraints**: Normalization ($\sum |c_i|^2 = 1$) and noise limit scalability
- **Algorithm Design**: Requires entanglement (qubit correlation) and interference (amplitude manipulation)

1.2.5 Encoding Bits into Qubits

There are several ways to encode bits into qubits. The most vanilla method consists of a one-to-one encoding, making one bit i to be encoded by one quantum state $|i\rangle$. In literature, such encoding is referred to as multi-register encoding. However, an enhancement can be given by the superposition principle: in a register of N qubits, it is possible to encode $n = 2^N$ bits. The multi-register and analog encoding are respectively shown in equation 1.11:

$$|i\rangle = |i_0\rangle \otimes |i_1\rangle \otimes ... \otimes |i_{N-1}\rangle, \qquad \sum_{i=0}^{2^N-1} c_i|i\rangle \tag{1.11}$$

Therefore, within the analog encoding (rightmost expression) via N qubits, it is possible to store $n = 2^N$ units of memory, which means N bits require $\log_2(N)$ qubits. One last encoding method is the *Hamiltonian encoding*. Within this frame, the data needs to be formatted into a Hamiltonian operator like the Ising Hamiltonian. This technique turns out to be useful mainly in order to perform quantum annealing, which we used in our project. This will be further explained in section 1.4.1

1.2.6 Qudits

The qudit is a generalization of the qubit, it is a d-level quantum system that can be used to represent more than two states. The qudit can be represented as a vector in a d-dimensional Hilbert space $\mathcal{H} = \mathbb{C}^d$. So, instead of a basis spanned by 2 vectors, the space of qudits is generated by D vectors. It is possible to choose a computation

basis $|0\rangle, |1\rangle, \ldots, |D-1\rangle$, thus a vector in the corresponding space can be represented analytically as [Konar et al., 2021]:

$$|\psi\rangle = \sum_{i=0}^{D-1} \alpha_i |i\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle + \cdots + \alpha_{D-1}|D-1\rangle \tag{1.12}$$

The state in Equation 1.12 is normalized, i.e., $\sum_{i=0}^{D-1} |\alpha_i|^2 = 1$. Qudits are particularly useful since, given M qudits, it is possible to store D^M units of binary memory. However, qudits are a nice value that we can add to Boltzmann machines, we did not include them in our experiments.

1.2.7 Qumodes

When dealing with continuous variables (CV) like position and momentum, the qubit representation is not sufficient. In this case, we need to use a different representation called *qumodes*. Quantum information is encoded in continuous degrees of freedom such as the amplitudes of the electromagnetic fields. In the phase space representation, the state of the single Qumode is described by two real and conjugate variables such as $(x, p) \in \mathbb{R}^2$, Qumodes states also have a representation as vectors or density matrices in the countably infinite Hilbert space spanned by the Fock states. Qumodes can be used to represent quantum states of light, such as coherent states and squeezed states, which are important in quantum optics and quantum information processing.

1.2.8 Operations in The Frame of Qumodes

Some possible operations to implement in the CV frame are reported by [Killoran et al., 2019], which can be summarized as follows: First, position and momentum operators are introduced:

$$\hat{X} = \int_{-\infty}^{+\infty} x|x\rangle\langle x|dx, \quad \hat{P} = -i\hbar \int_{-\infty}^{+\infty} \frac{\partial}{\partial x}|x\rangle\langle x|dx \tag{1.13}$$

Where $\hat{X}$ and $\hat{P}$ are the position and momentum operators, respectively. The position operator $\hat{X}$ is a Hermitian operator that represents the position of a particle in space, while the momentum operator $\hat{P}$ is an anti-Hermitian operator that represents the momentum of a particle. They are bound by:

$$\langle u|u'\rangle = \delta(u - u') \tag{1.14}$$

Where $\delta(u-u')$ is the Dirac delta function, which is a mathematical function that is zero everywhere except at $u = u'$, where it is infinite. The Dirac delta function is used to represent the inner product of two position states.

1.2.9 Embedding into QUBO Problems

Adiabatic quantum computers can find the optimal solution to a specific class of optimization problems: Quadratic Unconstrained Binary Optimization (QUBO) problems. A QUBO problem is mathematically described as:

$$\hat{H} = -\sum_{i=1}^{N} h_i \hat{\sigma}_i^z + \sum_{i=1}^{N} \sum_{j=i+1}^{N} J_{ij} \hat{\sigma}_i^z \hat{\sigma}_j^z = \hat{H}_{linear} + \hat{H}_{quadratic} \tag{1.15}$$

Where $\hat{\sigma}_i^z$ are the Pauli matrices that act along the z-direction, and J_{ij} and h_i represent the parameters of the problem to be solved. You can think of the hamiltonian $\hat{H}$ as the energy of the system that we are minimizing in this problem. Since the eigenvalues of the Hamiltonian H represent the possible solutions to the problem, the aim is to set the couplings J_{ij} and biases h_i so that the ground state of H represents the optimal state of the optimization problem. You can think of the QUBO problems to be represented by graphs, where nodes are associated with biases and edges with couplings. Ideally, we want to map the optimization problem graph directly into the quantum annealer QPU. This is analogous to how proteins fold into their optimal state in a matter of milliseconds, minimizing the energy of their amino acids, or how a system of magnets finds an optimal state. Ideally, we want to optimize the problem graph directly into the quantum annealer QPU.

1.3 Processing Quantum Information with The Circuital Model

A circuital model, following the Von Neumann-Zuse paradigm [Killoran et al., 2019], processes the inputs into outputs via a sequence of commands. Such architectures, can be labeled as circuital model, as both of them aim to install a circuit of logical, controlled, and sequential operations on the qubits. This makes it take two out of the three available architectures provided within the field of quantum computing. In contrast, the adiabatic computation provides a scheme of computation that remains unedited for any classical device.

1.3.1 Quantum Gates

Quantum logic gates, or simply quantum gates, are the most basic building blocks of all quantum circuits. They actually are a minimal quantum circuit operating on a small number of qubits. The gates correspond to operators used to manipulate the quantum state of qubits.

These gates are mathematically described by unitary matrices(unlike many classical gates (e.g. AND, OR)), and their action is always logically reversible.

By unitary we mean that it meets $U^\dagger = U^{-1}$condition, where U is the gate operation matrix and $U^\dagger$ is the adjoint of U. This property also ensures that it is a reversible gate operation [Barenco et al., 1995]. Below, we represent essential single-qubit and multi-qubit gates.

1.3.1.1 Single-Qubit Gates

- **Pauli-X Gate** ($\hat{\sigma}_x$): A rotation of π around the X-axis of the Bloch sphere, flipping $|0\rangle$ to $|1\rangle and |1\rangle to |0\rangle$. Matrix:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{1.16}$$

  Used to adjust superpositions.

- **Pauli-Y Gate** ($\hat{\sigma}_y$): A rotation of π around the Y-axis, mapping $|0\rangle \to i|1\rangle$, $|1\rangle \to -i|0\rangle$. Matrix:

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \tag{1.17}$$

  Used for phase manipulation.

- **Pauli-Z Gate** ($\hat{\sigma}_z$): A rotation of π around the Z-axis, mapping $|1\rangle \to -|1\rangle$ and keeping $|1\rangle$ on its base state . Matrix:

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{1.18}$$

  Used to alter phases.

- **Hadamard Gate** ($\hat{H}$): An isomorphism on $\mathbb{C}^2$, equivalent to a π rotation around the diagonal axis (between X and Z) of the Bloch sphere, flipping $|0\rangle \to \frac{|0\rangle+|1\rangle|}{\sqrt{2}}$ and

$|1\rangle \rightarrow \frac{|0\rangle - |1\rangle|}{\sqrt{2}}$ Matrix:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{1.19}$$

Used for equal superpositions.

- **Phase Gate** ($\hat{S}$): A rotation of $\pi/2$ around the Z-axis, mapping $|1\rangle \rightarrow i|1\rangle$. Matrix:

$$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \tag{1.20}$$

Used for phase adjustments.

1.3.1.2 Multi-Qubit Gates

- **Controlled-NOT Gate** ($C\hat{N}OT$): A two-qubit gate that flips the target qubit if the control is $|1\rangle$, creating entangled states like $\frac{|0\rangle|\otimes|0\rangle+|1\rangle\otimes|1\rangle}{\sqrt{2}}$. Matrix:

$$C\hat{N}OT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{1.21}$$

Used for entanglement.

- **Toffoli Gate** ($CC\hat{N}OT$): A three-qubit gate that flips the target qubit if both control qubits are $|1\rangle$, simulating a classical AND gate. Its matrix is an 8×8 matrix that swaps the states $|1\rangle \otimes |1\rangle \otimes |0\rangle$ and $|1\rangle \otimes |1\rangle \otimes |1\rangle$. Used for reversible classical computation.

These gates manipulate amplitudes and entanglement of quantum states, enabling the construction of quantum circuits.

1.3.2 Quantum Circuits

A circuit is a self-contained setup of quantum gates, qubits, and wires arranged to perform a specific task. Its functionality is similar to what we refer to as an algorithm.
Chaining circuits together (using the output of one as the input to the next) builds a complete quantum algorithm to solve a larger given problem Munoz Coreas and Thapliyal [2022].

Building on single- and multi-qubit superpositions and gates, we describe how gates are applied serially or in parallel to form circuits, in order to implement algorithms that exploit quantum phenomena like superposition and entanglement.

- **Serially Wired Gates**: Gates applied sequentially on the same qubit(s) combine via matrix multiplication, with the order reversed. For example, applying the Pauli-Y gate followed by the Pauli-X gate yields:

$$\hat{C} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} = -i\hat{\sigma}_z \tag{1.22}$$

  This adjusts superposition amplitudes.

- **Parallel Gates**: Gates applied simultaneously to different qubits combine via the tensor product. For example, $\hat{\sigma}_y \otimes \hat{\sigma}_x$ acts on two qubits, producing:

$$\hat{\sigma}_y \otimes \hat{\sigma}_x = \begin{pmatrix} & 0 & 0 & -i \\ 0 & 0 & -i & 0 \\ 0 & i & 0 & 0 \\ i & 0 & 0 & 0 \end{pmatrix} \tag{1.23}$$

  This enables multi-qubit operations.

- **Hadamard Transform**: Applying Hadamard gates in parallel ($\hat{H}^{\otimes n}$) creates uniform superpositions. For two qubits:

$$\hat{H}_2 = \hat{H} \otimes \hat{H} = \frac{|00\rangle + |01\rangle + |10\rangle + |11\rangle}{2} \tag{1.24}$$

  This produces equal-probability states, used in algorithms like amplitude amplification.

- **Entangled States**: To apply a gate (e.g., $\hat{H}$) to one qubit in an entangled state (e.g., $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$), extend it with the identity:

$$\hat{K} = \hat{H} \otimes \hat{I}, \hat{K}\frac{|00\rangle + |11\rangle}{\sqrt{2}} = \frac{|00\rangle \ + |01\rangle + |10\rangle| - |11\rangle}{2} \tag{1.25}$$

  This transforms entangled superpositions.

These gates and circuits manipulate amplitudes and entanglement of quantum states, enabling the construction of quantum algorithms.

1.3.3 Quantum Correlations

Correlation, a statistical concept, measures how knowledge of one system component predicts the behavior of another. In classical mechanics, correlations are deterministic: the state of a subsystem precisely determines that of another. Any observable in a classical system has only one possible outcome, with deviations attributed to measurement errors. In quantum mechanics, however, correlations exhibit a probabilistic nature. Observables in a quantum system do not possess a unique, predetermined outcome; instead, they have possible eigenvalues with associated probabilities, derived from a Hermitian matrix. As a result, even in the absence of external perturbation, repeated measurements of a quantum system can yield varying results. These outcomes are governed by eigenfunctions residing within a Hilbert space, a mathematical construct that encompasses all possible quantum states of the system.

1.3.4 Quantum Entanglement

Quantum entanglement is a phenomenon that occurs when two or more quantum systems become correlated in such a way that the state of one system cannot be described independently of the state of the other system. This phenomenon, which Albert Einstein referred to as a "spooky action at a distance", defies the classical notion of local realism and the human commonsense [Einstein et al., 1971]. Entanglement occurs when two particles become linked such that the state of one particle instantaneously influences the state of the other, regardless of the distance separating them [Cacciapuoti et al., 2020]. While entanglement itself, is nondeterministic, meaning the specific outcomes cannot be predicted with certainty, achieving deterministic control over entangled states could lead to groundbreaking applications across various fields. There are different forms of entanglement, but the most common are the Bell States, which represent maximally entangled qubits as in figure 1.5

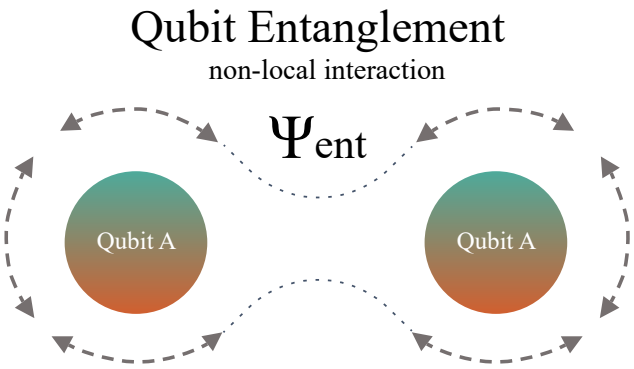

Figure 1.5: A simple visual model for the Bell states for two qubits in a non-local interaction

1.3.5 Heisenberg's Uncertainty Principle

When measuring a property of a quantum system, like the position of an electron, one can immediately think of shining a beam of light on that electron to measure its position. However, this will change the velocity of the electron, which in turn changes its momentum fundamentally as its kinetic energic increases. Thereby, the more precisely one property is measured, the less precisely the other can be determined. This inherent uncertainty in quantum mechanics leads to quantum noise, which manifests as fluctuations in the measured values of physical quantities. Quantum noise, therefore, can be partially understood through the lens of the Heisenberg uncertainty principle, which states that the product of the uncertainties in position and momentum is bounded by a constant as expressed by the famous Equation 1.26:

$$\Delta x \Delta p \geq \frac{\hbar}{2} \tag{1.26}$$

Where Δx and Δp are the uncertainties in position and momentum, respectively, and $\hbar$ is the reduced Planck's constant. This principle is a fundamental feature of quantum mechanics and has profound implications for our understanding of the nature of reality. The uncertainty principle states that it is fundamentally impossible to simultaneously measure the exact value of certain pairs of related physical quantities, such as position (x) and momentum (y), with infinite precision [Heisenberg, 1927]. It implied an intrinsic limitation on our observational ability to fully determine the state of any given system. Heisenberg's uncertainty principle provides a fundamental explanation for the presence of quantum noise, which arises from the inherent uncertainty in the measurement process. Qubits, as quantum mechanical entities, are particularly vulnerable to this noise.

Addressing these challenges is essential for advancing quantum computing and harnessing its full potential.

1.3.6 High-Dimensional Hilbert Spaces

High-dimensional Hilbert spaces $\mathcal{H}$ in quantum computing entail exponential growth of state dimensionality, leading to intractable memory and computational barriers for classical simulation and algorithm optimization. Tensor network methods truncate entanglement to a polynomially manageable subspace, mitigating this curse of dimensionality.

1.4 Processing Quantum Information with the Adiabatic Model

In the adiabatic model, the quantum system is initialized in the ground state of a simple Hamiltonian. The system is then slowly evolved to a more complex Hamiltonian whose ground state encodes the solution to the problem of interest. The adiabatic theorem guarantees that if the evolution is slow enough, the system will remain in its ground state throughout the process, leading to the desired solution with high probability.

1.4.1 Quantum Annealing

Quantum annealers are specialized quantum computers designed to find the optimal solution to a QUBO problem by measuring the ground state of the QPU, i.e., the qubit configuration corresponding to the minimum energy of the system. The basic idea is to prepare the qubits for the initial state, an easy configuration described by a Hamiltonian $\hat{H}_T$, and then let the system evolve until it becomes equal to $\hat{H}_P$, the Hamiltonian of the problem to be solved. The evolution is done by a time-dependent Hamiltonian $\hat{H}(t)$, which is a linear combination of the two Hamiltonians. If the evolution is sufficiently slow, the adiabatic theorem [Morita and Nishimori, 2008] guarantees that the system will remain in the ground state of the Hamiltonian throughout the evolution. The final state of the system will be the solution to the problem encoded in $\hat{H}_P$.

$$\hat{H}(t) = -G(t)\hat{H}_T - F(t)\hat{H}_P = -G(t)\sum_i \hat{\sigma}_i^x - F(t)(\sum_{i=1}^{N} h_i\hat{\sigma}_i^z + \sum_{i=1}^{N}\sum_{j=i+1}^{N} J_{ij}\hat{\sigma}_i^z\hat{\sigma}_j^z) \quad (1.27)$$

Where $G(t)$ and $F(t)$ are the annealing schedules that control the evolution of the system. The initial Hamiltonian $\hat{H}_T$ is a transverse field Hamiltonian, which is easy to prepare and has a known ground state. The final Hamiltonian $\hat{H}_P$ encodes the problem to be solved, and its ground state corresponds to the optimal solution.

1.4.2 Applicable Problems

Quantum annealing excels at finding low-energy states, which makes it suitable for two main classes of problems: optimization and probabilistic sampling.

Optimization Problems : These involve finding the best solution among many possible combinations, such as determining the most efficient delivery schedule or the shortest route for a traveling salesperson. Many of these problems can be reformulated as energy minimization tasks, where systems naturally move toward low-energy states. Quantum annealing applies this principle, making it a powerful tool for identifying optimal or near-optimal solutions efficiently.

Sampling Problems : In machine learning and probabilistic modeling, sampling from a range of low-energy configurations helps build models that account for uncertainty and noise in data. Quantum annealing facilitates this by exploring low-energy configurations to generate representative samples. This is particularly useful for training probabilistic models, like generating new digit images from the MNIST dataset, as it efficiently samples energy-based distributions.

1.4.3 Double Well Potential

In quantum mechanics, **a double well potential** describes a system with two stable energy minima separated by a potential barrier, allowing a particle, such as a qubit, to exist in one of two states or in a superposition of both. This landscape enables quantum tunneling, where the particle can transition between wells without exceeding the barrier, a key feature in systems like quantum annealing. In D-Wave's quantum processing units, the double well potential models qubit evolution: initially a single well (superposition of 0 and 1), it splits into two minima (classical states 0 or 1) as an energy barrier forms, with tunneling facilitating the exploration of solutions before collapsing into a low-energy state, as illustrated in Figure. 1.6 .This concept is essential for understanding quantum phenomena in computing, molecular dynamics, and condensed matter physics.

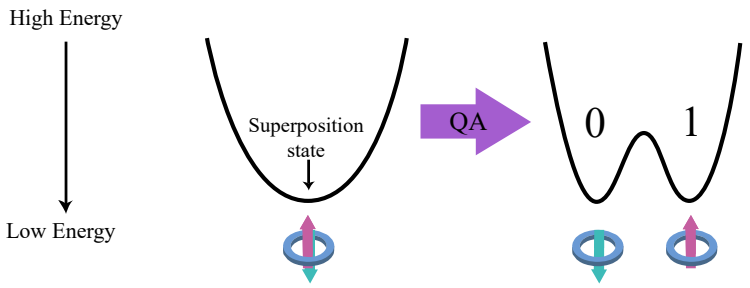

Figure 1.6: Annealing process's energy diagram showing the raising of the energy barrier for a single qubit, resulting in a 50/50 probability of ending in a classical state of 0 or 1.

1.4.4 The Hamiltonian and the Eigenspectrum

In physics, a Hamiltonian is a mathematical function that describes the energy of a system. For classical systems, it tells you the energy of any state. For instance, if you consider an apple either on a table or on the floor, gravity makes the state with the apple on the floor lower in energy.

In quantum systems, the Hamiltonian assigns energy values to specific states called *eigenstates*. Only when the system is in one of these eigenstates is its energy well-defined. The collection of these states and their energies is called the *eigenspectrum*.

quantum annealing use a time-dependent Hamiltonian to drive the annealing process. This Hamiltonian slowly evolves from a simple quantum state into a final classical state that encodes the solution:

$$H_{\text{ising}}(s) = -\frac{A(s)}{2}\sum_i \sigma_x^{(i)} + \frac{B(s)}{2}\left(\sum_i h_i\sigma_z^{(i)} + \sum_{i>j} J_{i,j}\sigma_z^{(i)}\sigma_z^{(j)}\right) \tag{1.28}$$

Here, σ_x and σ_z are Pauli matrices acting on qubit i, h_i represents a qubit's bias, and $J_{i,j}$ the coupling strength between qubits i and j

1.4.5 Annealing in Low-Energy States

The quantum annealing process can be visualized by plotting energy levels over time (Figure 1.7). The ground state (lowest energy) appears at the bottom, with excited states above it.

Initially, the system remains securely in the ground state. As annealing progresses,

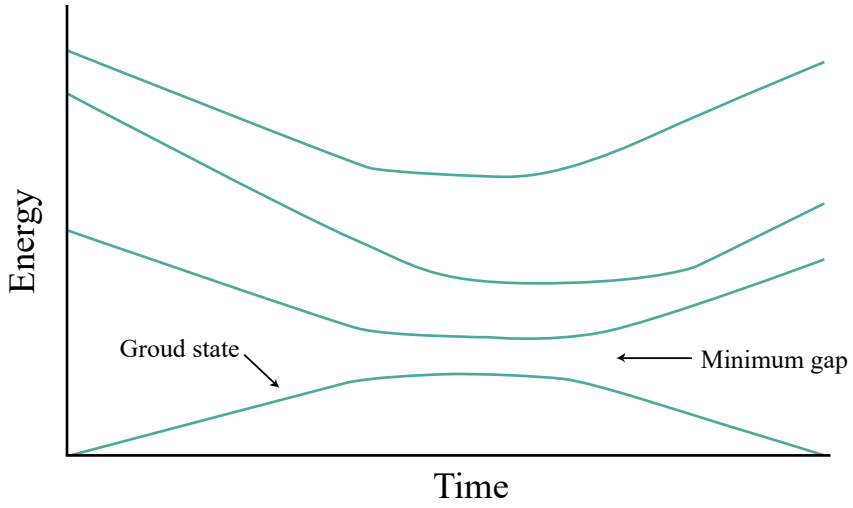

Figure 1.7: Energy spectrum showing ground state and excited states during annealing.

energy levels may approach each other, particularly the first excited state coming close to the ground state before separating again. The smallest distance between them is called the *minimum gap*.

In practice, two main factors can cause jumps to higher energy states:

- Natural thermal fluctuations
- Running the anneal too quickly

The ideal adiabatic process (slow, isolated evolution) is theoretical - real-world quantum annealing often produces useful low-energy states even when perfect ground states aren't achieved. Problem difficulty correlates with minimum gap size: smaller gaps mean harder problems.

1.4.6 Evolution of Energy States

During quantum annealing, the system evolves according to two key energy terms in the Hamiltonian: $A(s)$ (tunneling energy) and $B(s)$ (problem Hamiltonian energy), where s represents the anneal fraction ranging from 0 to 1.Figure. 1.8 illustrates how these energies vary over time. At the beginning ($s = 0$), $A(s)$ dominates, placing qubits in a delocalized quantum state. As annealing progresses ($s = t/t_f$, where t is time and t_f is total anneal time), $A(s)$ decreases while $B(s)$ increases. By the end ($s = 1$), only $B(s)$ persists, defining a classical Hamiltonian where each bitstring (qubit states of 0 or 1) corresponds to the energy of a solution.

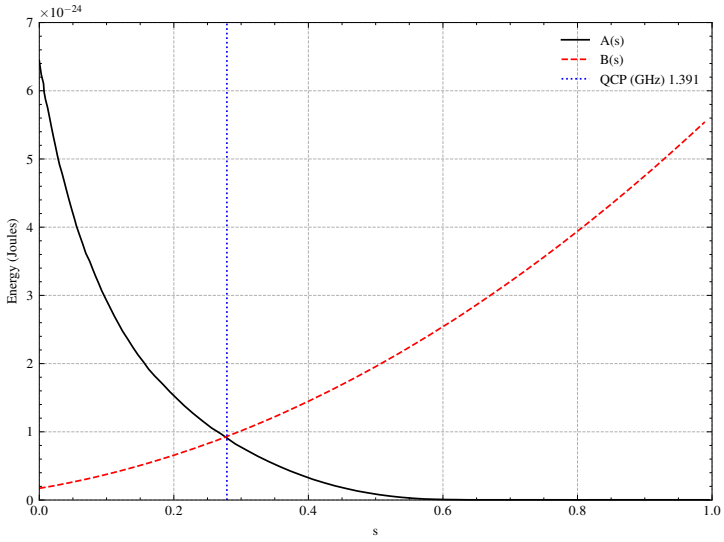

Figure 1.8: Annealing schedule showing $A(s)$ (tunneling energy) and $B(s)$ (problem energy) as functions of anneal fraction s. At $s = 0$, $A(s) \gg B(s)$, and at $s = 1$, $A(s) \ll B(s)$.

2 Quantum Machine Learning

Machine Learning (ML) is a class of advanced algorithms that perform a certain task. Given a large number of inputs and desired outputs, an ML model can be trained to make predictions on unseen data. If it is executed on a quantum processor, it becomes a quantum ML algorithm (QML). It is important to characterize different approaches based on the type of data and type of processor used to solve the problem [Aïmeur et al., 2006]. There are mainly four categories of QML algorithms:

- **CC**: refers to processing *Classical data using Classical computers* but by using quantum algorithms, such as recommendation systems algorithms.
- **CQ**: refers to processing *Classical data using Quantum computers* such as the quantum Boltzmann machine, and will be the main focus of this work.
- **QC**: refers to processing *Quantum data using Classical computers*. This is currently an area of investigation, an example of such is qubit characterization, control, and readout.
- **QQ**: refers to processing *Quantum data using Quantum computers*. It is also known as Fully Quantum Machine Learning (FQML). This is the most advanced area of research and is still in its infancy.

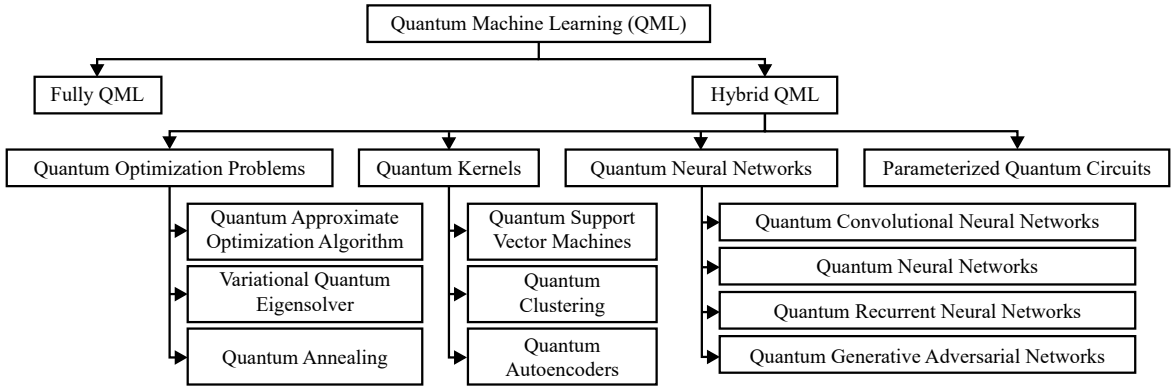

Figure 1.9: Classification of quantum machine learning algorithms.

2.1 Emulation of Quantum Computing Resources by HPC

Quantum computing promises efficient solutions to complex problems that are difficult for classical systems. While large-scale error-corrected quantum computers are not yet available, NISQ (Noisy Intermediate-Scale Quantum) devices have already demonstrated quantum supremacy on specific academic problems. However, real-world applications still require hybrid quantum-classical workflows, where the majority of computation is performed by classical hardware. Currently, NISQ devices are loosely integrated with classical systems, often involving remote quantum devices and local classical machines. This method offers flexibility and vendor independence but suffers from latency and data privacy concerns

2.2 Generalization of Neural Networks in Quantum Circuits

Machine learning, including deep learning, can be adapted to quantum hardware by encoding inputs and prompts into quantum circuits, although current hardware size limits scalability. While classical machine learning algorithms can be directly translated to quantum counterparts for performance benchmarking, neural networks require new architectures. Two primary approaches, variational and quantum perceptron, present promising paradigms for implementing neural networks on quantum systems, utilizing quantum circuits to adjust synaptic weights and process data.

2.2.1 Variational approach

Artificial Neural Networks (ANNs) map inputs to outputs via layers of weighted connections. By Hornik's theorem, a sufficiently complex ANN can approximate any function. In quantum computing, this architecture is replaced by quantum circuits composed of unitary operations $\hat{U}_i(\Phi_i)$ that act as quantum analogs of classical layers, with parameters Φ_i optimized to minimize a loss function. This framework leads to Quantum Neural Networks (QNNs), often realized as Variational Quantum Circuits (VQCs). A classical activation function takes the form:

$$f(x) = \sigma\left(\sum_j W_{ij}x_j + b_i\right) \tag{1.29}$$

where W denotes weights, b biases, and σ an activation function (e.g., sigmoid, ReLU, or Heaviside step). Despite their potential, VQCs can suffer from the *barren plateau* problem, where gradients vanish exponentially with the number of qubits. Strategies such as careful initialization and quantum convolutional architectures have been proposed to mitigate this issue

2.2.2 Neurons into Qubits

Encoding neurons into qubits for quantum neural networks (QNNs) is bounded by the qubit's definition and the maximum encoding capability of N-qubit registers.
Two primary encoding approaches exist:

2.2.2.1 1-to1 Encoding

Each and every neuron of the network is directly mapped to a single qubit. Information is stored as bit strings in classical base states or as superpositions of binary data in multi-qubit states, as a series of bit strings, as seen in Quantum Associative Memory models.[Ventura and Martinez, 2000]
The quron(quantum neuron) model(a different 1-to-1 option) further refines this by mapping a qubit's $|0\rangle$ and $|1\rangle$ states to resting and active neural firing states, respectively.

2.2.3 n-to-2^n (Superposition Coefficients) Encoding

Information is encoded in the coefficients of a superposition of quantum states, leveraging the exponential capacity of an n-qubit system. Since an n-qubit state resides in a 2^n-dimensional Hilbert space, this enables exponential compression (e.g., a megabit image can be encoded using around 20 qubits, as $2^{20} \approx 10^6$) . However, preparing such states is computationally intensive, requiring either quantum circuits to approximate target distributions or direct conversion of quantum data from sensing systems. The 1-to-1 approach is currently impractical for large-scale problems like image classification due to its linear scaling, whereas the n-to-2^n method offers exponential efficiency but faces challenges in state preparation.

2.3 Quantum Supervised Learning

Quantum formulation transposed the state-of-the-art supervised learning, a branch of machine learning. In this work, we will focus on quantum algorithms relevant to quantum Boltzmann machines, along with a description of their classical counterparts, activation functions for binary decision [Tacchino et al., 2019], kernel methods in general [Schuld and Killoran, 2019], and Support Vector Machine (SVM) [Rebentrost et al., 2014]. Classification by quantum tensor network on reduced MNIST with 4 categories has been shown to return the same performances as best supervised learning algorithms, but more intriguingly, it was able to discriminate quantum ground states carrying entanglement. In this article, we introduce a range of techniques that vary from the implementation of a natively quantum perceptron to the employment of adiabatic computation to improve the performance of the quantum Restricted Boltzmann Machine. All the aforementioned techniques perform well when dealing with sampling from probabilistic distributions.

2.3.1 Quantum FFNN for binary decisions:

A quantum feed-forward neural network for binary decisions implements a perceptron model using quantum circuits. The input vector $\vec{i} = (i_0, \ldots, i_{m-1})$ and weight vector $\vec{w} = (w_0, \ldots, w_{m-1})$, where $i_k, w_k = \pm 1$, are encoded into quantum states in a Hilbert

space $\mathcal{H}^{\otimes N}$ with $N = \log_2 m$:

$$|\psi_i\rangle = \frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} i_j |j\rangle, \quad |\psi_w\rangle = \frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} w_j |j\rangle, \tag{1.30}$$

where $|j\rangle$ are computational basis states. The dot product $\vec{i} \cdot \vec{w}$ is computed via the inner product $\langle \psi_w | \psi_i \rangle = \frac{\vec{i} \cdot \vec{w}}{m}$. A unitary operator $\tilde{U}_w$ is defined such that $\tilde{U}_w |\psi_w\rangle = |m-1\rangle$, where $|m-1\rangle = |1\rangle^{\otimes N}$. Applying $\tilde{U}_w$ to $|\psi_i\rangle$ yields $|\psi_{i,w}\rangle = \tilde{U}_w |\psi_i\rangle$. This state is entangled with an ancilla qubit $|0\rangle$ using a multi-CNOT gate controlled by $|m-1\rangle$, producing:

$$C_{|j\rangle,|m\rangle} |\psi_{i,w}\rangle |0\rangle = \frac{1}{\sqrt{m}} \left[\sum_{j=0}^{m-2} c_j |j\rangle |0\rangle + c_{m-1} |m-1\rangle |1\rangle \right]. \tag{1.31}$$

Measuring the ancilla in the $|1\rangle$ state occurs with probability $|c_{m-1}|^2 = \left(\frac{\vec{i} \cdot \vec{w}}{m}\right)^2$, enabling probabilistic activation of the perceptron. This quantum approach provides an analog to the classical perceptron for binary decisions.

2.4 Quantum Approximate Optimization Algorithm

Quantum computation has emerged as a powerful tool for addressing complex optimization challenges, such as those encountered in data clustering. Data clustering involves grouping multidimensional data into natural clusters based on similarity measures, a process critical to applications like network traffic monitoring and graph-based network security. In these scenarios, servers are often modeled as nodes and data flows as edges in a graph. Traditional clustering methods frequently rely on optimization tasks, such as minimizing a square error function, but graph-based anomaly detection can become computationally intensive, often scaling with the number of nodes and turning NP-hard when using techniques like graph similarity. In 2022, Li et al. introduced a clustering algorithm designed to monitor web traffic flow, highlighting the practical relevance of such techniques. However, the Quantum Approximate Optimization Algorithm (QAOA) offers a quantum alternative that can potentially solve these graph-based problems in polynomial time, making it a promising approach for enhancing efficiency in large-scale clustering tasks.

2.4.1 NP-Hard Problems

In computational complexity theory, a problem is classified as NP-hard if it is at least as hard as the hardest problems in NP (nondeterministic polynomial time). This means that if a solution for an NP-hard problem can be found in polynomial time, then all problems in NP can also be solved in polynomial time NP-hard problems due to the complex energy landscapes in high-dimensional Hilbert spaces(e.g., combinatorial optimization) are intractable classically, but are potential targets for:

- Quantum algorithms (e.g, QAOA) offering speedups.
- Quantum machine learning models leveraging superposition/entanglement (1.3.4, 1.2.2). In particular, quantum annealing uses quantum circuits to tackle NP-hard optimization problems by evolving quantum states in high-dimensional Hilbert spaces toward the ground state of a problem's energy function.

2.4.2 The MaxCut problem

The MaxCut problem is an NP-hard combinatorial task defined on a weighted graph G = (V,E) where V=1,...,n are vertices E their connections, the weights of the (i,j) $\in$ E connections are given by the weight matrix w_{ij}.
The goal is to partition V into subsets S and $\bar{S}$ to maximize the total weight between them. We can formulate this problem mapped into a Hamiltonian formulation inspired by the Ising model (1.3) as follows:

$$\max_{1\leq i<j\leq n} \sum \frac{w_{ij}}{2}(1 - z_i z_j) \longrightarrow \hat{H}_C = \sum_{ij} \frac{w_{ij}}{2}(\hat{I} - \hat{Z}_i \hat{Z}_j) \tag{1.32}$$

where:

- $z_i = \pm 1$ (classical spin variables)
- $\hat{Z}_i$ are Pauli-Z operators
- $\hat{I}$ is the identity operator
- The expectation value satisfies $z_i = \langle \hat{Z}_i \rangle$

The cut is set by minimizing the $E_C = \langle \widehat{H_C} \rangle$ observable. When two vertices belong to the same subset S or $\bar{S}$, then $z_i = z_j$ and the contribution to E_C is null. Thus, the set of

S and $\bar{S}$ can be thought of as a partition of the system into up and down spins.

2.4.3 QAOA Formulation

The quantum approximate optimization algorithm (QAOA) is created to resolve the Max-Cut problem.
This algorithm consists first of creating a register of n-qubits in the eigenstate of the Hamiltonian $\hat{H}_B$:

$$|\Psi(t=0)\rangle = |+\rangle^{\otimes N}, \hat{H}_B = \bigotimes_{i=1}^{N} \hat{X}_i \Rightarrow \hat{H}_B|\Psi(0)\rangle = |\Psi(0)\rangle \tag{1.33}$$

where $|+\rangle^{\otimes N}$ represents the tensor product of n qubits each in the $|+\rangle$ state and is the maximum for the Hamiltonian $\hat{H}_B$ which governs the time evolution of this state.
with $\Psi(t=0)\rangle$ is an eigenstate of $\hat{H}_B$ with the corresponding eigenvalue.
The purpose is making $\Psi(t=0)$ evolve to the maximum eigenstate for the $-\hat{H}_C$ Hamiltonian from Equation 1.32 using the adiabatic theorem 1.4, i.e. to the minimum eigen state for $\hat{H}_C$.
Thus, we introduce a time-dependent Hamiltonian $\hat{H}(t)$:

$$\hat{H}(t) = \left(1 - \frac{t}{T}\right)\hat{H}_B - \frac{t}{T}\hat{H}_C, \quad 0 \leq t \leq T, \tag{1.34}$$

Via setting the annealing schedules by the $1 - t/T$ and t/T and using the adiabatic theorem, it is possible to make $|\Psi(t=0)\rangle$ evolve from the higher energy state of $\hat{H}_B$ to the higher energy state of $-\hat{H}_C$ (and therefore to the ground state of $\hat{H}_C$) then our equation 1.34 becomes:

$$|\Psi(T)\rangle = \exp\left(-\frac{i}{\hbar}\int_0^T dt, \hat{H}(t)\right)|\Psi(0)\rangle = \exp\left(-\frac{iT}{\hbar}\int_0^1 ds, \hat{H}(s)\right)|\Psi(0)\rangle, \tag{1.35}$$

with s= $\frac{t}{T}$ This evolution is approximated with the Trotter formula ([Sun et al., 2018]) by discretizing the trajectory t into p steps of duration ($\epsilon = \frac{1}{p}$), so that the operator in the equation 1.34 is written as:

$$\exp\left(-\frac{iT}{\hbar}\int_0^1 ds, \hat{H}(s)\right) \approx \prod_{k=0}^{p-1} \exp\left(-\frac{iT}{p\hbar}\hat{H}_k\right), \tag{1.36}$$

where $\hat{H}_k = \hat{H}(k/p)$

For $p \to \infty$, the same Trotter formula splits $\hat{H}_k$ into $\hat{H}_B$ and $\hat{H}_C$ evolving the expression to:

$$\prod_{k=1}^{p} \exp\left(-\frac{iT}{p\hbar}\left(1-\frac{k}{p}\right)\hat{H}_C\right)\exp\left(-\frac{iT}{p\hbar}\frac{k}{p}\hat{H}_B\right)|\Psi(0)\rangle. \tag{1.37}$$

Thus, the approximation is more efficient as p gets bigger.

We can treat the terms in the round brackets as a set of angles γ_j, β_j, transforming the state into:

$$|\Psi(\gamma,\beta)\rangle = e^{-i\beta_p\hat{H}_B}e^{i\gamma_p\hat{H}_C}\cdots e^{-i\beta_1\hat{H}_B}e^{i\gamma_1\hat{H}_C}|+\rangle^{\otimes N}, \tag{1.38}$$

where p represents the circuit depth.

The final state is expressed as:

$$|\Psi(\gamma,\beta)\rangle = \prod_{i=1}^{p}\hat{U}_B(\beta_i)\hat{U}_C(\gamma_i)|+\rangle^{\otimes N}. \tag{1.39}$$

The cost function

$$E_p(\gamma,\beta) = \langle\Psi(\gamma,\beta)|\hat{H}_C|\Psi(\gamma,\beta)\rangle) \tag{1.40}$$

improves with increasing p. Eventually, the adiabatic process is mapped to an optimization over 2p parameters γ, β, often solved using hybrid algorithms combining classical gradient descent with quantum circuit backpropagation.

A useful metric, to assess how far the state is from the solution (i.e.the ground state of $\hat{H}_C$ is the approximation ratio r = E/E_{max}, where E_{max} is the maximum eigenvalue of $\hat{H}_C$ and $|\Psi\rangle$ is the state of the system, measures proximity to the ground state, and r $\to$ 1 indicates the exact solution.

2

Work Methodology

Introduction

Boltzmann machine (BM) is an energy-based model defined on an undirected graph and is used for unsupervised learning, serving as a basis for powerful deep learning models such as deep belief networks and deep Boltzmann machines The graph vertices are segregated into a set of visible and hidden units. The probability of each state is dependent on the total energy of the graph for that state. Furthermore, only the state of a visible unit is "visible" to the user. Thus, these visible states' marginalized probabilities are a nonlinear function of the energy parameters. These BMs can be trained on either Maximum-Likelihood (ML) learning or Contrastive Divergence (CD) learning techniques [Srivastava and Sundararaghavan, 2023]. It is well known that ML learning of Markov random fields (MRF) is a challenging task due to the large state space the parameters reside. This presents an intrinsic complexity that limits the performance of the system. Markov Chain Monte Carlo (MCMC) methods typically take a long time to converge on unbiased estimates. CD learning, on the other hand, provides a computationally inexpensive way of training MRFs. However, in general, it provides biased estimates.

Previous works [Hinton, 2002] introduced a subclass of BM called Restricted Boltzmann Machine (RBM) where the hidden and visible nodes had a bipartite structure. This allows the update of visible states by hidden states' knowledge and vice versa. This, in turn, accelerates RBM training on classical computers. Thereby, Boltzmann machines have received much attention as building blocks of multi-layer learning architectures for speech and image recognition [Jaitly and Hinton, 2011]. By stacking RBM outputs to other RBM inputs, one can construct the architecture of a Deep Boltzmann machine. In principle,

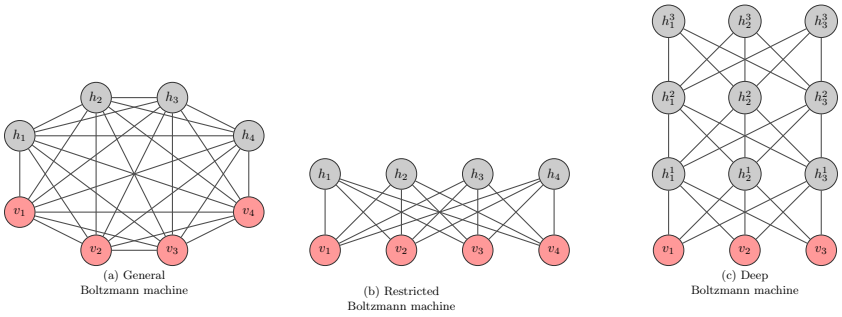

Figure 2.1: General, restricted, and deep Boltzmann machine architectures. The visible units are represented by circles, while the hidden units are represented by squares. The connections between the visible and hidden units are represented by lines.

one could consider more general energy functions to bring more flexibility [Hinton et al., 2006, Sejnowski et al., 1986] but training can become impractical, and generalization suffers as the number of parameters grows. As BMs consist of visible and hidden binary units, which we jointly denote by $(z_a)_{1<a<=N}$, where N is the total number of units, rather than traditional binary values. However, to maintain the consistency with quantum mechanics, the notation is rather utilized to be $z_a \in \{-1, +1\}$ This transformation induces the equivalence of the corresponding probability distributions with a linear bias of their parameters. To distinguish the visible and hidden variables, we follow the notation $z_a = (z_\nu, z_i)$, with ν the index for visible variables and i for hidden [Ghojogh et al., 2021] We also use vector notations $v, h,$ and $z = (v, h)$, to represent the states of visible, hidden, and combined units, respectively. In physics language, the quadratic energy function over binary units z_a is referred to as the Ising model with the energy function:

$$E_z = -\sum_a b_a z_a - \sum_{a,b} w_{ab} z_a z_b = E_{bias} + E_{coupling} = \sum_a E_{bias} + \sum_{a,b} E_{coupling} \tag{2.1}$$

where b_a is the bias of the unit a and w_{ab} is the coupling between units a and b. The first term in the energy function is the bias energy, which is a sum of the biases of each unit. The second term is the coupling energy, which is a sum of the couplings between

pairs of units. The total energy of the system is given by the sum of these two terms. It natively calculates how many neurons are *excited* together and how much they are *resting* together. Or in other words, how much they are correlated and how much of a stable state the system is in. The dimensionless parameters b_a and w_{ab} are tuned during the training [Amin et al., 2018]. In equilibrium, the probability of observing a state v of the visible parameters is given by the Boltzmann distribution summed over the hidden variables:

$$P(v) = \frac{1}{Z}\sum_h e^{-E_z}, \quad Z = \sum z = (v,h)e^{-E_z} \tag{2.2}$$

where Z is the partition function, which normalizes the probability distribution. The partition function is a sum over all possible states of the system, and it ensures that the probabilities sum to 1. P_v is called *marginal distribution*. Our goal is to determine Hamiltonian parameters, $\theta \in \{b_a, w_{ab}\}$, such that the marginal distribution of the visible units matches the training data distribution $P_v^{(data)}$. To achieve this, prior works [Ghojogh et al., 2021, Hinton, 2002] have proposed to minimize the average log-likelihood, or equivalently, minimize the average negative log-likelihood NLL defined by:

$$\mathcal{L}(\theta) = \mathbb{E}_{v\sim P_{\text{data}}}\left[\log P_\theta(v)\right] = -\frac{1}{N}\sum_v P_v^{(data)} \log \frac{\sum_h e^{-E_z}}{\sum_{z'} e^{-E_z}} \tag{2.3}$$

However, we tend to ignore the term $1/N$ in the NLL, as it is a constant factor that does not affect the optimization process. We aim to find $\hat{\theta} = arg\min_\theta \mathcal{L}(\theta)$. One commonly used approach can be done using convex optimization algorithms such as the gradient descent technique. In each iteration, the parameter θ is changed by a small step in the direction opposite to the gradient:

$$\theta^{(t+1)} = \theta^{(t)} - \eta\nabla_\theta\mathcal{L}(\theta^{(t)}) \tag{2.4}$$

where η is the learning rate, which controls the step size of the update. The gradient $\nabla_\theta\mathcal{L}(\theta^{(t)})$ is computed using backpropagation, which is a method for efficiently computing the gradient of a function with respect to its parameters. For simplicity measures, we will denote the gradient as $\partial_\theta\mathcal{L}$ instead of $\nabla_\theta\mathcal{L}$. This technique is widely used in machine learning [Amari, 1993, Haji and Abdulazeez, 2021, Sutton et al., 2009] for its ability to

calculate gradients efficiently. Using 2.3, we have:

$$\partial_\theta \mathcal{L} = \sum_v P_v^{data} \frac{\sum_h \partial_\theta E_z e^{-E_z}}{\sum_h e^{-E_z}} - \frac{\sum_z \partial_\theta E_z e^{-E_z}}{\sum_z e^{-E_z}} = \overline{\langle \partial_\theta E_z \rangle_v} - \langle \partial_\theta E_z \rangle \tag{2.5}$$

Where $\overline{\langle \cdots \rangle}$ and $\langle \cdots \rangle_v$ are Boltzmann averages with free and fixed variables, respectively, and $\overline{\langle \cdots \rangle} = \sum_v \langle \cdots \rangle_v$ denotes double averaging. Fixing visible variables to the data is usually called *clamping*. Hence, we obtain:

$$\delta b_a = \nu(\overline{\langle z_a \rangle_v} - \langle z_a \rangle), \quad \delta w_{ab} = \nu(\overline{\langle z_a z_b \rangle_v} - \langle z_a z_b \rangle) \tag{2.6}$$

The gradient steps are expressed in terms of differences between the clamped (i.e., fixed v) and unclamped averages. These two terms are sometimes called the *positive phase* and *negative phase*, respectively. The positive phase is the average of the clamped variables, while the negative phase is the average of the unclamped variables. The difference between these two averages is used to update the parameters of the model.

1 Restricted Boltzmann machines and Deep Belief Network

The Ising model, describing interacting spins, was introduced in 1925 [Ising, 1925a] and found in 1974 to be able to be a neural network [Little, 1974a]. Hence, Hopfield network was proposed in 1982, which modeled an Ising model in a network for modeling memory. Boltzmann Machines (BMs) and Restricted Boltzmann Machines (RBMs), energy-based models, emerged inspired by Hopfield's associative memory systems [Hopfield, 1982].
Unlike classical neural networks trained via backpropagation, BMs leverage the Boltzmann-Gibbs [Gibbs, 1902] distribution to probabilistically model data, with parameters learned through maximum likelihood estimation.
RBMs, a simplified variant restricting intra-layer connections, became widely used in deep learning through Deep Belief Networks (DBNs), which simulated a stack of RBM models with a greedy algorithm for training and allowed the networks to become deep by preparing a good initialization of weights (using RBM training) for backpropagation.

1.1 Probabilistic graphical model and Markov random field

A Probabilistic Graphical Model (PGM) is a graph-based representation of complex probability distributions in high-dimensional spaces.
We can think of it as a combination of graph theory and probability theory, so that the random variables are the nodes or variables, and the edges are seen as probabilistic interactions between them.
Different conditional probabilities can be represented by a PGM. There exist two types of PGM which are Markov networks (or Markov Random Field)(undirected edges), MRFs, and Bayesian networks (direct edges), BNs [Koller and Friedman, 2009].
With direct edges, we mean the interactions are asymmetric, which often implies causality or conditional dependency. While undirected edges, we mean that the interactions have no direction, are symmetric, and mutual.
Indeed, BMs and RBMs, fully connected networks, are MRFs for their links are undirected. Their undirected structure reflects energy-based interactions between visible and hidden units, mirroring physical systems like the Ising model, and this symmetry actually simplifies inference and training via Gibbs sampling.

1.2 Gibbs Sampling

Gibbs sampling [Geman and Geman, 1984]is an iterative Markov Chain Monte Carlo (MCMC) method used to approximate complex, high-dimensional (n-dimensional), multivariate distributions. It is particularly effective when working with models like MRFs, where the joint distribution is difficult to sample from directly. This sampling algorithm assumes that the conditional distributions of every dimension of data, conditioned on the rest of the coordinates, are simple to draw samples from. At each iteration, Gibbs sampling updates each dimension by sampling from its conditional distribution, given the current values of all other dimensions. This is done for all the n dimensions, specifically, for a distribution $P(x_1, \ldots, x_n)$, the update for each variable x_i at time $t+1$ is performed as:

$$x_i^{(t+1)} \sim P(x_i \mid x_1^{(t+1)}, \ldots, x_{i-1}^{(t+1)}, x_{i+1}^{(t)}, \ldots, x_n^{(t)}). \tag{2.7}$$

By iterating this process, the algorithm generates a series of samples that asymptotically approximate the target distribution.
However, we should mention that we accept all samples after some burn-in iterations,

assuming some initial samples are not yet valid because the algorithm has started from a not-necessarily valid vector.

Gibbs sampling is used in BM and RBM for generating visible and hidden samples, as in its implementation, it leverages the conditional independence of units within layers, enabling efficient sampling despite the high dimensionality of the joint distribution.
In RBMs, thus, alternating updates of all visible units given hidden units, and all hidden units given visible units, avoids sequential updates of individual units, significantly reducing computational cost compared to fully connected BMs. we first start with a data vector$v^{(0)}$ (a random value after burn-in). Sampling hidden units is implemented as follows:

$$h^{(t)} \sim P(h \mid v^{(t)}), \tag{2.8}$$

and sampling visible units is implemented as follows:

$$v^{(t+1)} \sim P(v \mid h^{(t)}), \tag{2.9}$$

where each $h_j^{(t)}$ and $v_i^{(t+1)}$ is sampled independently, and we iterate over the T Gibbs steps.

This alternation, termed block Gibbs sampling, avoids sequential updates of individual units, significantly reducing computational cost compared to fully connected BMs. And by leveraging quantum effects, we seek to accelerate convergence and improve sample quality in high-dimensional Hilbert spaces.

1.3 Statistical Physics and Ising Model

In statistical physics, a system of particles contains several particles $x_{i i=1}^{d}$. These particles can be seen as random variables that can randomly have a state. For example, if the particles are electrons, they can have states $+1$ and -1, which represent the counterclockwise and clockwise spins of the electrons, respectively. The Boltzmann-Gibbs distribution can show the probability that a physical system can have a specific state. i.e., all the particles have specific states. This means that the probability function of this distribution can be described as:

$$P(x) = \frac{1}{Z} e^{-\beta E(x)} \tag{2.10}$$

Where $E(x)$ is the energy of a variable x, and Z is the normalization constant so that the probabilities sum to one. This normalization constant is called the partition function, which is hard to compute as it sums over all possible configurations of states(values) that the particles can have. This would be explained in detail in section 1.3.1.

1.3.1 Boltzmann-Gibbs Distribution

The Boltzmann [Boltzmann, 1868] or the Boltzmann-Gibbs [Gibbs, 1902] distribution is an energy-based distribution that describes the probability of a physical system occupying a state s with energy E(s) at thermal equilibrium. For classical machines, this is expressed as:

$$P(s) = \frac{1}{Z} e^{-\beta E(s)} \tag{2.11}$$

where $\beta = \frac{1}{k_B T}$ (with k_B as Boltzmann's constant and T as temperature) quantifies thermal fluctuations, and $Z = \sum_s \exp -\beta E(s)$ is the partition function of the distribution. With: the free energy is defined as

$$F(\beta) := -\frac{1}{\beta} \ln(Z) \tag{2.12}$$

Later, in learning phases, these will be adopted as the parameters of the model, which enables creating the energy-based framework: a Boltzmann Machine.

The internal energy is defined as

$$U(\beta) = \sum_{x \in \mathbb{R}^d} E(x) \frac{e^{-\beta E(x)}}{Z} = \sum_{x \in \mathbb{R}^d} P(x) E(x) \tag{2.13}$$

The entropy is defined as

$$\mathcal{H}(\beta) := -\sum_{x \in \mathbb{R}^d} P(x) \ln P(x) = \beta \sum_{x \in \mathbb{R}^d} P(x) E(x) + \ln(Z) = -\beta F(\beta) + \beta U(\beta) \tag{2.14}$$

Further, RBMs defined the joint probability of visible (v) and hidden (h) units through the energy function:

$$E(v, h) = -v^T W h - b^T v - c^T h \tag{2.15}$$

where W represents interaction weights, and b,c are biases. Training relies on adjusting

these parameters to maximize the likelihood of observed data, typically via Gibbs sampling. The inverse temperature $\beta(\approx 1/T)$ controls the distribution's sharpness, influencing sampling behavior.

Furthermore, Quantum Boltzmann Machines (QBMs) extended this framework and used annealing by replacing the classical energy E(s) with a time-dependent quantum Hamiltonian during annealing:

$$\hat{H}(s) = -A(s)\sum_a \sigma_x^a + B(s)\left[\sum_a h_a\sigma_z^a + \sum_{a<b} J_{ab}\sigma_z^a\sigma_z^b\right] \tag{2.16}$$

where σ_x^a and σ_x^z are Pauli matrices, J_{ab}, h_a are tunable parameters, and A(s), B(s) are the annealing schedules with $s = t/t_a$, based on the QB distribution:

$$\rho(s) = Z^{-1}e^{-\beta\hat{H}(s)} \tag{2.17}$$

1.4 Hopfield Network

The Hopfield network initially proposed by [Little, 1974a] and later extended by Hopfield [Hopfield, 1982] serves as an associative memory system, designed to store and retrieve patterns through energy minimization. This network consists of binary units or neurons x_i, $i = 1, \ldots, d$, with states $x_i \in \{-1, +1\}$ interconnected via symmetric weights w_{ij}, where $w_{ij} = w_{ji}$ and $w_{ii}=0$.
These weights w_{ij} connecting units i and j are learned using Hebbian law of association [Hebb, 1949], defined as:

$$w_{ij} := \begin{cases} x_i x_j & \text{if } i \neq j, \\ 0 & \text{otherwise,} \end{cases} \tag{2.18}$$

where x_i and x_j are the states of the units during training.
For binary states $x_i \in \{0, 1\}$, the Hebbian rule is adjested to $w_{ij} = (2x_i - 1)(2x_j - 1)$ for $i \neq j$.
After training, the state of each unit is updated if the weighted summation of inputs to

the unit passes a threshold θ:

$$x_i := \begin{cases} +1 & \text{if } \sum_{j=1}^{d} w_{ij} x_j \geq \theta, \\ -1 & \text{otherwise,} \end{cases} \tag{2.19}$$

enabling the network to retrieve stored patterns by minimizing an energy function akin to the Ising models, given by $E(z) = -\sum_a b_a z_a - \sum_{a<b} w_{ab} z_a z_b$ [Hinton and Sejnowski, 1983a].

The BM and RBM models are Hopfield networks (as both are energy-based models inspired by the Ising model) whose weights are learned using maximum likelihood estimation rather than Hebbian learning. However, as the Hopfield networks' convergence to low-energy states for pattern retrieval in high-dimensional spaces shows computational limits, we aim to enhance it by mapping the Ising-like Hamiltonian to a quantum system, improving efficiency in high-dimensional Hilbert spaces, compared to classical methods.

1.5 Structure of Restricted Boltzmann Machine

The Restricted Boltzmann Machine (RBM) is a generative model and a Probabilistic Graphical Model (PGM) that serves as a building block for many probabilistic models. It employs the Boltzmann distribution, from which it derives its name. An RBM comprises a visible layer $\boldsymbol{v} = [v_1, \ldots, v_d] \in \mathbb{R}^d$, representing observable data, and a hidden layer $\boldsymbol{h} = [h_1, \ldots, h_p] \in \mathbb{R}^p$, capturing latent features or embeddings of the visible data. Unlike the general Boltzmann Machine (BM), which permits intra-layer connections, the RBM restricts connections such that there are no links between units within the same layer (i.e., $\boldsymbol{L} = \boldsymbol{J} = \boldsymbol{0}$). This restriction enhances computational tractability.

The energy function of the RBM is defined as:

$$E(\boldsymbol{v}, \boldsymbol{h}) := -\boldsymbol{b}^\top \boldsymbol{v} - \boldsymbol{c}^\top \boldsymbol{h} - \boldsymbol{v}^\top \boldsymbol{W} \boldsymbol{h}, \tag{2.20}$$

where $\boldsymbol{W} = [w_{ij}] \in \mathbb{R}^{d \times p}$ denotes the weight matrix between visible and hidden units, $\boldsymbol{b} = [b_1, \ldots, b_d] \in \mathbb{R}^d$ and $\boldsymbol{c} = [c_1, \ldots, c_p] \in \mathbb{R}^p$ are bias vectors for the visible and hidden layers, respectively.

The joint probability distribution over $\boldsymbol{v}$ and $\boldsymbol{h}$ follows the Boltzmann distribution:

$$\mathbb{P}(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z} \exp(-E(\boldsymbol{v}, \boldsymbol{h})), \tag{2.21}$$

where Z is the partition function:

$$Z := \sum_{\boldsymbol{v} \in \mathbb{R}^d} \sum_{\boldsymbol{h} \in \mathbb{R}^p} \exp(-E(\boldsymbol{v}, \boldsymbol{h})). \tag{2.22}$$

1.6 Conditional Distributions

Due to the absence of intra-layer connections, the RBM exhibits conditional independence: hidden units are independent given the visible units, and vice versa. This property facilitates efficient inference and sampling.

The conditional probability of the hidden units given the visible units is:

$$\mathbb{P}(\boldsymbol{h} \mid \boldsymbol{v}) = \prod_{j=1}^{p} \mathbb{P}(h_j \mid \boldsymbol{v}), \tag{2.23}$$

and for the visible units given the hidden units:

$$\mathbb{P}(\boldsymbol{v} \mid \boldsymbol{h}) = \prod_{i=1}^{d} \mathbb{P}(v_i \mid \boldsymbol{h}). \tag{2.24}$$

For binary units ($v_i, h_j \in \{0, 1\}$), these probabilities are:

$$\mathbb{P}(h_j = 1 \mid \boldsymbol{v}) = \sigma(c_j + \boldsymbol{v}^\top \boldsymbol{W}_{:j}), \tag{2.25}$$

$$\mathbb{P}(v_i = 1 \mid \boldsymbol{h}) = \sigma(b_i + \boldsymbol{W}_{i:} \boldsymbol{h}), \tag{2.26}$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$ is the sigmoid function, $\boldsymbol{W}_{:j}$ is the j-th column of $\boldsymbol{W}$, and $\boldsymbol{W}_{i:}$ is the i-th row.

1.7 Sampling Hidden and Visible Variables

Gibbs sampling is employed to sample from the joint distribution. The iterative process is:

$$\boldsymbol{h}^{(\nu)} \sim \mathbb{P}(\boldsymbol{h} \mid \boldsymbol{v}^{(\nu)}), \tag{2.27}$$

$$\boldsymbol{v}^{(\nu+1)} \sim \mathbb{P}(\boldsymbol{v} \mid \boldsymbol{h}^{(\nu)}), \tag{2.28}$$

continuing until convergence, often achieved in a few iterations due to the conditional independence.

1.8 Training Restricted Boltzmann Machine by Maximum Likelihood Estimation

The RBM parameters $\theta = \{\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{c}\}$ are trained via Maximum Likelihood Estimation (MLE) on a dataset $\{\boldsymbol{v}_i\}_{i=1}^n$. The log-likelihood is:

$$\ell(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{c}) = \sum_{i=1}^{n} \log \mathbb{P}(\boldsymbol{v}_i), \tag{2.29}$$

where $\mathbb{P}(\boldsymbol{v}_i) = \sum_{\boldsymbol{h}} \mathbb{P}(\boldsymbol{v}_i, \boldsymbol{h})$.

The gradient of the log-likelihood is:

$$\nabla_\theta \ell(\theta) = \sum_{i=1}^{n} \mathbb{E}_{\mathbb{P}(\boldsymbol{h}|\boldsymbol{v}_i)} \left[\nabla_\theta(-E(\boldsymbol{v}_i, \boldsymbol{h}))\right] - n\mathbb{E}_{\mathbb{P}(\boldsymbol{v},\boldsymbol{h})} \left[\nabla_\theta(-E(\boldsymbol{v}, \boldsymbol{h}))\right]. \tag{2.30}$$

Specific gradients are:

$$\nabla_{\boldsymbol{W}} \ell(\theta) = \sum_{i=1}^{n} \boldsymbol{v}_i \mathbb{E}_{\mathbb{P}(\boldsymbol{h}|\boldsymbol{v}_i)}[\boldsymbol{h}^\top] - n\mathbb{E}_{\mathbb{P}(\boldsymbol{v},\boldsymbol{h})}[\boldsymbol{v}\boldsymbol{h}^\top], \tag{2.31}$$

$$\nabla_{\boldsymbol{b}} \ell(\theta) = \sum_{i=1}^{n} \boldsymbol{v}_i - n\mathbb{E}_{\mathbb{P}(\boldsymbol{v},\boldsymbol{h})}[\boldsymbol{v}], \tag{2.32}$$

$$\nabla_{\boldsymbol{c}} \ell(\theta) = \sum_{i=1}^{n} \mathbb{E}_{\mathbb{P}(\boldsymbol{h}|\boldsymbol{v}_i)}[\boldsymbol{h}] - n\mathbb{E}_{\mathbb{P}(\boldsymbol{v},\boldsymbol{h})}[\boldsymbol{h}], \tag{2.33}$$

where $\widehat{\boldsymbol{h}}_i = \mathbb{E}_{\mathbb{P}(\boldsymbol{h}|\boldsymbol{v}_i)}[\boldsymbol{h}]$.

1.9 Contrastive Divergence

Exact computation of the gradients is intractable due to the partition function. Contrastive Divergence (CD) approximates these by sampling from a short Gibbs chain initialized at the data:

$$\nabla_{\boldsymbol{W}} \ell(\theta) \approx \sum_{i=1}^{n} \boldsymbol{v}_i \widehat{\boldsymbol{h}}_i^{\top} - \sum_{i=1}^{n} \tilde{\boldsymbol{v}}_i \tilde{\boldsymbol{h}}_i^{\top}, \tag{2.34}$$

$$\nabla_{\boldsymbol{b}} \ell(\theta) \approx \sum_{i=1}^{n} \boldsymbol{v}_i - \sum_{i=1}^{n} \tilde{\boldsymbol{v}}_i, \tag{2.35}$$

$$\nabla_{\boldsymbol{c}} \ell(\theta) \approx \sum_{i=1}^{n} \widehat{\boldsymbol{h}}_i - \sum_{i=1}^{n} \tilde{\boldsymbol{h}}_i, \tag{2.36}$$

where $\tilde{\boldsymbol{v}}_i$ and $\tilde{\boldsymbol{h}}_i$ are Gibbs-sampled reconstructions.

1.10 Conditional Restricted Boltzmann Machine

The Conditional RBM (CRBM) extends the RBM for time-series data by incorporating temporal dependencies via directed links from past visible units (over τ time steps) to current visible and hidden units, denoted by $\boldsymbol{G}^{(t-\tau)} \in \mathbb{R}^{d \times d}$ and $\boldsymbol{Q}^{(t-\tau)} \in \mathbb{R}^{d \times p}$, respectively. The gradients for these additional links are:

$$\nabla_{G_{i:}^{(t-\tau)}} \ell(\theta) = v_i^{(t-\tau)} \left(\sum_{k=1}^{n} v_k^{(t)} - \sum_{k=1}^{n} \tilde{v}_k^{(t)} \right), \tag{2.37}$$

$$\nabla_{Q_{i:}^{(t-\tau)}} \ell(\theta) = v_i^{(t-\tau)} \left(\sum_{k=1}^{n} \widehat{h}_k - \sum_{k=1}^{n} \tilde{h}_k \right), \tag{2.38}$$

integrated into the training alongside the standard RBM parameters.

1.11 Deep Belief Network

Before the advent of ReLU [Glorot et al., 2011] and dropout [Srivastava et al., 2014], deep networks struggled with random initial weights unsuitable for backpropagation optimization. The Deep Belief Network (DBN) [Hinton et al., 20 06],multilayer neural network, was designed to overcome the vanishing gradient problem by unsupervised pre-training the network layer-wise using a stack of RBMs, followed by fine-tuning with backpropagation [Rumelhart et al., 1986a].

A DBN, mainly, consists of multiple layers. Each pair of successive layers is treated as one RBM as shown in the figure 2.2 with the first layer receiving input data $x_i \in \mathbb{R}^d$, $i = 1, \ldots, n$, and subsequent layers containing p_l neurons, where $p_1 = d$ by convention. we introduce training dataset $x_i \in \mathbb{R}^d, \quad i = 1, \ldots, n$ as the visible variable $v_i, \quad i = 1, \ldots, n$ of the first pair of layers.
Weights W_l and biases b_l of this layer are trained using RBM methods (e.g., contrastive divergence 1.9), and later, n hidden variables h_i are generated via Gibbs sampling (1.2) These hidden variables serve as visible inputs for the next RBM pair, and the process repeats greedily layer by layer [Bengio et al., 2007].

After pre-training, the initialized weights and biases are fine-tuned with backpropagation. This approach, explained by [Hinton and Salakhutdinov, 2006, Hinton et al., 20 06] and was used for dimensionality reduction. As the RBM pretraining provides a robust starting point, increasing l to any large number, we can make the network as deep as we want without being worried about vanishing gradients because weights are initialized well for backpropagation. And this is why this network is referred to as the Deep Belief Network, as it can get deep and is pretrained by belief propagation. It is worth mentioning that pre-training of DBN is an unsupervised task because RBM training is unsupervised. Fine-tuning of DBN can be either unsupervised or supervised, depending on the loss function for backpropagation.

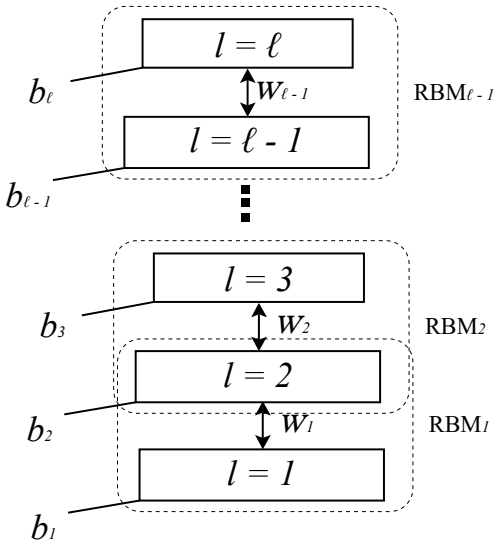

Figure 2.2: Pre-training a deep belief network by considering every pair of layers as an RBM.

2 Quantum Approach in High-Dimensional Hilbert Spaces

2.1 Quantum Boltzmann machine

In this section, we replace the classical spins or bits in the equation of Ising energy 2.1 with quantum bits (qubits). The mathematics of quantum mechanics is used to describe the quantum states of the system, where it is based on matrices (operators) with dimensionality equal to the number of possible states 2^N. Contrasively to vectors with dimensionality equal to the number of variables (N) used in common machine learning techniques. For instance, instead of the energy function 2.1, one considers a $2^N \times 2^N$ diagonal matrix $\hat{H}$ which is the Hamiltonian of the system. The Hamiltonian is a matrix that describes the energy of the system and its evolution over time. The Hamiltonian is used to calculate the energy of the system and to determine how the system evolves over time.

$$\hat{H} = -\sum_a b_a \sigma_z^a - \sum_{a,b} w_{ab} \sigma_z^a \sigma_z^b \tag{2.39}$$

This diagonal matrix is constructed in a way that its diagonal elements are energy values corresponding to all the 2^N binary states z ordered lexicographically. To generate such a Hamiltonian, we replace z_a in 2.1 with a $2^N \times 2^N$ matrix

$$\sigma_a^z \equiv \overbrace{I \otimes \ldots \otimes I}^{a-1} \otimes \sigma_z \otimes \overbrace{I \otimes \ldots \otimes I}^{N-a} = \sigma_z^a \otimes I^{\otimes(N-1)} \tag{2.40}$$

where $\otimes$ is the tensor/Kronecker outer product, σ_z, which is the Pauli Z operator that represents the spin of a qubit in the z-direction.

$$\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{2.41}$$

Every element in σ_z^a is an identity matrix (I) except the a^{th} element, which is σ_z. Thus, we replace 2.1 by the diagonal Hamiltonian where b_a, w_{ab} remain scalars. We represent the eigenstates of this Hamiltonian by $|v, h\rangle$, where again v and h denote visible and hidden variables, respectively. We can now define the matrix exponentiation through Taylor expansion, $e^{-\hat{H}} = \sum_{n=0}^{\infty} \frac{(-\hat{H})^n}{n!}$. In our case, the Hamiltonian is diagonal, and the

exponentiation $e^{-\hat{H}}$ is simply the diagonal matrix with elements e^{-E_z}, where E_z is the energy of the state z. The partition function is then given by $Z = \sum_z e^{-E_z}$, which is the sum of the exponentials. For a better notation, we use $Z = Tr[e^{-H}]$, we define the density matrix as:

$$\rho = \frac{e^{-\hat{H}}}{Z} \tag{2.42}$$

Hence, the diagonal elements of ρ are Boltzmann probabilities of all possible states. For a given state $|v\rangle$ of the visible variables, we can obtain the marginal Boltzmann probability P_v by tracing over the hidden variables:

$$P_v = Tr_h[\Lambda_v \rho] \tag{2.43}$$

where Λ_v limits the trace only to diagonal terms that correspond to the visible variables being in state v. Thus, Λ_v is a diagonal matrix with elements 1 for the states that correspond to v and 0 otherwise. The trace operation sums over all possible states of the hidden variables, effectively marginalizing them out. This results in a reduced density matrix that describes the visible variables alone, allowing us to compute their probabilities. In operator notation, we write:

$$\Lambda_v = |v\rangle\langle v| \otimes \mathcal{I}_h = \prod_\nu (\frac{1 + v_\nu \sigma_\nu^z}{2}) \otimes \mathcal{I}_h \tag{2.44}$$

Where $\mathcal{I}_h$ is the identity operator in the Hilbert space of the hidden variables. $|v\rangle\langle v| \equiv \prod_\nu (\frac{1+v_\nu \sigma_\nu^z}{2})$ is a projection operator in the subspace of visible variables. This operator projects the state of the system onto the subspace defined by the visible variables, effectively filtering out the contributions from the hidden variables. The resulting density matrix ρ_v describes the state of the visible variables alone, allowing us to compute their probabilities and perform further analysis on them. We can now add a transverse field to the Ising Hamiltonian by introducing non-diagonal matrices that represent transverse components of spin. The transverse Ising Hamiltonian is written as

$$\hat{H} = -\sum_a \Gamma_a \sigma_a^x - \sum_a b_a \sigma_a^z - \sum_{a,b} w_{ab} \sigma_a^z \sigma_b^z \tag{2.45}$$

where Γ_a is the transverse field strength of the a^{th} qubit and σ_a^x is the Pauli X operator which is defined as:

$$\sigma_a^x = \overbrace{I \otimes \ldots \otimes I}^{a-1} \otimes \sigma_x \otimes \overbrace{I \otimes \ldots \otimes I}^{N-a}, \quad \sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{2.46}$$

Every eigenstate of H is now a superposition in the computational basis made of the classical states $|v, h\rangle$. As the probabilistic model for QBM, we use the Boltzmann distribution density matrix, which now has off-diagonal elements. In each measurement, the states of the qubits are read out in the σ_z-basis, and the outcome will be a classical value ± 1. Because of the statistical nature of quantum mechanics, after each measurement, a classical output v will appear for the visible variables with the probability P_v given by the diagonal elements of the density matrix ρ.

To train the QBM, we change the parameters θ such that the probability distributions P_v become as close as possible to P_v^{data}. This is achieved by minimizing the negative log-likelihood (NLL), which becomes:

$$\mathcal{L}_{NLL}(\theta) = -\sum_v P_v^{(data)} \log \frac{Tr[\Lambda_v e^{-H}]}{Tr[e^{-H}]} \tag{2.47}$$

The gradient of $\mathcal{L}_{NLL}$ with respect to the parameters θ is given by:

$$\partial_\theta \mathcal{L}_{NLL} = \sum_v P_v^{(data)} \frac{Tr[\Lambda_v \partial_\theta e^{-H}]}{Tr[e^{-H}]} - \frac{Tr[\partial_\theta e^{-H}]}{Tr[e^{-H}]} \tag{2.48}$$

In classical physics, gradient estimation can be efficiently performed through sampling techniques. However, in quantum mechanics, the non-commutativity of operators such as $\partial_\theta H$ (the derivative of the Hamiltonian with respect to a parameter θ) and $\partial_\theta - H$ precludes the direct application of conventional expectation value computations, necessitating alternative methodologies.

To circumvent this issue, the concept of imaginary time is introduced, wherein time is treated as an imaginary quantity. The exponential term e^{-H}, which governs the statistical distribution, is expressed as $[e^{-\delta\tau H}]^n$, with $\delta\tau = 1/n$. This decomposition yields:

$$\partial_\theta e^{-H} = \sum_{m=1}^{n} e^{-m\delta\tau H}(-\partial_\theta H)\delta\tau\, e^{-(n-m)\delta\tau H} \tag{2.49}$$

In the limit as $n \to \infty$, this summation transitions into an integral, resulting in Equation

(20):

$$\partial_\theta e^{-H} = -\int_0^1 d\tau\, e^{-\tau H} \partial_\theta H e^{-(1-\tau)H} \tag{2.50}$$

The trace operation, which exhibits cyclic invariance, facilitates the simplification of these expressions. Applying the trace to Equation (20) produces Equation (21):

$$\mathrm{Tr}[\partial_\theta e^{-H}] = -\mathrm{Tr}[\partial_\theta H e^{-H}] \tag{2.51}$$

This relation is subsequently incorporated into Equation [18], yielding Equation (22):

$$\frac{\mathrm{Tr}[\partial_\theta e^{-H}]}{\mathrm{Tr}[e^{-H}]} = -\langle \partial_\theta H \rangle \tag{2.52}$$

where $\langle \cdot \rangle$ represents the expectation value under a Boltzmann distribution, mirroring classical statistical mechanics.

A significant challenge emerges in the evaluation of the first term in Equation [18]. For an observable $\mathcal{A}$, Equation is derived:

$$\frac{\mathrm{Tr}[\mathcal{A}\partial_\theta e^{-H}]}{\mathrm{Tr}[\mathcal{A}e^{-H}]} = -\int_0^1 dt \frac{\mathrm{Tr}[\mathcal{A}e^{-tH}\partial_\theta H e^{-(1-t)H}]}{\mathrm{Tr}[\mathcal{A}e^{-H}]} \tag{2.53}$$

This expression cannot be efficiently estimated via sampling, presenting a scalability barrier for quantum machine learning applications, such as the training of QBMs.

To address this limitation, an alternative strategy is proposed: rather than directly minimizing the exact loss function, an upper bound on the loss is minimized, a method referred to as bound-based QBM (bQBM). This approach, commonly employed in machine learning, renders the optimization problem more computationally feasible.

2.1.1 Bound-Based QBM

The Golden-Thompson inequality Forrester and Thompson [2014] provides a lower bound for the probability as follows:

$$\mathrm{Tr}[e^{A+B}] \geq \mathrm{Tr}[e^A e^B], \tag{2.54}$$

which is valid for any Hermitian matrices A and B. Using this, we can express the probability P_V as:

$$P_V = \frac{\mathrm{Tr}[e^{-H_v + \ln \Delta_V}]}{\mathrm{Tr}[e^{-H}]} \geq \frac{\mathrm{Tr}[e^{-H_v} e^{\ln \Delta_V}]}{\mathrm{Tr}[e^{-H}]}. \tag{2.55}$$

We introduce a new Hamiltonian:

$$H_v = H - \ln \Delta_V, \tag{2.56}$$

allowing us to rewrite P_V as:

$$P_V \geq \frac{\mathrm{Tr}[e^{-H_v}]}{\mathrm{Tr}[e^{-H}]}. \tag{2.57}$$

Note that H_v imposes an infinite energy penalty on any state of the visible qubits that differs from v. Thus, for any practical purposes:

$$H_v = H(\sigma_x^z = 0, \sigma^z = v). \tag{2.58}$$

This represents a clamped Hamiltonian because every visible qubit σ_i^z is fixed to its corresponding classical data value v_i.

From Equations (4) and (6), we derive:

$$\mathcal{L} \leq \hat{\mathcal{L}} = -\sum_v P_v^{\mathrm{data}} \log \frac{\mathrm{Tr}[e^{-H_v}]}{\mathrm{Tr}[e^{-H}]}. \tag{2.59}$$

Instead of directly minimizing $\mathcal{L}$, we minimize its upper bound $\hat{\mathcal{L}}$ using the gradient:

$$\theta\mathcal{L} = \sum_v P_v^{\mathrm{data}} \left(\frac{\mathrm{Tr}[e^{-H_v} \theta H_v]}{\mathrm{Tr}[e^{-H_v}]} - \frac{\mathrm{Tr}[e^{-H} \theta H]}{\mathrm{Tr}[e^{-H}]} \right), \tag{2.60}$$

where:

$$\langle \cdot \rangle_v = \frac{\sum_v P_{v\cdot}^{\mathrm{data}}}{\sum_v P_{v\cdot}^{\mathrm{data}}}, \quad \sum_v P_{v\cdot}^{\mathrm{data}} \frac{\mathrm{Tr}[e^{-H_v \cdot}]}{\mathrm{Tr}[e^{-H \cdot}]}. \tag{2.61}$$

The gradient steps are expressed in terms of the difference between the unclamped and clamped averages, $\langle \cdot \rangle$ and $\langle \cdot \rangle_v$, which can be obtained by sampling from the Boltzmann distribution with Hamiltonians H and H_v, respectively.

$$\theta\Gamma_v = \eta(\langle \sigma^z \rangle_v^- - \langle \sigma^z \rangle^-). \tag{2.62}$$

For certain problems, one might calculate $\langle \sigma^z \rangle^-$ by sampling. However, the first term in Equation (12) is always zero for visible variables, i.e., $\langle \sigma_i^z \rangle_v^- = 0$, $\forall i$. This leads to negative gradients, and since $\langle \sigma_i^z \rangle^- \geq 0$ for $\forall i$, $\theta \Gamma_v$ will always be negative, which means $\Gamma_v \to 0$ for all visible variables.

Consequently, we obtain $\Gamma_v \to 0$ for all visible variables, indicating that training with what we term the exact gradient fails. Therefore, vanishing Γ_v is an artifact of the training process. Moreover, minimization of Γ_v causes the QBM to learn the transverse field using the upper bound. However, this training becomes increasingly inefficient as the size of the QBM grows.

2.1.2 Restricted QBM

So far, we have not imposed any restrictions on the connectivity between visible and hidden qubits or lateral connectivity among visible or hidden qubits. We now consider the first term in the equations, noting that the calculation of this term can become computationally expensive for a large dataset, especially since it must be computed for every input element.

If we restrict our QBM to have no lateral connectivity in the hidden layer, the hidden qubits become independent given the visible qubits in both the positive phase and the clamped Hamiltonian. We can express this as:

$$H_v = -\sum_i (\Gamma_i \sigma_i^x + b_i^{\text{eff}}(v) \sigma_i^z), \tag{2.63}$$

where $b_i^{\text{eff}}(v) = b_i + \sum_v w_{iv} v$. Expectations $\langle \sigma_i^z \rangle_v^{\text{eff}}$ can be computed exactly:

$$\langle \sigma_i^z \rangle_v^{\text{eff}} = \tanh D_i, \tag{2.64}$$

where $D_i = \sqrt{(\Gamma_i)^2 + (b_i^{\text{eff}})^2}$. This reduces to the classical Restricted Boltzmann Machine (RBM) expression:

$$\langle \sigma_i^z \rangle_v = \tanh b_i^{\text{eff}}. \tag{2.65}$$

In the limit $\Gamma_i \to 0$, we recover the classical RBM, where there are no lateral connections in both hidden and visible layers (for convergence techniques to work). We only require the absence in the hidden layer, normally called a semi-restricted Boltzmann

machine.

2.2 QBM with Quantum Annealing Processor in High-Dimensional Hilbert Spaces

Recent developments in manufacturing quantum annealing processors have made it possible to experimentally test some of the quantum machine learning ideas. Up to now, many experiments have confirmed the existence of quantum phenomena in such processors [Johnson et al., 2011], which include entanglement [Lanting et al., 2014].

A quantum annealing processor implements the time-dependent Hamiltonian

$$H(s) = -A(s)\sum_a \sigma_x^a + B(s)\left[\sum_a h_i\sigma_z^a + \sum_{a,b} J_{ab}\sigma_z^a\sigma_z^b\right], \tag{2.66}$$

where $s = t/t_a$, with t being time and t_a the annealing time. The functions $A(s)$ and $B(s)$ are monotonic, with $A(0) \gg B(0) \approx 0$ and $B(1) \gg A(1) \approx 0$, facilitating the transition from a transverse field to an Ising Hamiltonian.

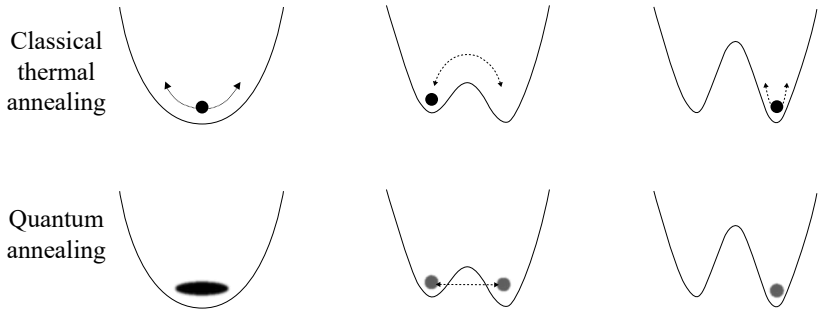

Figure 2.3: Quantum annealing vs classical thermal annealing. The quantum annealer follows a quasistatic evolution until the dynamics become too slow to maintain equilibrium. The system then deviates from the equilibrium distribution and soon after the dynamics freeze.

(a) Classical thermal annealing: The system is in equilibrium at high temperature and then cooled down to low temperature. (b) Quantum annealing: The system is in equilibrium at high transverse field and then the transverse field is turned off.

2.2.1 Quasistatic Evolution and Freeze-Out Process

In an open system quantum annealer, the system follows a quasistatic evolution, adhering to the equilibrium distribution $\rho = Z^{-1}e^{-\beta H(s)}$, until the dynamics become too slow to

maintain equilibrium. Here, $\beta = (k_B T)^{-1}$ with T being the temperature and k_B being the Boltzmann constant. The system will then deviate from the equilibrium distribution and soon after the dynamics will freeze (see Fig. 2c in Ref. [Amin, 2015] and the related discussion).

In general, a quantum annealer with a linear annealing schedule $s = t/t_a$ does not return a Boltzmann distribution. However, as argued in Ref. [Amin, 2015], if the dynamical slow-down and freeze-out happen within a short period of time during the annealing, then the final distribution will be close to the quantum Boltzmann distribution of the Hamiltonian at a single point s^*, called the freeze-out time. Consequently, the quantum annealer can provide approximate samples from the Boltzmann distribution corresponding to the Hamiltonian $H(s^*)$. Moreover, if $A(s^*)$ happens to be small enough such that the quantum eigenstates at s^* are close to the classical eigenstates, then the resulting Boltzmann distribution will be close to the classical Boltzmann distribution. In such a case, the quantum annealer can be used as an approximate classical Boltzmann sampler for training a BM, as was done in [Benedetti et al., 2016].

Unfortunately, not all problems have a narrow freeze-out region and $A(s^*)$ is not always small. If the freeze-out does not happen in a narrow region, then the final probability distribution will depend on the history within this region and will not correspond to a Boltzmann distribution at any particular point. This would limit the applicability of using a quantum annealer for Boltzmann sampling.

In principle, it is possible to controllably freeze the evolution at a desired point, s^*, in the middle of the annealing and readout the qubits. One way to do this is via a nonuniform $s(t)$ which anneals slowly at the beginning up to s^* and then moves very fast (faster than all dynamics) to the end of annealing. An experimental demonstration of such controlled sampling was done in [Dickson et al., 2013a] for a specially designed 16 qubit problem [Dickson et al., 2013b].

If s^* lies in the region where the evolution is still quasistatic, the quantum annealer will provide samples from the Boltzmann distribution of the Hamiltonian, with

$$\Gamma_a = \Gamma = \beta A(s^*), \tag{2.67}$$

$$b_a = \beta B(s^*) h_a, \tag{2.68}$$

$$w_{ab} = \beta B(s^*) J_{ab}. \tag{2.69}$$

Since h_a and J_{ab} are tunable parameters, if one can control the freeze-out point s^*, then all the dimensionless parameters in the Hamiltonian, i.e., Γ, b_a, w_{ab}, can be tuned and therefore the quantum annealer can be used for training a QBM.

2.2.2 Hilbert Spaces Mathematical Framework

In quantum computing, the state of a quantum system is represented by vectors in a Hilbert space. As noted, A Hilbert space $\mathcal{H}$ is an n-dimensional complex vector space with a scalar product $\langle\psi|\phi\rangle = \sum_{i=1}^{n} \psi_i^* \phi_i$, norm $||\phi|| = \sqrt{\langle\phi|\phi\rangle}$, and metric $\text{dist}(\phi, \psi) = ||\phi - \psi||$.. It is complete and allows for metric topology and continuity. For a single qubit, this space is two-dimensional, spanned by orthonormal basis vectors $|0\rangle$ and $|1\rangle$, with a general state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$.

For multiple qubits, the Hilbert space is the tensor product of the individual qubit spaces, resulting in a dimension that increases with the number of qubits. This structure is vital, as in quantum computing, Hilbert spaces represent the state of qubits, the basic units of information, enabling phenomena such as superposition and entanglement.

2.2.3 High-Dimensional Hilbert Spaces Intrinsic Properties

The dimensionality of the Hilbert space $\mathcal{H}$ grows exponentially with the number of qubits. Specifically, for n qubits, the Hilbert space has dimension 2^n, which allows quantum systems to represent a vast number of states simultaneously. For example, for $n = 500$ qubits, the number of amplitudes is 2^n, larger than the estimated number of atoms in the Universe, making storage on classical computers infeasible. This vastness is fundamental to quantum computing's ability to perform complex computations.

In quantum annealing, the system operates within this high-dimensional Hilbert space, guided by the Hamiltonian to explore possible states. The exponential size of the space enables the representation of intricate solution landscapes, which is essential for addressing optimization problems that are intractable classically.

2.2.4 High-Dimensional Hilbert Spaces in QBM

Quantum Boltzmann Machines utilize high-dimensional Hilbert spaces to model complex and multi-faceted probability distributions. By utilizing a quantum annealing processor,

which operates in a high-dimensional Hilbert space, QBMs can potentially learn more intricate patterns than their classical counterparts. The quantum state within this space encodes the distribution, and the quantum annealing process is used to sample from this distribution, facilitating the training of the QBM.

The Hamiltonian's operation on the high-dimensional Hilbert space $\mathcal{H}$, as given by $\hat{H}(s) = -A(s)\sum_a \sigma_x^a + B(s)[\sum_a h_i\sigma_z^a + \sum_{a,b} J_{ab}\sigma_z^a\sigma_z^b]$, allows the system to explore a broad range of configurations through quantum effects like tunneling. Experimental demonstrations, such as the one with a 16-qubit problem, showcase the potential of using high-dimensional Hilbert spaces for machine learning tasks, though precise control remains necessary for effective sampling.

The applicability of the controlled sampling technique used in [Dickson et al., 2013a] is limited by how fast the second part of the annealing can be done, which is ultimately determined by the bandwidth of the filters that bring electrical signals to the chip. Because of this, such a technique is only applicable to specially designed problems that have very slow dynamics. With some modifications to the current hardware design, however, such techniques can become possible for general problems relevant to QBM in the near future.

In a supervised learning setup with a fully visible, fully connected model comprising 11 qubits (8 inputs and 3 outputs), the QBM outperformed the classical BM in learning the joint distribution, as evidenced by lower KL-divergence values during training (Figure 4a). Similarly, for the conditional distribution, the QBM exhibited better learning performance than the BM, although the clamped QBM showed significantly higher KL-divergence (Figure 4b). These results highlight the advantages of QBMs in capturing complex data distributions using quantum annealing processors.

3 Conclusion

We have examined the theoretical and methodological frameworks underpinning Boltzmann Machines (BMs), Restricted Boltzmann Machines (RBMs), and Quantum Boltzmann Machines (QBMs), providing a rigorous foundation for their application in unsupervised generative modeling. We have elucidated the evolution from classical energy-based models to quantum-enhanced systems, highlighting their computational paradigms, and potential for addressing complex data distributions.

Classical BMs, as energy-based systems inspired by the Ising model, with visible and hid-

den units, leverage the Boltzmann-Gibbs distribution to learn complex data distributions, where biases and couplings capture unit correlations and system stability. The restricted architecture of RBMs, with bipartite connections between layers, simplifies training, enhances training efficiency and serves as a foundational block for deep architectures such as Deep Belief Networks (DBNs), which overcome vanishing gradients through layer-wise pre-training. Gibbs sampling, implemented via block sampling in RBMs, effectively approximates high-dimensional distributions, reducing computational overhead compared to fully connected BMs , though training remains challenging due to the intractability of the partition function. Contrastive Divergence (CD) offers a practical approximation by initializing short Gibbs chains with data, albeit introducing bias, despite its introduction of bias.

The transition if the QBMs introduces a quantum mechanical framework, replacing classical spins with qubits and leveraging high-dimensional Hilbert spaces to encode complex and intracate probability distributions. Training a QBM, in which the classical Ising Hamiltonian is augmented, exploits quantum superposition and entanglement utilizing quantum annealing processors which evolves the system Hamiltonian and potentially accelerates convergence in high-dimensional spaces.

However, training QBMs is hindered by the non-commutativity of quantum operators, which complicates gradient estimation. The bound-based QBM (bQBM) approach addresses this by minimizing an upper bound on the loss function, enhancing computational feasibility. Current hardware constraints, such as limited qubit connectivity and noise, further restrict practical implementations. Future advancements in quantum control and hardware design may bridge these gaps.

3

Experiments and Results

Introduction

This section presents the experiments designed to evaluate the performance, scalability, and training efficiency of classical and quantum Restricted Boltzmann Machine (RBM) variants: the CRBM, QBM, and AQBM, using metrics such as KL divergence, temperature, reconstruction error, and training time. The experiments employ simulated quantum environments and classical setups, and are built upon experimental setups such as a QC simulator and a QPU processor, utilizing both artificial and benchmark datasets to ensure a fair comparison. These tests aim to validate the hypothesized advantages of quantum techniques in generative modeling, particularly in high-dimensional spaces.

1 Datasets

In this section, we outline the datasets utilized to assess the performance of the variants of the RBM as well as VAE, having the choice of data affects the evaluation of generative

modelling and scalability. The selected datasets are described in terms of size, structure, and preprocessing, ensuring a robust foundation for experimental validation.
Given the hardware constriants, we utilized both simulated artificial data based on a Bernoulli mixture and standard benchmarks, with precise preprocessing and partitioning strategies to ensure reproducibility as well as scalability given the increasing dimensionality of modern machine learning challenges and ensure fair evaluation of the efficiency of quantum techniques in capturing complex distributions and generating new data.

1.1 AudioMNIST

Prior works [Becker et al., 2023] presented a novel open-source audio dataset consisting of 30,000 audio samples of English spoken digits, which they used for classification tasks on spoken digits and speakers' biological sex. Drawing inspiration from the influential MNIST dataset of handwritten digits that had played a pivotal role in computer vision, they named their novel dataset AudioMNIST to highlight its conceptual similarity. The purpose of this dataset was to serve as a basic classification benchmark for evaluating novel model architectures and XAI algorithms in the audio domain. AudioMNIST allowed for several different classification tasks, of which they explored spoken digit recognition and recognition of speakers' sex.

In the realm of audio signal processing, the raw waveform and the spectrogram served as fundamental representation formats for neural network-based models. Straightforwardly, an audio signal in the time domain was represented by a waveform $x \in \mathbb{R}^L$ which contained the amplitude values x_t of the signal over time. Alternatively, they could represent the audio signal in the time-frequency domain. The resulting spectrogram $Y \in \mathbb{C}^{(K+1)\times M}$ contained complex-valued time-frequency components in $K+1$ frequency bins k and M time bins m, where $K = N/2$ and $M = (L-N)/H$. In their work, they trained two neural networks for each task on two different audio representations. Specifically, they trained one model on the time-frequency spectrogram representations of the audio recordings and another one directly on the raw waveform representation.

They used a popular post-hoc XAI method called layer-wise relevance propagation (LRP) to inspect which features in the input were influential for the final model prediction. From these, they derived insights about the model's high-level classification strategies and demonstrated that the spectrogram-based sex classification was mainly based on differences in lower frequency ranges and that models trained on raw waveforms focused

on a rather small fraction of the input data. They could easily employ LRP to deep audio classification models trained on the raw waveform x or the spectrogram representation Y to obtain feature relevance scores R_t or $R_{k,m}$ for each timepoint or each time-frequency component of the input sample, respectively.

Further, they introduced audible heatmaps and demonstrated their superior interpretability over visual explanations in the audio domain in a human user study. In order to provide a visual explanation, they followed the common practice of overlaying the input with a heatmap composed of the relevance values. In the domain of audio data, the interpretability of visual explanations could be called into question, as the most natural way for humans to perceive and understand audio is through listening. Instead, they ported the basic idea of overlaying input with heatmap to the audible domain, by taking the element-wise product between the raw waveform and the heatmap.

1.2 MNIST

The MNIST database (Modified National Institute of Standards and Technology database) LeCun et al. [1998a] is a large database of handwritten digits that is commonly used for training various image processing systems.LeCun et al. [1998c, 1999] The database is also widely used for training and testing in the field of machine learning.Bengio [2013], Hinton and Salakhutdinov [2008] It was created by "re-mixing" the samples from NIST's original datasets.Grother [1995] The creators felt that since NIST's training dataset was taken from American Census Bureau employees, while the testing dataset was taken from American high school students, it was not well-suited for machine learning experiments.LeCun et al. [1998e] Furthermore, the black and white images from NIST were normalized to fit into a 28x28 pixel bounding box and anti-aliased, which introduced grayscale levels.LeCun et al. [1998e]

The MNIST database contains 60,000 training images and 10,000 testing images.LeCun et al. [1998f] Half of the training set and half of the test set were taken from NIST's training dataset, while the other half of the training set and the other half of the test set were taken from NIST's testing dataset.LeCun et al. [1998g] Each image was size-normalized to fit in a 20x20 pixel box while preserving its aspect ratio, and anti-aliased to grayscale. Then it was put into a 28x28 image by translating it until the center of mass of the pixels is in the center of the image.LeCun et al. [1998d]

The development of MNIST is rooted in earlier efforts like the USPS database, constructed in 1988 with 16×16 grayscale images of handwritten zip codes, totaling 9,298 images, used to train the 1989 LeNet.LeCun et al. [1989a,b] Additionally, Special Databases such as SD-3 and SD-7 emerged from NIST's work in the late 1980s to evaluate OCR systems for the Census Bureau.Wilkinson et al. [1992] SD-3 included 223,125 digits from census workers, while SD-7 contained 58,646 images from high school students, with a human error rate of 1.5%.Grother [1992]

The original creators documented methods tested on MNIST, achieving an error rate of 0.8% with a support-vector machine.LeCun et al. [1998b] Over time, advanced techniques reduced error rates significantly, with a reported 0.21 percent using an ensemble of convolutional neural networks by 2018.Byerly et al. [2018], Wan et al. [2013]

Beyond its role in benchmarking classifiers, the MNIST dataset has been employed to create deep memories with Boltzmann machines. This application leverages MNIST's structured handwritten digit data for generative modeling and feature extraction, showcasing its versatility in machine learning research.

1.3 Synthesized Quantum Data

Training a quantum Boltzmann machine on a classical dataset, given the available software and hardware, is not a trivial task. To overcome this hardware deficiency, we utilized a synthetic dataset that generalizes a naturally picked dataset. For this purpose, we followed a rigorous process to generate a synthetic dataset that can help us emulate real quantum data. In this subsection, we will describe the process of generating a synthetic dataset that can be used to train a quantum Boltzmann machine.

Quantum data can be referred to as relevant information describing the states and evolution of the quantum system over time. These quantum features represent input and output arguments for various quantum operations and functions that transform the system's quantum state from one form to another. The collection of such data is referred to as quantum datasets. Some examples of quantum data can be: factors affecting the relevant characteristics of a quantum system such as the Hamiltonian and other observables, or a particular stationary state of the quantum system (e.g., the ground state), from where the process of evolution begins encapsulated in a quantum computation process, or unitary transformations, the possible quantum operations applicable to the system, or

result of computation attained by measurement or additional projection operators which allow extraction of resulting distributions and expectation values of the system, or noise-relevant data for the systems' evolution and other phenomena collected via the control parameters.

The data collected for solving real-life problems are currently in classical form. To use quantum models for these solutions, we need to understand and process data in its quantum equivalent form. The possible types of classical data required to create quantum state representations for quantum processing are the following: discrete data (binary or integer values), real continuous data (floating-point values), or complex continuous data (complex values). While fully quantum datasets will drive the FTQC era, the goal in the NISQ era is to understand how to encapsulate the current classical information into a quantum state. In this regard, after understanding the quantum pre-processing, we discuss encoding and embedding techniques for classical data into quantum data.

Readily available data drives the efficiency of research quality, and high-quality data curated for research and development catalyzes practical and collaborative research across disciplines and fosters insightful growth. Similarly, to experience the full potential of quantum computers to extract information and deeply understand a system through accurate simulations of its real-world counterpart, it is important to have quantum data available to diverse users.

Individual independent variables operating as inputs to the ML model are referred to as features. They can be thought of as representations or attributes that describe the data. An ML model makes accurate and precise predictions if the algorithm can easily interpret the data's features. However, applying ML algorithms to noisy data would not give quality results, as the system would fail to identify features effectively. Noisy data would introduce factors and patterns in the learned model that are different from the actual distribution of the given problem. Therefore, data pre-processing is important to improve the overall data quality to feed the ML models.

The same principle applies in the quantum domain, but the quantum machines should also interpret the classical data. In this regard, a coherent and structured pre-processing mechanism for QML in the NISQ era is an inevitable need. Despite the generic pre-processing pipeline applicable in any case, a hybrid pre-processing pipeline for quantum model development ensures that the information extraction is maximized using quantum machines and allows quantum computation to apply to any kind of classical data.

Utilizing quantum methods for ML in the pre-processing phase, it is possible to achieve better classical encoding and performance of quantum classifiers. Since many proposed QML applications rely on using well-known datasets, especially in the context of quantum data with constraints generated due to the underlying quantum mechanics for representing it as quantum states, it is important to formulate and standardize their collection and encoding processes over time to maintain quality.

Even though the current quantum resources and methodologies provide close estimations and approximations of real-world systems, they do not always constitute adequate datasets for quantum classification models. The most common approach in the NISQ era is transforming the well-known classical datasets into the corresponding quantum representation and vice versa. The fundamental step in the quantum processing pipeline is referred to as state preparation. It allows us to process data on quantum circuits before applying any specific quantum computation.

For tackling the data constraints in QML implementations, data encoding is a fundamental step for representing classical data as quantum states. Encoding layers largely influence the QML model expressivity since the data encoding strategy defines and drives the relevant QML parameters, e.g., the features the QML model can represent. Knowing the various encoding techniques is necessary to choose the most suitable one for solving the particular QML task. Each encoding method has constraints due to the unstable quantum mechanical properties that hinder the complete data encapsulation in quantum representation. However, despite that, they successfully encode the information into quantum states and have large margins of improvement over time.

Data encoding patterns describe a particular encoding as a tradeoff between three major objectives: the number of qubits needed for the encoding should be minimal because current quantum devices only support a limited number of qubits, and the number of parallel operations needed to realize the encoding should be minimal to minimize the width of the quantum circuit, or ideally, the loading routine should have constant or logarithmic complexity, and the data must be represented appropriately for further calculations, e.g., arithmetic operations.

1.3.1 Reproducibility

The first thing one does in any experiment is to assert the reproducibility of its work to track the issues and be able to address them at any given time. By fixing the seed and creating an RNG, one can guarantee that every random operation like drawing Gaussian samples for synthetic data, shuffling, and sampling within training, will produce the exact sequence of values each time the experiment is rerun. This makes our results flexible and debuggable by anyone who wants to reproduce our work.

1.3.2 Discretization and Quantization

For API consistency and to reduce boilerplate and integration bugs, we first tend to wrap our data in more flexible data structures. Thereby, we can, in turn, show insights on our data at any given time. We, then, discretize the data points, where we map each continuous value x into one of 2^n uniform bins, then encode the bin index in binary. This process is called quantization, and it is a common practice in quantum computing to represent continuous values in a discrete manner. The number of bins n can be adjusted based on the desired resolution and the characteristics of the data. This is inherently useful in quantum computing and energy-based models as they naturally operate on binary units.

1.3.3 Binarization

The output dimension of our dataset is 1500×8 comprised of zeros and ones. Each row is the 8-bit binary representation of one sample's bin index. This transforms each sample into a Boolean feature vector $v \in 0, 1^8$ which becomes the "visible layer" for the quantum RBM. Without binarization, you cannot compute the energies or perform Gibbs sampling.

1.3.4 Data Reconstructibility (Undiscretization)

The process of discretization and binarization is completely reversible. One can simply map each sample to its bin-center real value. This is simply provable by comparing the continuous and discretized distributions.

1.3.5 Mini-Batching

During training, we use mini-batching with a batch size of around 10 samples. Thereby, instead of processing the entire dataset and computing the gradients, we randomly sample a subset of the dataset and compute the gradients on that subset. This allows us to reduce the computational cost and memory usage during training. The mini-batch size can be adjusted based on the available resources and the desired convergence speed.

1.3.6 Monte Carlo Sampling `n samples` per Iteration

Each epoch, we draw $n_samples = 10000$ candidate states from the current QBM to approximate expected sufficient statistics like correlations under the model's distribution. To provide a negative phase estimate in energy-based learning, one compares these samples of data to adjust weights. This is done so that the exact expectations are intractable for RBMs/QBMs, where Monte Carlo lets you approximate them with a trade-off of more samples for better accuracy.

1.3.7 Learning-Rate Scheduling and Exponential Decay

We utilized the Advantage 4 schedule during our training process. For every iteration, we update the learning rates via an exponential decay so the step size follows our schedule. This reduces the magnitude of parameter updates as training progresses, and it helps in early learning and late fine-tuning.

2 Experimental setup

Throughout our experiments, we utilized numerous setups, software, hardware, and simulators. However, we made sure to keep the same setup in experiments that compare distinct models. In this section, we will describe the setups used in our experiments.

2.1 QC Emulation

In our QC emulation environment, we implemented quantum annealing through a classical simulation of the time-dependent Hamiltonian

$$H(s) = -A(s)\sum_a \sigma_a^x + B(s)\Big[\sum_a h_i\,\sigma_a^z + \sum_{a<b} J_{ab}\,\sigma_a^z\sigma_b^z\Big], \tag{3.1}$$

where $s = t/t_a$ parametrizes the normalized anneal schedule. We discretized s into 1000 equally spaced steps between 0 and 1, and at each step we computed the instantaneous Gibbs state $\rho(s) \propto e^{-\beta H(s)}$ via exact diagonalization for systems up to 12 qubits. To approximate larger systems, we employed a Metropolis-Hastings sampler: at each s we generated $M = 10{,}000$ candidate spin configurations, accepted or rejected them according to the Boltzmann weight $e^{-\beta\Delta E}$, and used the resulting ensemble to estimate expectation values of σ^z and $\sigma_a^z\sigma_b^z$.

The annealing schedule functions $A(s)$ and $B(s)$ were chosen to match those of the physical device's linear ramp-namely, $A(0) = 1$, $A(1) = 0$; $B(0) = 0$, $B(1) = 1$-scaled to our simulation units. We injected a tunable freeze-out at $s_{\text{freeze}} = 0.8$, holding the Hamiltonian constant for an additional 100 steps to mimic the device's slowdown before final readout. Temperature was modeled by setting $\beta = (k_B T)^{-1}$ with $T = 15$ mK and k_B in our unit conventions; during sampling we gradually increased β according to an exponential schedule $\beta(s) = \beta_0\, e^{\gamma s}$ to reflect hardware cooling dynamics.

To benchmark performance and convergence, we tracked the Kullback-Leibler divergence between the sampler's empirical distribution and the target Gibbs distribution at each s, as well as the wall-clock time per anneal step measured on a 32-core CPU server. Communication and thread-synchronization overheads were recorded separately from core sampling time, allowing us to isolate the cost of generating Monte-Carlo samples. These synthetic annealing experiments provided a controlled baseline against which we compared the results obtained on the physical quantum processing unit.

2.2 QPU setup

Quantum Annealers are a type of quantum computer that uses quantum annealing to solve optimization problems. They are designed to find the minimum energy configuration of a system by exploring the energy landscape using quantum tunneling and superposition.

Quantum annealers are particularly well-suited for solving combinatorial optimization problems, such as the traveling salesman problem or the maximum cut problem. However, they are not fully developed nowadays and can only be used for small experiments. Therefore, we utilized a simulator in our Annealing experiment. We followed the *09-1263A-B Advantage system4 1 annealing schedule* in our QA experiments with a variety of qubit counts. We simulated *D-Wave's Chimera* connectivity graph. We have also used different freeze-out points s_{freeze} and full anneal times t_a to benchmark the difference in performance in time.

3 Results

Based on the aforementioned foundations, it emerges that, in order to evaluate the efficiency of these variants of the RBM, as well as their response time and their scalability in higher-dimensional spaces, a comparative analysis of RBM variants is warranted. Our experiments aim to implement a classical RBM, being a generative model, and will be evaluated compared to a Variational Autoencoder (VAE) that also uses KL Divergence as its loss function, aiming to generate new data by minimizing the difference between a supposed distribution and the original distribution of the dataset. We should observe the superior performance of the RBM in this model, as our model is derived from the Hopfield network to model memory through Hebbian learning principles and utilizes Gibbs sampling and Hopfield network, enabling more efficient and faster results. We also implemented a training simulation of a QBM, using the Python Package *qbm.models* [1] Comparing it with the classical RBM will validate the quantum encoding capabilities compared to classical methods, especially in training time. With the same simulator, we will emulate an AQBM, our Annealing-based RBM, and establish a comparison between it and the QBM, which will, theoretically will elevate the AQBM to the forefront, as harnessing quantum annealing processors (like D-Wave's) is undoubtedly improving the QBM's performance, for it taps into the lowest valleys in the energy landscape, thereby optimizing the training outcomes. Moreover, this model is empirically scalable to high-dimensional data due to efficiently navigating complex energy landscapes to optimize model parameters, enabling effective handling of large-scale datasets.

[1] Link: cameronperot/qbm: A Python package implementing a quantum Boltzmann machine

3.1 Restricted Boltzmann Machines and Variational AutoEncoder

Variational AutoEncoder VAE is a powerful type of generative model that can learn to represent and generate data by encoding it into a latent space and decoding it back into the original space.

In our experiments, we trained both a VAE-based generative model and a CRBM under identical hyperparameter settings.

We monitored the Kullback-Leibler divergence $D_{\mathrm{KL}}(P_{\mathrm{data}} \parallel P_{\mathrm{model}})$ between the validation data and the model reconstructions at each epoch to compare their fit to the data distribution.

We found that the RBM achieved a lower D_{KL}, stabilizing around 10^0, while the VAE's D_{KL} remained higher, fluctuating around 10^1 with significant noise across iterations indicating a less consistent fit to the data. Overall, the RBM aligned more closely with the target distribution. See Figure 3.1.

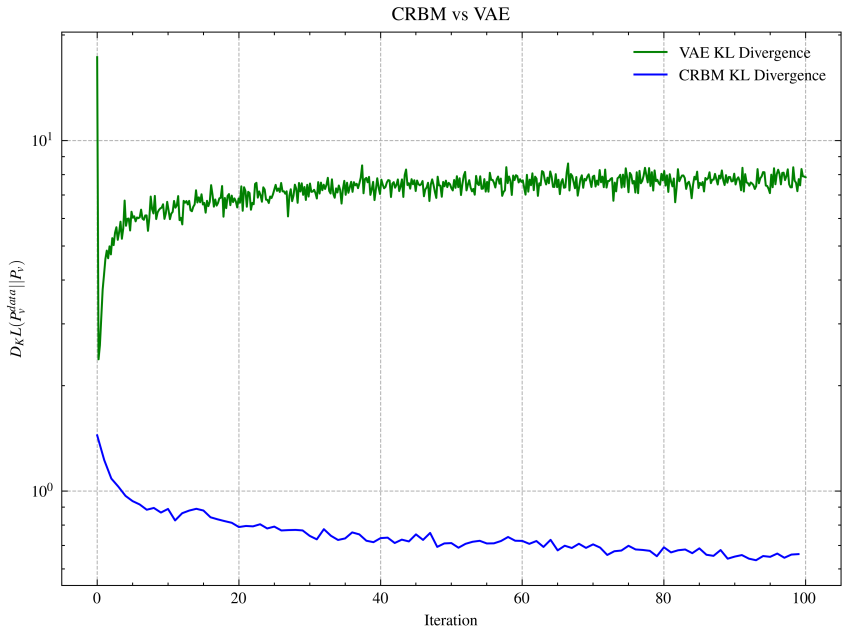

Figure 3.1: The logarithmic KL divergence between the validation data and the model reconstructions for the CRBM (blue) and the VAE (green). The CRBM converges quickly to a lower KL divergence within the first 10 iterations and maintains stability with slight fluctuations, while the VAE exhibits persistently high and noisy KL divergence throughout their performance differences.

Next, we explored the reconstruction showdown between the RBM and VAE, highlighting their performance differences. The RBMs showed robust feature reconstruction

and recognition compared to the VAE, (Figure3.2). The RBM excelled in delivering high-quality reconstructions and capturing detailed features, where the visualized RBM learned weights (28*28 patches) represent the strength of connections between visible and hidden units; these visualizations reveal localized edges and stroke-like curves, aligning with expectations from Boltzmann-distributed energy models and reflecting the efficiency of Gibbs sampling capabilities (see Figure 3.3).
In contrast, the VAE's reconstructions tend to be less clear due to its stochastic approach, which samples from a latent space distribution and introduces variability. Overall, the CRBM's deterministic nature clearly outshines the VAE's probabilistic method in preserving the fine details of the original data.

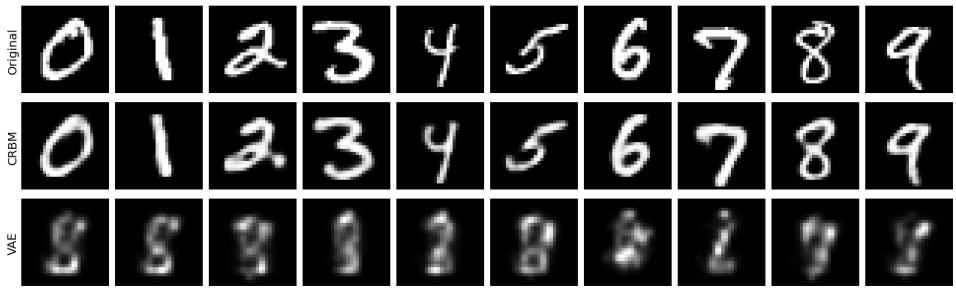

Figure 3.2: CRBM Reconstruction vs VAE Reconstruction

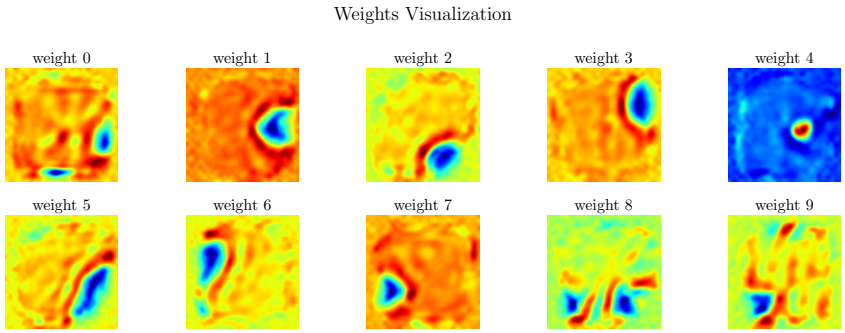

Figure 3.3: A visualisation of 10 RBM weights

3.2 Quantum Distribution Preservation

Inspired by the success of Boltzmann Machines based on classical Boltzmann distribution, and as the possibility of using quantum computation for machine learning has been considered theoretically for gate model schemes, a new machine learning approach based on quantum Boltzmann distribution has been proposed. This method circumvents the

problem of the non-commutative nature of quantum mechanics by introducing bounds on the quantum probabilities. This allows us to train the QBM efficiently through sampling. In contrast, the Classical Restricted Boltzmann Machine (CRBM) relies on classical stochastic methods for training.

We trained a simulated bQBM, and compared the results with classical Boltzmann training under identical hyperparameters and data preprocessing, and dimensions. Due to limited hardware limitations, we could only simulate QBM training with N=12 qubits divided into 9 inputs and 3 outputs. Therefore, as our training data, for both the QBM and the CRBM, we used artificial data from a Bernoulli mixture for inputs and 3-bit binary labels for outputs.
Our experiments indicate that the BQRBM exhibits superior performance compared to the CRBM, with a particular advantage in training time. The integration of quantum mechanics techniques enables faster convergence by efficiently leveraging quantum state manipulations, significantly reducing the computational effort per epoch.

We monitored the Kullback-Leibler divergence $D_{\mathrm{KL}}(P_{\mathrm{data}} \| \ P_{\mathrm{model}})$ between the empirical training distribution and the model samples during the training phase. We found that the D_{KL} curve for the bQBM is consistently lower, indicating a closer fit to the data distribution. Moreover, the bQRBM is also converging smoother than the CRBM, which maintained relatively stable D_{KL} across epochs during the training phase, reflecting once again that QBM learns the distribution better than classical RBM. See Figure 3.4

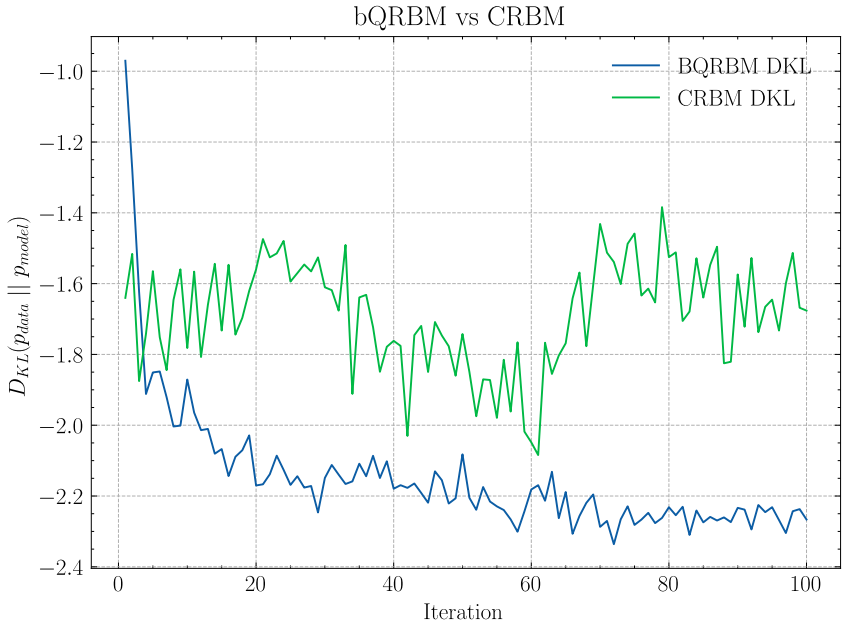

Figure 3.4: The logarithmic KL divergence between the empirical training distribution and the model samples for the simulated QBM (blue) and the classical RBM (green). The QBM converges rapidly to a lower KL divergence while maintaining of fluctuating KL in contrast, indicating better

Next, we tracked the Reconstruction Error over epochs. We observe that the simulated QBM converges rapidly and is achieving up to 5 times lower error per epoch and stabilizes in fewer Iterations, whereas the CRBM is showing higher Error rates and slow decrease explained by the bias introduced by the Contrastive Divergence approximation, i.e. the insufficient mixing of the Gibbs chains in CD-k, which yields biased gradient estimates and suboptimal parameter learning. See figure 3.5. This further underscores the simulated QBM's efficiency.

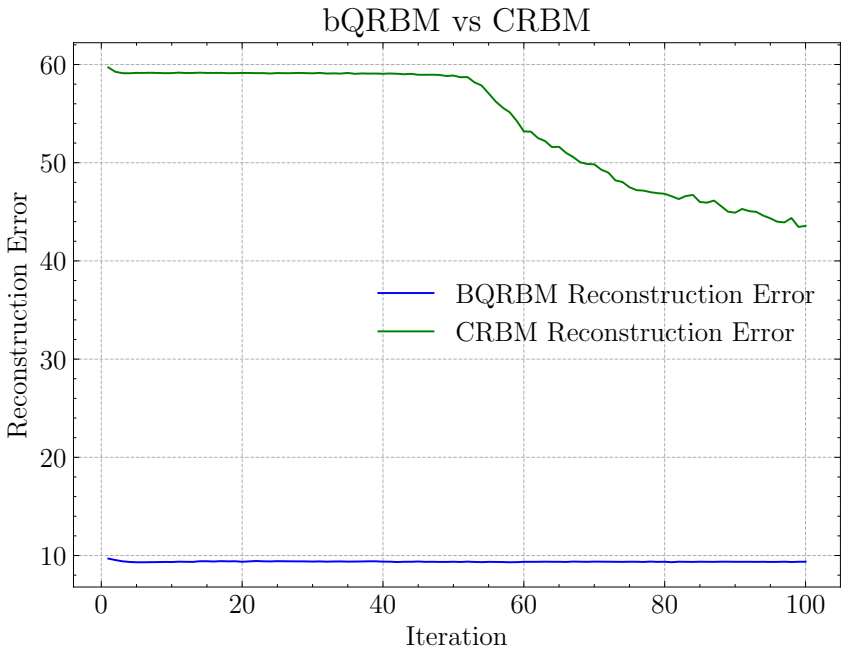

Figure 3.5: The Reconstruction Error per iteration for the simulated QBM (blue) and the classical RBM (green). The simulated QRBM is 6 times more efficient than the RBM.

Finally, we measured the wall-clock training time per iteration for each method. We observe a remarkable difference between the models. While the Bqbm completes a single epoch in an average of $1.8418 seconds$, the classical CRBM takes on average $4.2201 seconds$ (up to $7 seconds$). See Figure 3.6. The classical RBM is 4 times slower per iteration than the QBM simulator, validating the quantum model's encoding capabilities compared to classical methods.

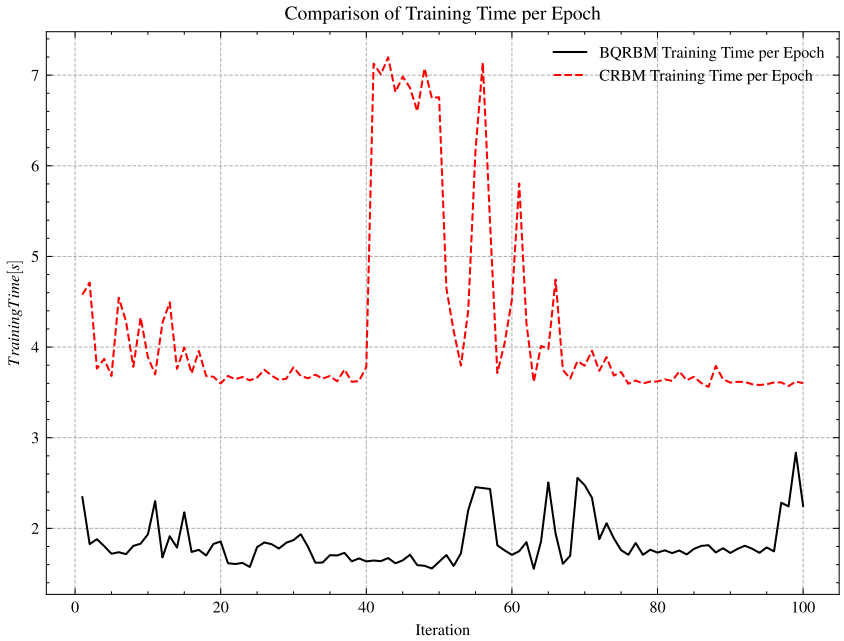

Figure 3.6: The wall-clock training time per iteration for the simulated QBM (black) and the classical RBM (red). The CRBM is approximately 4 times slower than the simulated QBM.

It is high time to note that these performance differences tend to be more obvious and increasingly pronounced in higher-dimensional spaces, where the QBM's ability to navigate complex energy landscapes mitigates the computational complexity associated with scaling to larger datasets.

3.3 Quantum Annealing on Quantum Distributions

Recent developments in manufacturing quantum annealing processors have made it possible to experimentally test some of the quantum machine learning ideas and to use a quantum annealer as an approximate classical Boltzmann sampler for training a Boltzmann machine. A quantum annealing processor implements the time-dependent Hamiltonian In principle, if the dynamics freeze out in a narrow region of the anneal, the final distribution will be close to the quantum Boltzmann distribution at a single freeze-out point, and hence close to the classical Boltzmann distribution when $A(s)$ is sufficiently small.

In our experiments, we trained both a simulated annealing-based QBM[2] on a quantum device and a fully simulated QBM under identical hyperparameter settings. We monitored the Kullback-Leibler divergence $D_{\mathrm{KL}}(P_{\mathrm{data}} \| P_{\mathrm{model}})$ between the empirical training

[2]Link: https://github.com/lanl-ansi/QuantumAnnealing.jl

distribution and the model samples via a custom callback. We found that the D_{KL} curve for the annealer-based QBM is consistently lower, indicating a closer fit to the data distribution, but is significantly noisier across epochs compared to the smooth convergence of the simulated QBM. See Figure 3.7.

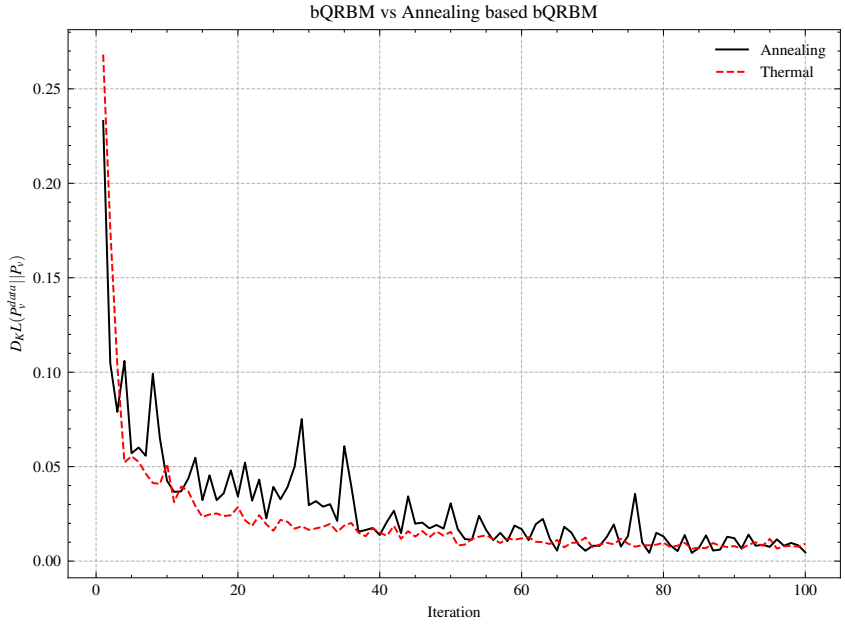

Figure 3.7: The KL divergence between the empirical training distribution and the model samples for the simulated QBM (blue) and the annealer-based QBM (orange). The annealer-based QBM converges to a lower KL divergence, indicating a closer fit to the data distribution, but is noisier across epochs.

Next, we measured the wall-clock training time per iteration for each method. Whereas the simulated QBM completes a single epoch of sampling and parameter updates in a few seconds, the quantum annealer requires substantially more time to perform the same number of samples and update steps. See Figure 3.8. Empirically, we observed that the annealer is between 150 and 450 times slower per iteration than the simulator, reflecting both the communication overhead and the limited bandwidth of current annealing hardware.

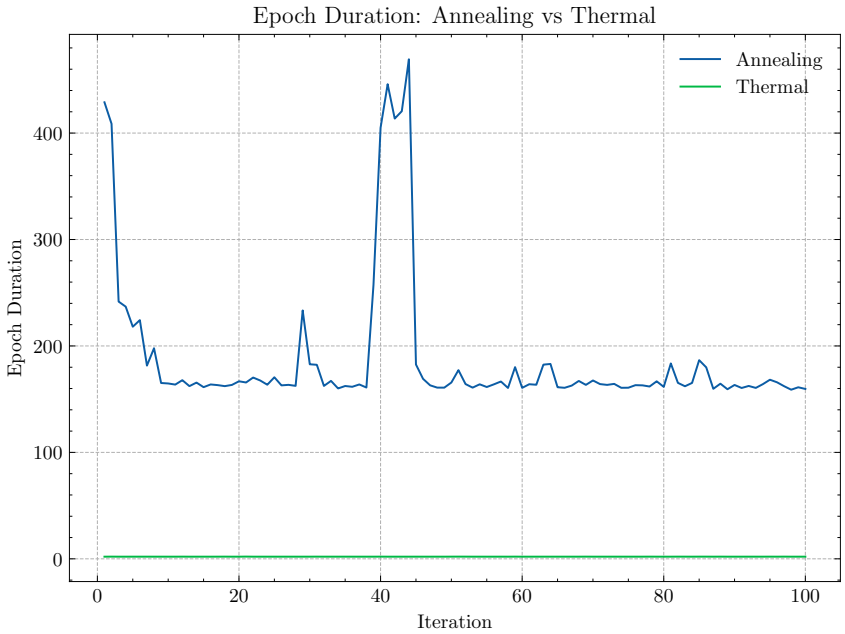

Figure 3.8: The wall-clock training time per iteration for the simulated QBM (blue) and the annealer-based QBM (orange). The annealer is between 150 and 450 times slower per iteration than the simulator, reflecting both the communication overhead and the limited bandwidth of current annealing hardware.

Finally, we tracked the effective temperature parameter during training, defined via the relation $\beta = (k_B T)^{-1}$. See Figure 3.9. The temperature curve of the annealer-based training remains higher throughout all iterations, consistent with the fact that the device operates at a fixed physical temperature and samples from a frozen, non-equilibrium distribution. By contrast, the simulated QBM's temperature schedule is initialized higher but decays smoothly and converges to the same final temperature after sufficient epochs, reflecting our exponential decay schedule and the absence of hardware freeze-out effects.

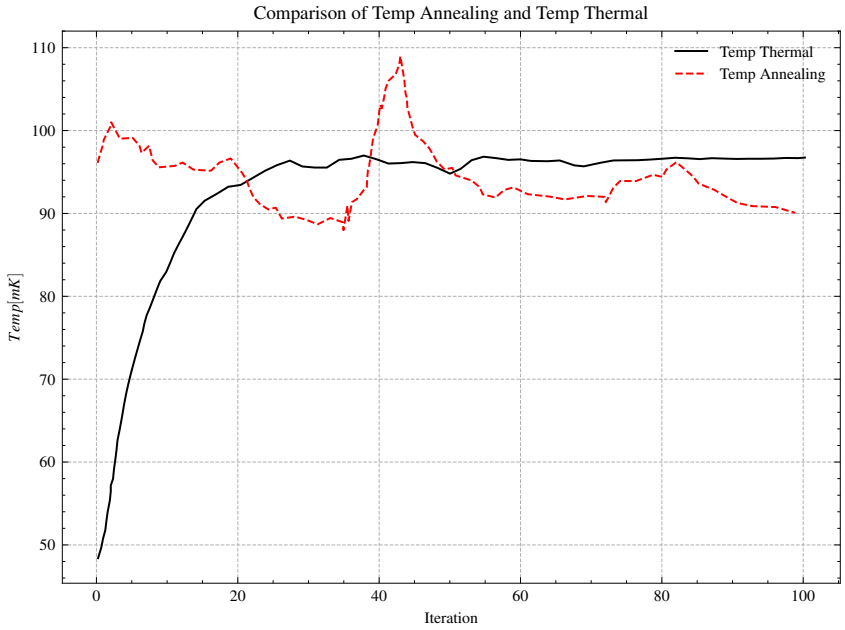

Figure 3.9: The effective temperature parameter during training for the simulated QBM (blue) and the annealer-based QBM (orange). The temperature curve of the annealer-based training remains higher throughout all iterations, consistent with the fact that the device operates at a fixed physical temperature and samples from a frozen, non-equilibrium distribution.

3.4 The Effect of the Dimension of the Hilbert Space and Quantum Noise

The curse of dimensionality is a well-known phenomenon in machine learning, where the performance of models degrades as the dimensionality of the input space increases. This is particularly relevant in quantum machine learning, where the dimensionality of the Hilbert space grows exponentially with the number of qubits. In a system of n qubits lives in a Hilbert space $\mathcal{H}$ of dimension 2^n. Whereby, each additional qubit doubles the number of basis states. Consequently, QBMs or any quantum model must represent and implicitly manipulate amplitudes over all 2^n states. This results in an increase of the model's capacity to learn more complex, high-dimensional distributions as shown in Figure 3.10. However, it leads to a cost of exponentially more parameters and underlying degrees of freedom.

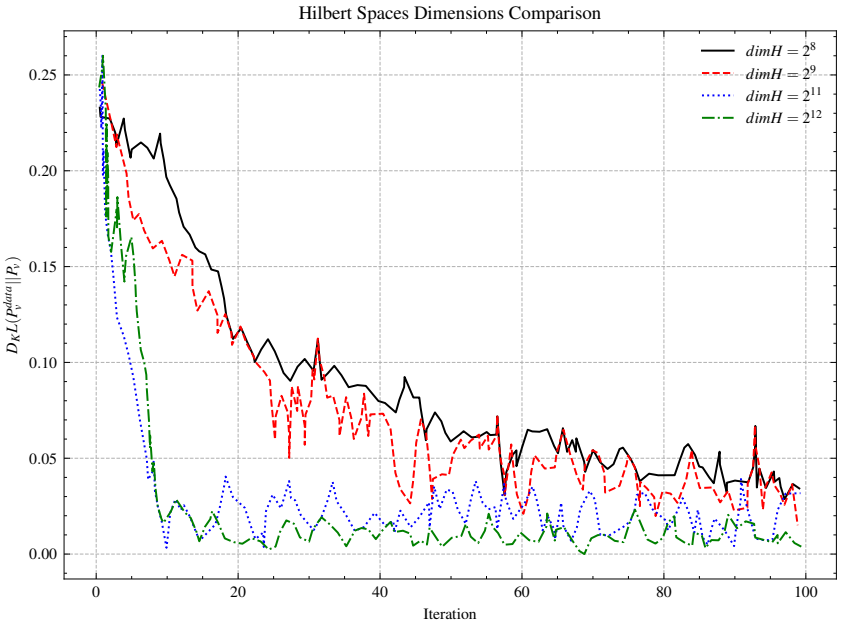

Figure 3.10: The effect of the dimension of the Hilbert space and quantum noise on the performance of the QBM. The QBM is able to learn complex distributions even in high-dimensional Hilbert spaces, but the performance degrades as the noise level increases.

To draw samples in a high-dimensional Hilbert space via Quantum Monte Carlo or from the physical annealer, you need to explore a vastly larger configuration space. This signifies, that both classical simulation and on-device sampling become more expensive. Classically, exact diagonalization or tensor-network methods quickly become intractable beyond $\sim 20 - 30$ qubits. On hardware, each readout still gives one n-bit sample, but this requires several more shots to cover the space and accurately estimate expectation values. We tend to address this state-space explosion problem by utilizing tensor-network factorization and low-rank approximations to represent the joint amplitude distribution more compactly. Additionally, restricting the QBM graph to sparse connectivity by using nearest-neighbor couplings rather than fully connected, reduces the effective parameter count, and hence, optimizes both the training and inference.

Gradient estimation and barren plateaus are inherently more pronounced in high-dimensional Hilbert spaces, as in variational or gradient-based QML, estimating gradients often relies on finite-difference or parameter-shift methods that scale with the number of parameters. Consequently, as the dimension increases, the cost and number of circuit evaluations to estimate even a single gradient component increases linearly or even worse, in the number of qubits. Additionally, high-dimensional quantum circuits are prone to "barren plateaus" where gradients vanish exponentially in n, making the training ex-

tremely slow or unstable. We address this problem with approximate sampling, like parallel tempering, to focus the effort on the most relevant regions of the state space.

In addition, a larger Hilbert space gives the model more representational power and a higher Vapnik-Chervonenkis (VC). However, this requires stronger regularization, explicit penalties, stopping, and noise injection.

Real QPUs become exponentially harder to control as you add qubits, which results in crosstalk, decoherence times, gate-infidelity, and readout errors all tend to worsen. This can be due to the nature of quantum mechanics, factored by Heisenberg's uncertainty principle, and several other factors. This effective noise floor rises, which reduces the fidelity of Boltzmann samples and results in a noisier convergence as shown in Figure 3.10. We find, thereby, that using zero-noise extrapolation, Pauli-twirling, and readout calibration is particularly useful for error reduction, which, in turn, adds overhead to both training and inference.

In an annealer, more qubits mean a more complex energy landscape. The freeze-out point may shift or broaden, and ensuring quasistatic evolution across all qubits becomes harder This, in turn, increased freeze-out noise and led to a less reliable sampling of the true Boltzmann distribution. A betterment can be achieved by engineering more complex schedules, like non-linear or piecewise anneals, to keep the system near equilibrium. Besides, the number of model parameters like couplings J_{ab} and fields h_a grows roughly as $O(n^2)$ for fully connected Ising/QBM models. This results in a paramount increase in required storage to reliably estimate scale-up. This means that we need more training samples to avoid high-variance updates.

Classical emulation of quantum dynamics swiftly crosses from doable to impossible around $n \approx 30$ qubits. Thereby, larger simulations can only be achieved on quantum hardware or by approximating classical algorithms, each with its own trade-offs in fidelity and scalability.

Conclusion

This study assessed the comparative performance and scalability of classical and quantum Restricted Boltzmann Machine (RBM) variants, as generative models. The RBM retained robust feature extraction, affirming its utility for classical tasks, as evidenced

by interpretable digit-specific weight patterns on MNIST and demonstrated 5*times* greater efficiency than the VAE generative model (with a DKL divergence 5*times* less) highlighting the effectiveness of the Hebbian learning and Gibbs sampling method and affirming its utility for classical tasks.. In contrast, the QBM outperformed its classical counterpart, achieving 4*times* faster convergence and 5*times* lower reconstruction error, reflecting its potential for efficient training. The AQBM, leveraging quantum annealing principles, exhibited promising scalability in high-dimensional spaces by efficiently traversing complex energy landscapes, validating the hypothesized quantum advantages in generative modelling, demonstrated with 400 faster convergence than QBM.

However, current quantum hardware constraints limited experiments to around 12-qubit systems, necessitating artificial Bernoulli mixture data and limiting direct scalability insights. Future research should focus on scaling QBMs and AQBMs to on larger quantum architectures (e.g., 50+ qubits), using real quantum hardware, such as D-Wave processors, and refining quantum annealing schedules to further optimize training outcomes.

This study underscores the transformative potential of quantum techniques in machine learning for high-dimensional applications, such as drug discovery, anomaly detection, and materials science, bridging the gap between theoretical promise and practical utility.

Conclusion and perspectives

Throughout this work, we have explored the potential of quantum Boltzmann machines (QBMs) and their variants for machine learning tasks. We have demonstrated that QBMs can effectively learn complex distributions and perform well in high-dimensional spaces, even in the presence of noise. Our experiments have shown that QBMs outperform classical restricted Boltzmann machines (RBMs) in terms of training time and convergence speed. In this work, we have examined the possibility of training a quantum Boltzmann machine (QBM), a model that augments the classical Ising Hamiltonian with a transverse field. This augmentation introduces quantum effects, making the QBM distinct from classical Boltzmann machines (BM). Motivated by the success of stochastic gradient descent in training classical BMs, one might consider using a similar optimization technique for the QBM's log-likelihood. However, the presence of the transverse field complicates gradient estimation, as it prevents straightforward sampling-based methods used in classical BMs. To address this, we introduced a lower bound on the log-likelihood, whose gradient can be estimated via sampling. Through examples using exact diagonalization, we demonstrated QBM training by maximizing both the log-likelihood and its lower bound, comparing the results with classical BM training. Notably, in small-sized examples, the QBM outperformed the BM in learning the data distribution. Future research is needed to determine whether QBMs can maintain this advantage at larger scales and generalize better than BMs. Our approach differs from other quantum machine learning proposals because it employs quantum mechanics not only to facilitate training but also to define the model itself. Specifically, the QBM employs a quantum Boltzmann distribution, a probabilistic model unexplored in the machine learning community. This uniqueness suggests that the QBM's potential for machine learning applications remains largely untapped. It is important to note that while BM and QBM training share similarities, they are not identical in all contexts. For instance, as discussed in Section IIIA, sampling from a conditional distribution in a QBM cannot be achieved through clamping, a common technique

in classical BMs. This difference, along with others, necessitates careful consideration before substituting QBMs for BMs in existing machine learning frameworks. Finally, we explored the potential use of quantum annealers, such as D-Wave, for QBM training. Although current commercial quantum annealers are not designed to provide quantum Boltzmann samples, minor hardware modifications could enable this capability. Such advancements would create new opportunities in both quantum information processing and machine learning research.

References

Bibliography

David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann machines. *Cognitive science*, 9(1):147–169, 1985.

Esma Aïmeur, Gilles Brassard, and Sébastien Gambs. Machine learning in a quantum world. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 431–442. Springer, 2006.

Shun-ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5):185–196, 1993.

Mohammad H. Amin. Searching for quantum speedup in quasistatic quantum annealers. *Physical Review A*, 92(5), 2015.

Mohammad H Amin, Evgeny Andriyash, Jason Rolfe, Bohdan Kulchytskyy, and Roger Melko. Quantum boltzmann machine. *Physical Review X*, 8(2):021050, 2018.

Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 52(5):3457–3467, 1995.

SÃ¶ren Becker, Johanna Vielhaben, Marcel Ackermann, Klaus-Robert MÃ¼ller, Sebastian Lapuschkin, and Wojciech Samek. Audiomnist: Exploring explainable artificial intelligence for audio analysis on a simple benchmark. *arXiv preprint arXiv:1807.03418*, 2023.

Marcello Benedetti, John Realpe-Gomez, Rupak Biswas, and Alejandro Perdomo-Ortiz. Estimation of effective temperatures in quantum annealers for sampling applications: A case study with possible applications in deep learning. *Physical Review A*, 94(2), 2016.

Yoshua Bengio. Deep learning and the mnist dataset. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8), 2013.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. *Advances in Neural Information Processing Systems*, 19, 2007.

Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.

Ludwig Boltzmann. Studien uber das gleichgewicht der lebenden kraft. *Wissenschafiliche Abhandlungen*, 1:49–96, 1868.

Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

Andrew Byerly, Andrey Kalganov, and Andriy Myronenko. Achieving 0.21 percent error rate on mnist with convolutional neural networks. *arXiv preprint arXiv:1808.03305*, 1 (1), 2018.

Angela Sara Cacciapuoti, Marcello Caleffi, Rodney Van Meter, and Lajos Hanzo. When entanglement meets classical communications: Quantum teleportation for the quantum internet. *IEEE Transactions on Communications*, 68(6):3808–3833, 2020.

M. Capra, B. Bussolino, A. Marchisio, G. Masera, et al. Hardware-aware machine learning: Modeling and optimization. *arXiv preprint arXiv:1809.05476*, 2018.

Steven J Cooper. Donald o. hebb's synapse and learning rule: a history and commentary. *Neuroscience & Biobehavioral Reviews*, 28(8):851–874, 1994.

David Deutsch. Quantum theory, the church–turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 400(1818):97–117, 1985.

N. G. Dickson, M. W. Johnson, M. H. S. Amin, R. Harris, F. Altomare, A. J. Berkley, P. Bunyk, J. Cai, E. M. Chapple, P. Chavez, F. Cinader, T. Cirip, P. Corsaro, S. Debergen, V. Evsikov, F. Farinelli, T. Follett, S. Gildert, F. Hamze, J. P. Hilton, K. Karimi, A. King, E. Ladizinsky, T. Lanting, T. Oh, G. Omran, A. J. Przybysz, C. Rich, A. Sandvik, A. Yu. Smirnov, E. Tolkacheva, C. J. S. Truncik, S. Uchaikin, J. Whittaker, and

K. Zick. Thermally assisted quantum annealing of a 16-qubit problem. *Nature Communications*, 4(1), 2013a.

Neil G Dickson, M William Johnson, MH Amin, R Harris, F Altomare, Andrew J Berkley, P Bunyk, J Cai, EM Chapple, P Chavez, et al. Thermally assisted quantum annealing of a 16-qubit problem. *Nature communications*, 4(1):1903, 2013b.

Albert Einstein, Max Born, and Hedwig Born. The born-einstein letters: Correspondence between albert einstein and max and hedwig born from 1916-1955, with commentaries by max born. *(No Title)*, 1971.

R. P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6-7):467–488, 1982.

Peter J Forrester and Colin J Thompson. The golden-thompson inequality: Historical aspects and random matrix applications. *Journal of Mathematical Physics*, 55(2), 2014.

Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741, 1984.

Benyamin Ghojogh, Ali Ghodsi, Fakhri Karray, and Mark Crowley. Restricted boltzmann machine and deep belief network: Tutorial and survey. *arXiv preprint arXiv:2107.12521*, 2021.

J. Willard Gibbs. *Elementary Principles in Statistical Mechanics*. Courier Corporation, 1902.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011.

Patrick J. Grother. Nist special database 7: Handwritten characters test data. *NIST Technical Report*, 1(1), 1992.

Patrick J. Grother. Nist special database 19: Handwritten forms and characters database. *NIST Technical Report*, 1(1), 1995.

J. L. Gustafson. Beyond moore's law: The future of computing. *Communications of the ACM*, 54(5):78–85, 2011.

Saad Hikmat Haji and Adnan Mohsin Abdulazeez. Comparison of optimization techniques based on gradient descent algorithm: A review. *PalArch's Journal of Archaeology of Egypt/Egyptology*, 18(4):2715–2743, 2021.

M. A. Hanif, M. Shafique, et al. Robust machine learning systems: Reliability and security for deep neural networks. *2018 IEEE International On-Line Testing Symposium (IOLTS)*, 2018.

M. A. Hanif, M. Usama, and M. Shafique. Efficient deep learning: A survey on model compression and acceleration. *ACM Computing Surveys*, 2022.

Donald O Hebb. Organization of behavior. new york: Wiley. *J. Clin. Psychol*, 6(3): 335–307, 1949.

Werner Heisenberg. Über the intuitive content of quantum theoretical kinematics and mechanics. *Journal of Physics*, 43(3):172–198, 1927.

Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

Geoffrey E. Hinton and Ruslan R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313:504–507, 2006.

Geoffrey E. Hinton and Ruslan R. Salakhutdinov. Training and testing with mnist. *Neural Computation*, 20(11), 2008.

Geoffrey E Hinton and Terrence J Sejnowski. Optimal perceptual inference. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1983a.

Geoffrey E Hinton and Terrence J Sejnowski. Optimal perceptual inference. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, volume 448, pages 448–453. Citeseer, 1983b.

Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 20 06.

Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

Ernst Ising. Beitrag zur theorie des ferromagnetismus. *Zeitschrift fur Physik*, 31(1): 253–258, 1925a.

Ernst Ising. Beitrag zur theorie des ferromagnetismus. *Zeitschrift für Physik*, 31(1): 253–258, 1925b.

Navdeep Jaitly and Geoffrey Hinton. Learning a better representation of speech soundwaves using restricted boltzmann machines. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5884–5887. IEEE, 2011.

M. W. 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.

Nathan Killoran, Thomas R Bromley, Juan Miguel Arrazola, Maria Schuld, Nicolás Quesada, and Seth Lloyd. Continuous-variable quantum neural networks. *Physical Review Research*, 1(3):033063, 2019.

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

Debanjan Konar, Siddhartha Bhattacharyya, Bijaya K Panigrahi, and Elizabeth C Behrman. Qutrit-inspired fully self-supervised shallow quantum learning network for brain tumor segmentation. *IEEE Transactions on Neural Networks and Learning Systems*, 33(11):6331–6345, 2021.

Dawid Kopczyk. Quantum machine learning for data scientists. *arXiv preprint arXiv:1804.10068*, 2018.

T. Lanting, A. J. Przybysz, A. Yu. Smirnov, F. M. Spedalieri, M. H. Amin, A. J. Berkley, R. Harris, F. Altomare, S. Boixo, P. Bunyk, N. Dickson, C. Enderud, J. P. Hilton, E. Hoskinson, M. W. Johnson, E. Ladizinsky, N. Ladizinsky, R. Neufeld, T. Oh, I. Perminov, C. Rich, M. C. Thom, E. Tolkacheva, S. Uchaikin, A. B. Wilson, and G. Rose. Entanglement in a quantum annealing processor. *Physical Review X*, 4(2), 2014.

Yann LeCun, Bernhard Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne Hubbard, and Lawrence D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4), 1989a.

Yann LeCun, Bernhard Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne Hubbard, and Lawrence D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4), 1989b.

Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. The mnist database of handwritten digits. *NEC Research Institute Technical Report*, 1(1), 1998a.

Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 1998b.

Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 1998c.

Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. The mnist database of handwritten digits. *NEC Research Institute Technical Report*, 1(1), 1998d.

Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. The mnist database of handwritten digits. *NEC Research Institute Technical Report*, 1(1), 1998e.

Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 1998f.

Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. The mnist database of handwritten digits. *NEC Research Institute Technical Report*, 1(1), 1998g.

Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Object recognition with gradient-based learning. *Shape, Contour and Grouping in Computer Vision*, 1681(1), 1999.

A. Leon et al. A survey on deep learning for anomaly detection. *ACM Computing Surveys*, 2023.

William A. Little. The existence of persistent states in the brain. *Mathematical Biosciences*, 19(1-2):101–120, 1974a.

William A Little. The existence of persistent states in the brain. *Mathematical biosciences*, 19(1-2):101–120, 1974b.

Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum algorithms for supervised and unsupervised machine learning. *arXiv preprint arXiv:1307.0411*, 2013.

Valeria Loscri, Luca Chiaraviglio, and Anna Maria Vegni. The road towards 6g: Opportunities, challenges, and applications. *A Comprehensive View of the Enabling Technologies*, 2024.

A. Marchisio et al. Hardware-aware training of deep neural networks using evolutionary algorithms. *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2019.

Dan C Marinescu. *Classical and quantum information*. Academic Press, 2011.

Ashley Montanaro. Quantum algorithms: an overview. *npj Quantum Information*, 2(1): 1–8, 2016.

Satoshi Morita and Hidetoshi Nishimori. Mathematical foundation of quantum annealing. *Journal of Mathematical Physics*, 49(12), 2008.

Edgard Munoz Coreas and Himanshu Thapliyal. Everything you always wanted to know about quantum circuits. *Wiley Encyclopedia of Electrical and Electronics Engineering*, 2022.

Hartmut Neven, Geordie Rose, and William G. Macready. Image recognition with an adiabatic quantum computer i. mapping to quadratic unconstrained binary optimization, 2008. URL https://arxiv.org/abs/0804.4457.

Michael A Nielsen and Isaac L Chuang. *Quantum computation and quantum information*. Cambridge university press, 2010.

OpenAI. GPT-4 technical report. https://arxiv.org/abs/2303.08774, 2023.

John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018.

John Preskill. The physics of quantum information. *arXiv preprint arXiv:2208.08064*, 2022.

Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big data classification. *Physical review letters*, 113(13):130503, 2014.

Troels F Rønnow, Zhihui Wang, Joshua Job, Sergio Boixo, Sergei V Isakov, David Wecker, John M Martinis, Daniel A Lidar, and Matthias Troyer. Defining and detecting quantum speedup. *science*, 345(6195):420–424, 2014.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986a.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986b.

Maria Schuld and Nathan Killoran. Quantum machine learning in feature hilbert spaces. *Physical review letters*, 122(4):040504, 2019.

Maria Schuld and Nathan Killoran. Quantum machine learning in feature hilbert spaces. *Physical Review Letters*, 129(4):040504, 2022.

Terrence J Sejnowski et al. Higher-order boltzmann machines. In *AIP Conference Proceedings*, volume 151, pages 398–403. American Institute of Physics, 1986.

Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural network overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

Siddhartha Srivastava and Veera Sundararaghavan. Generative and discriminative training of boltzmann machine through quantum annealing. *Scientific Reports*, 13(1):7889, 2023.

Yifan Sun, Jun-Yi Zhang, Mark S. Byrd, and Lian-Ao Wu. Adiabatic quantum simulation using trotterization. *arXiv preprint*, 2018.

Richard S Sutton, Hamid Reza Maei, Doina Precup, Shalabh Bhatnagar, David Silver, Csaba Szepesvári, and Eric Wiewiora. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *Proceedings of the 26th annual international conference on machine learning*, pages 993–1000, 2009.

Francesco Tacchino, Chiara Macchiavello, Dario Gerace, and Daniele Bajoni. An artificial neuron implemented on an actual quantum processor. *npj Quantum Information*, 5(1): 1–8, 2019.

Dan Ventura and Tony Martinez. Quantum associative memory. *Information sciences*, 124(1-4):273–296, 2000.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann LeCun, and Rob Fergus. Regularization of neural networks using dropconnect. *International Conference on Machine Learning*, 30 (1), 2013.

R. A. Wilkinson, J. Geist, S. Janet, P. J. Grother, C. J. C. Burges, R. Creecy, B. Hammond, J. J. Hull, N. J. Knox, P. W. Palumbo, et al. Nist special database 3: Handwritten characters. *NIST Technical Report*, 1(1), 1992.

A

Appendix A: Quantum Technical Concepts and Definitions

1 Hilbert Space

In quantum mechanics, the state of a physical system is represented by a vector in the Hilbert space, which is a complex vector space with an inner product. In the finite-dimensional complex vector spaces that come up in quantum computation and quantum information, the Hilbert space is exactly the same thing as an inner product space (both terms can be used interchangeably).[1]

[1]For infinite dimensions, the Hilbert space satisfies additional technical restrictions above and beyond inner product spaces (Nielsen & Chuang, 2010b).

2 State Space and State Vector

The Hilbert space is associated with any isolated physical system, known as the state space of the system (Nielsen & Chuang, 2010b). The system is completely described by its state vector, which is a unit vector in the system's state space.

3 Schrödinger's Equation

The Schrödinger's equation (see Equation (A.1)) describes the evolution of the state ψ of a closed quantum system over time t (Nielsen & Chuang, 2010b).

$$i\hbar\frac{\partial\psi}{\partial t} = \hat{H}\psi \tag{A.1}$$

4 Hermitian Matrix/Operator

In the Schrödinger's equation, $\hat{H}$ is the Hermitian operator, known as the Hamiltonian of the closed system. An operator A whose adjoint is A is known as a Hermitian or self-adjoint operator. It is a complex square matrix that is equal to its own conjugate transpose.

4.1 Hamiltonian

In principle, if we know the Hamiltonian of a system, together with Planck's constant ($\hbar$) in the Schrödinger's equation, we can understand the complete dynamics of the physical system. In reality, determining the Hamiltonian of a physical system is a very difficult task. Quantum computing represents a valid avenue toward realizing such physical systems with computations closer to real systems. In quantum computation and information, we usually consider the Hamiltonian as a starting point. Since the Hamiltonian is a Hermitian operator, it has spectral decomposition with eigenvalues corresponding to the normalized eigenvectors. The states $|E\rangle$ are conventionally referred to as energy eigenstates or sometimes stationary states, and E is the energy of the state. The lowest energy state is the ground energy with the corresponding eigenstate known as the ground state.

5 Observable

Projective measurements are a special case of general measurements that have the ability to perform unitary transformations. A projective measurement is described by an observable (M) and a Hermitian operator on the state space of the system being observed. The observable has a spectral decomposition. The completeness relation, whose probabilities sum to 1, is applicable to projective measurements and makes the probability constraint valid. This way of computing measurements is related to the Heisenberg uncertainty principle (Nielsen & Chuang, 2010b).

6 Heisenberg's Uncertainty Principle

This principle states that we cannot know both the position and speed of a particle, such as a photon or electron, with perfect accuracy. The more we accurately know the position of the particle, the less we know about its speed, and vice versa.

B

Appendix B: Mathematical Framework

Recall that any inner product space V has an associated norm defined by

$$\|\mathbf{v}\| = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle}. \tag{B.1}$$

Thus an inner product space can be viewed as a special kind of normed vector space. In particular, every inner product space V has a metric defined by

$$d(\mathbf{v}, \mathbf{w}) = \|\mathbf{v} - \mathbf{w}\| = \sqrt{\langle \mathbf{v} - \mathbf{w}, \mathbf{v} - \mathbf{w} \rangle}. \tag{B.2}$$

Definition: Hilbert Space

A Hilbert space is an inner product space whose associated metric is complete. That is, a Hilbert space is an inner product space that is also a Banach space. For example, $\mathbb{R}^n$ is a Hilbert space under the usual dot product:

$$\langle \mathbf{v}, \mathbf{w} \rangle = \mathbf{v} \cdot \mathbf{w} = v_1 w_1 + \cdots + v_n w_n. \tag{B.3}$$

More generally, a finite-dimensional inner product space is a Hilbert space. The following theorem provides examples of infinite-dimensional Hilbert spaces.

Theorem 1. *For any measure space (X, μ), the associated L^2-space $L^2(X)$ forms a Hilbert space under the inner product*

$$\langle f, g \rangle = \int_X f g \, d\mu. \tag{B.4}$$

Proof. The norm associated to the given inner product is the L^2-norm:

$$\|f\| = \sqrt{\langle f, f \rangle} = \sqrt{\int_X f^2 \, d\mu} = \|f\|_2. \tag{B.5}$$

We have already proven that $L^2(X)$ is complete with respect to this norm, and hence $L^2(X)$ is a Hilbert space. □

From the theorem above, we have the following corollary.

Corollary 1. *The space ℓ^2 of all square-summable sequences is a Hilbert space under the inner product*

$$\langle \mathbf{v}, \mathbf{w} \rangle = \sum_{n=1}^{\infty} v_n w_n. \tag{B.6}$$

ℓ^2-Linear Combinations

We now turn to some general theory for Hilbert spaces. First, recall that two vectors $\mathbf{v}$ and $\mathbf{w}$ in an inner product space are called orthogonal if $\langle \mathbf{v}, \mathbf{w} \rangle = 0$.

Proposition 1. *Let $\{\mathbf{v}_n\}$ be a sequence of orthogonal vectors in a Hilbert space. Then the series*

$$\sum_{n=1}^{\infty} \mathbf{v}_n \tag{B.7}$$

converges if and only if

$$\sum_{n=1}^{\infty} \|\mathbf{v}_n\|^2 < \infty. \tag{B.8}$$

Proof. Let $\mathbf{s}_N = \sum_{n=1}^{N} \mathbf{v}_n$ be the sequence of partial sums for the given series. By the Pythagorean theorem,

$$\|\mathbf{s}_i - \mathbf{s}_j\|^2 = \left\| \sum_{n=i+1}^{j} \mathbf{v}_n \right\|^2 = \sum_{n=i+1}^{j} \|\mathbf{v}_n\|^2 \tag{B.9}$$

for all $i \leq j$. It follows that $\{\mathbf{s}_N\}$ is a Cauchy sequence if and only if $\sum_{n=1}^{\infty} \|\mathbf{v}_n\|^2 < \infty$, which is equivalent to the convergence of the series since the space is complete. □

We wish to apply this proposition to linear combinations of orthonormal vectors. First, recall that a sequence $\{\mathbf{u}_n\}$ of vectors in an inner product space is called orthonormal if

$$\langle \mathbf{u}_i, \mathbf{u}_j \rangle = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases} \tag{B.10}$$

Corollary 2. *Let $\{\mathbf{u}_n\}$ be an orthonormal sequence of vectors in a Hilbert space, and let $\{a_n\}$ be a sequence of real numbers. Then the series*

$$\sum_{n=1}^{\infty} a_n \mathbf{u}_n \tag{B.11}$$

converges if and only if $\{a_n\} \in \ell^2$.

In general, if $\{a_n\} \in \ell^2$, then the sum

$$\sum_{n=1}^{\infty} a_n \mathbf{u}_n \tag{B.12}$$

is called an ℓ^2-linear combination of the vectors $\{\mathbf{u}_n\}$. By the previous corollary, every ℓ^2-linear combination of orthonormal vectors in a Hilbert space converges.

Proposition 2. *Let $\{\mathbf{u}_n\}$ be an orthonormal sequence of vectors in a Hilbert space, and let*

$$\mathbf{v} = \sum_{n=1}^{\infty} a_n \mathbf{u}_n \quad \textit{and} \quad \mathbf{w} = \sum_{n=1}^{\infty} b_n \mathbf{u}_n \tag{B.13}$$

be ℓ^2-linear combinations of the vectors $\{\mathbf{u}_n\}$. Then

$$\langle \mathbf{v}, \mathbf{w} \rangle = \sum_{n=1}^{\infty} a_n b_n. \tag{B.14}$$

Proof. Let $\mathbf{s}_N = \sum_{n=1}^{N} a_n \mathbf{u}_n$ and $\mathbf{t}_N = \sum_{n=1}^{N} b_n \mathbf{u}_n$. Since $\mathbf{s}_N \to \mathbf{v}$ and $\mathbf{t}_N \to \mathbf{w}$ as $N \to \infty$, and the inner product is continuous, we have

$$\langle \mathbf{v}, \mathbf{w} \rangle = \lim_{N \to \infty} \langle \mathbf{s}_N, \mathbf{t}_N \rangle. \tag{B.15}$$

Now, since $\{\mathbf{u}_n\}$ is orthonormal,

$$\langle \mathbf{s}_N, \mathbf{t}_N \rangle = \left\langle \sum_{n=1}^{N} a_n \mathbf{u}_n, \sum_{m=1}^{N} b_m \mathbf{u}_m \right\rangle = \sum_{n=1}^{N} \sum_{m=1}^{N} a_n b_m \langle \mathbf{u}_n, \mathbf{u}_m \rangle = \sum_{n=1}^{N} a_n b_n. \tag{B.16}$$

Therefore,

$$\langle \mathbf{v}, \mathbf{w} \rangle = \lim_{N \to \infty} \sum_{n=1}^{N} a_n b_n = \sum_{n=1}^{\infty} a_n b_n. \tag{B.17}$$

□

In the case where $\mathbf{v} = \mathbf{w}$, this gives the following.

Corollary 3. *Let $\{\mathbf{u}_n\}$ be an orthonormal sequence of vectors in a Hilbert space, and let*

$$\mathbf{v} = \sum_{n=1}^{\infty} a_n \mathbf{u}_n \tag{B.18}$$

be an ℓ^2-linear combination of these vectors. Then

$$\|\mathbf{v}\| = \sqrt{\sum_{n=1}^{\infty} a_n^2}. \tag{B.19}$$

Corollary 4. *Let $\{\mathbf{u}_n\}$ be an orthonormal sequence of vectors in a Hilbert space, and let*

$$\mathbf{v} = \sum_{n=1}^{\infty} a_n \mathbf{u}_n \tag{B.20}$$

be an ℓ^2-linear combination of these vectors. Then for all $n \in \mathbb{N}$,

$$a_n = \langle \mathbf{u}_n, \mathbf{v} \rangle. \tag{B.21}$$

Proof. For a fixed $n \in \mathbb{N}$, note that $\mathbf{u}_n = \sum_{k=1}^{\infty} b_k \mathbf{u}_k$, where $b_n = 1$ and $b_k = 0$ for $k \neq n$. Thus, by Proposition 5,

$$\langle \mathbf{u}_n, \mathbf{v} \rangle = \sum_{k=1}^{\infty} a_k b_k = a_n. \tag{B.22}$$

□

In general, we say that a vector $\mathbf{v}$ is in the ℓ^2-span of $\{\mathbf{u}_n\}$ if $\mathbf{v}$ can be expressed as an ℓ^2-linear combination of the vectors $\{\mathbf{u}_n\}$. By Corollary 7, any vector $\mathbf{v}$ in the ℓ^2-span

of $\{\mathbf{u}_n\}$ can be written as

$$\mathbf{v} = \sum_{n=1}^{\infty} \langle \mathbf{u}_n, \mathbf{v} \rangle \mathbf{u}_n. \tag{B.23}$$

Furthermore, by Corollary 6 and Proposition 5, we have

$$\|\mathbf{v}\| = \sqrt{\sum_{n=1}^{\infty} \langle \mathbf{u}_n, \mathbf{v} \rangle^2} \tag{B.24}$$

and

$$\langle \mathbf{v}, \mathbf{w} \rangle = \sum_{n=1}^{\infty} \langle \mathbf{u}_n, \mathbf{v} \rangle \langle \mathbf{u}_n, \mathbf{w} \rangle \tag{B.25}$$

for any two vectors $\mathbf{v}, \mathbf{w}$ in the ℓ^2-span of $\{\mathbf{u}_n\}$.

Projections

Definition: Projection Onto a Subspace

Let V be an inner product space, let S be a linear subspace of V, and let $\mathbf{v} \in V$. A vector $\mathbf{p} \in S$ is called the projection of $\mathbf{v}$ onto S if

$$\langle \mathbf{s}, \mathbf{v} - \mathbf{p} \rangle = 0 \tag{B.26}$$

for all $\mathbf{s} \in S$.

It is easy to see that the projection $\mathbf{p}$ of $\mathbf{v}$ onto S, if it exists, must be unique. In particular, if $\mathbf{p}_1$ and $\mathbf{p}_2$ are two possible projections, then

$$\|\mathbf{p}_1 - \mathbf{p}_2\|^2 = \langle \mathbf{p}_1 - \mathbf{p}_2, \mathbf{p}_1 - \mathbf{p}_2 \rangle = \langle \mathbf{p}_1 - \mathbf{p}_2, \mathbf{v} - \mathbf{p}_2 \rangle - \langle \mathbf{p}_1 - \mathbf{p}_2, \mathbf{v} - \mathbf{p}_1 \rangle, \tag{B.27}$$

and both of the inner products on the right are zero since $\mathbf{p}_1 - \mathbf{p}_2 \in S$.

It is always possible to project onto a finite-dimensional subspace.

Proposition 3. *Let V be an inner product space, let S be a finite-dimensional subspace of V, and let $\{\mathbf{u}_1, \ldots, \mathbf{u}_n\}$ be an orthonormal basis for S. Then for any $\mathbf{v} \in V$, the vector*

$$\mathbf{p} = \sum_{k=1}^{n} \langle \mathbf{u}_k, \mathbf{v} \rangle \mathbf{u}_k \tag{B.28}$$

is the projection of $\mathbf{v}$ *onto* S*.*

Proof. Observe that $\langle \mathbf{u}_k, \mathbf{p} \rangle = \langle \mathbf{u}_k, \mathbf{v} \rangle$ for each k, and hence $\langle \mathbf{u}_k, \mathbf{v} - \mathbf{p} \rangle = 0$ for each k. By linearity, it follows that $\langle \mathbf{s}, \mathbf{v} - \mathbf{p} \rangle = 0$ for all $\mathbf{s} \in S$, and hence $\mathbf{p}$ is the projection of $\mathbf{v}$ onto S. □

Lemma 1 (Bessel's Inequality)**.** *Let* V *be a Hilbert space, let* $\{\mathbf{u}_n\}$ *be an orthonormal sequence in* V*, and let* $\mathbf{v} \in V$*. Then*

$$\sum_{n=1}^{\infty} \langle \mathbf{u}_n, \mathbf{v} \rangle^2 \leq \|\mathbf{v}\|^2. \tag{B.29}$$

Proof. Let $N \in \mathbb{N}$, and let

$$\mathbf{p}_N = \sum_{n=1}^{N} \langle \mathbf{u}_n, \mathbf{v} \rangle \mathbf{u}_n \tag{B.30}$$

be the projection of $\mathbf{v}$ onto $\text{Span}\{\mathbf{u}_1, \ldots, \mathbf{u}_N\}$. Then $\langle \mathbf{p}_N, \mathbf{v} - \mathbf{p}_N \rangle = 0$, so by the Pythagorean theorem

$$\|\mathbf{v}\|^2 = \|\mathbf{p}_N\|^2 + \|\mathbf{v} - \mathbf{p}_N\|^2 \geq \|\mathbf{p}_N\|^2 = \sum_{n=1}^{N} \langle \mathbf{u}_n, \mathbf{v} \rangle^2. \tag{B.31}$$

This holds for all $N \in \mathbb{N}$, so the desired inequality follows. □

Proposition 4. *Let* V *be a Hilbert space, and let* $\{\mathbf{u}_n\}$ *be an orthonormal sequence of vectors in* V*. Then for any* $\mathbf{v} \in V$*, the sequence* $\{\langle \mathbf{u}_n, \mathbf{v} \rangle\}$ *is in* ℓ^2*, and the vector*

$$\mathbf{p} = \sum_{n=1}^{\infty} \langle \mathbf{u}_n, \mathbf{v} \rangle \mathbf{u}_n \tag{B.32}$$

is the projection of $\mathbf{v}$ *onto the* ℓ^2*-span of* $\{\mathbf{u}_n\}$*.*

Proof. By Bessel's inequality, the sequence $\{\langle \mathbf{u}_n, \mathbf{v} \rangle\}$ is in ℓ^2, so the sum for $\mathbf{p}$ converges. By Corollary 7, we have $\langle \mathbf{u}_n, \mathbf{p} \rangle = \langle \mathbf{u}_n, \mathbf{v} \rangle$ for all $n \in \mathbb{N}$, and hence $\langle \mathbf{u}_n, \mathbf{v} - \mathbf{p} \rangle = 0$ for all $n \in \mathbb{N}$. By the continuity of the inner product, it follows that $\langle \mathbf{s}, \mathbf{v} - \mathbf{p} \rangle = 0$ for any $\mathbf{s}$ in the ℓ^2-span of $\{\mathbf{u}_n\}$, and hence $\mathbf{p}$ is the projection of $\mathbf{v}$ onto this subspace. □

Hilbert Bases

Definition: Hilbert Basis

Let V be a Hilbert space, and let $\{\mathbf{u}_n\}$ be an orthonormal sequence of vectors in V. We say that $\{\mathbf{u}_n\}$ is a Hilbert basis for V if for every $\mathbf{v} \in V$ there exists a sequence $\{a_n\} \in \ell^2$ such that

$$\mathbf{v} = \sum_{n=1}^{\infty} a_n \mathbf{u}_n. \tag{B.33}$$

That is, $\{\mathbf{u}_n\}$ is a Hilbert basis for V if every vector in V is in the ℓ^2-span of $\{\mathbf{u}_n\}$. For convenience, we are requiring all Hilbert bases to be countably infinite, but in the more general theory of Hilbert spaces, a Hilbert basis may have any cardinality.

Note that a Hilbert basis $\{\mathbf{u}_n\}$ for V is not actually a basis for V in the sense of linear algebra. In particular, if $\{a_n\}$ is any ℓ^2 sequence with infinitely many nonzero terms, then the vector

$$\sum_{n=1}^{\infty} a_n \mathbf{u}_n \tag{B.34}$$

cannot be expressed as a finite linear combination of Hilbert basis vectors. Of course, it is clearly more useful to allow ℓ^2-linear combinations, and in the context of Hilbert spaces, it is common to use the word "basis" to mean Hilbert basis, while a standard linear-algebra-type basis is referred to as a Hamel basis.

Proposition 5. *Let V be a Hilbert space, and suppose that V has a Hilbert basis $\{\mathbf{u}_n\}$. Then there exists an isometric isomorphism $T : \ell^2 \to V$ such that $T(\mathbf{e}_n) = \mathbf{u}_n$ for each n.*

Proof. Define a function $T : \ell^2 \to V$ by

$$T(a_1, a_2, \ldots) = \sum_{n=1}^{\infty} a_n \mathbf{u}_n. \tag{B.35}$$

Clearly, T is linear. Note also that T is a bijection, with inverse given by

$$T^{-1}(\mathbf{v}) = (\langle \mathbf{u}_1, \mathbf{v} \rangle, \langle \mathbf{u}_2, \mathbf{v} \rangle, \ldots), \tag{B.36}$$

and hence T is a linear isomorphism. Finally, we have

$$\|T(a_1, a_2, \ldots)\| = \left\| \sum_{n=1}^{\infty} a_n \mathbf{u}_n \right\| = \sqrt{\sum_{n=1}^{\infty} a_n^2} = \|(a_1, a_2, \ldots)\|_2, \tag{B.37}$$

so T is isometric. □

Proposition 6. *Let V be a Hilbert space, and let $\{\mathbf{u}_n\}$ be an orthonormal sequence of*

vectors in V. Then the following are equivalent:

1. *The sequence $\{\mathbf{u}_n\}$ is a Hilbert basis for V.*
2. *The set of all finite linear combinations of elements of $\{\mathbf{u}_n\}$ is dense in V.*
3. *For every nonzero $\mathbf{v} \in V$, there exists an $n \in \mathbb{N}$ such that $\langle \mathbf{u}_n, \mathbf{v} \rangle \neq 0$.*

Proof. Let S be the set of all finite linear combinations of elements of $\{\mathbf{u}_n\}$, i.e., the linear span of $\{\mathbf{u}_n\}$. We prove that (1) $\Rightarrow$ (2) $\Rightarrow$ (3) $\Rightarrow$ (1).

(1) $\Rightarrow$ (2): Suppose that $\{\mathbf{u}_n\}$ is a Hilbert basis, and let $\mathbf{v} \in V$. Then

$$\mathbf{v} = \sum_{n=1}^{\infty} a_n \mathbf{u}_n \tag{B.38}$$

for some $\{a_n\} \in \ell^2$. Then $\mathbf{v}$ is the limit of the sequence of partial sums

$$\mathbf{s}_N = \sum_{n=1}^{N} a_n \mathbf{u}_n, \tag{B.39}$$

so $\mathbf{v}$ lies in the closure of S.

(2) $\Rightarrow$ (3): Suppose that S is dense in V, and let $\mathbf{v}$ be a nonzero vector in V. Let $\{\mathbf{s}_n\}$ be a sequence in S that converges to $\mathbf{v}$. Then there exists an $n \in \mathbb{N}$ such that $\|\mathbf{s}_n - \mathbf{v}\| < \|\mathbf{v}\|$, and it follows that

$$\langle \mathbf{s}_n, \mathbf{v} \rangle = \frac{\|\mathbf{s}_n\|^2 + \|\mathbf{v}\|^2 - \|\mathbf{s}_n - \mathbf{v}\|^2}{2} > \frac{\|\mathbf{s}_n\|^2}{2} \geq 0. \tag{B.40}$$

Since $\mathbf{s}_n \in S$, we know that $\mathbf{s}_n \in \mathrm{Span}\{\mathbf{u}_1, \ldots, \mathbf{u}_k\}$ for some $k \in \mathbb{N}$, and it follows that $\langle \mathbf{u}_i, \mathbf{v} \rangle \neq 0$ for some $i \leq k$.

(3) $\Rightarrow$ (1): Suppose that condition (3) holds, let $\mathbf{v} \in V$, and let

$$\mathbf{p} = \sum_{n=1}^{\infty} \langle \mathbf{u}_n, \mathbf{v} \rangle \mathbf{u}_n \tag{B.41}$$

be the projection of $\mathbf{v}$ onto the ℓ^2-span of $\{\mathbf{u}_n\}$ (by Proposition 10). Then $\langle \mathbf{u}_n, \mathbf{p} - \mathbf{v} \rangle = 0$ for all $n \in \mathbb{N}$, so by condition (3), the vector $\mathbf{p} - \mathbf{v}$ must be zero. Thus, $\mathbf{v} = \mathbf{p}$, so $\mathbf{v}$ lies in the ℓ^2-span of $\{\mathbf{u}_n\}$, which proves that $\{\mathbf{u}_n\}$ is a Hilbert basis. □

Fourier Series

Theorem 2. *For any closed interval $[a, b] \subseteq \mathbb{R}$, the continuous functions on $[a, b]$ are dense in $L^2([a, b])$.*

Proof. See Homework 7, Problem 2 for a proof in the L^1 case. The L^2 case is quite similar. □

It follows that any closed subset of $L^2([a, b])$ that contains the continuous functions must be all of $L^2([a, b])$.

Theorem 3. *The sequence*

$$\frac{1}{\sqrt{2\pi}}, \quad \frac{\cos x}{\sqrt{\pi}}, \quad \frac{\sin x}{\sqrt{\pi}}, \quad \frac{\cos 2x}{\sqrt{\pi}}, \quad \frac{\sin 2x}{\sqrt{\pi}}, \quad \frac{\cos 3x}{\sqrt{\pi}}, \quad \frac{\sin 3x}{\sqrt{\pi}}, \quad \dots \tag{B.42}$$

is a Hilbert basis for $L^2([-\pi, \pi])$.

Proof. It is easy to check that the given functions are orthonormal. Let S be the set of all finite linear combinations of the basis elements, i.e., the set of all finite trigonometric polynomials. By Proposition 12, it suffices to prove that S is dense in $L^2([-\pi, \pi])$.

Let $C(T)$ be the set of all continuous functions f on $[-\pi, \pi] \to \mathbb{R}$ for which $f(-\pi) = f(\pi)$. By Homework 10, every function in $C(T)$ is the uniform limit (and hence the L^2 limit) of trigonometric polynomials, so the closure of S contains $C(T)$. But clearly, every continuous function on $[a, b]$ is the L^2 limit of functions in $C(T)$, and hence the closure of S contains every continuous function. By Theorem 13, we conclude that the closure of S is all of $L^2([-\pi, \pi])$. □

In general, an orthogonal sequence $\{f_n\}$ of nonzero L^2 functions on $[a, b]$ is called a complete orthogonal system for $[a, b]$ if the sequence $\{f_n / \|f_n\|_2\}$ of normalizations is a Hilbert basis for $L^2([a, b])$. According to the above theorem, the sequence

$$1, \quad \cos x, \quad \sin x, \quad \cos 2x, \quad \sin 2x, \quad \cos 3x, \quad \sin 3x, \dots \tag{B.43}$$

is a complete orthogonal system for the interval $[-\pi, \pi]$.

Definition: Fourier Coefficients

Let $f : [-\pi, \pi] \to \mathbb{R}$ be an L^2 function. Then the Fourier coefficients of f are defined as follows:

$$a = \frac{\langle f, 1\rangle}{2\pi} = \frac{1}{2\pi}\int_{[-\pi,\pi]} f\, dm, \tag{B.44}$$

$$b_n = \frac{\langle f, \cos nx\rangle}{\pi} = \frac{1}{\pi}\int_{[-\pi,\pi]} f(x)\cos nx\, dm(x), \tag{B.45}$$

$$c_n = \frac{\langle f, \sin nx\rangle}{\pi} = \frac{1}{\pi}\int_{[-\pi,\pi]} f(x)\sin nx\, dm(x). \tag{B.46}$$

Note that the Fourier coefficients are the coefficients for the functions

$$1, \quad \cos x, \quad \sin x, \quad \cos 2x, \quad \sin 2x, \quad \cos 3x, \quad \sin 3x, \ldots, \tag{B.47}$$

which are not unit vectors. The actual coefficients of the Hilbert basis vectors are

$$a\sqrt{2\pi}, \quad \{b_n\sqrt{\pi}\}, \quad \text{and} \quad \{c_n\sqrt{\pi}\}. \tag{B.48}$$

Corollary 5 (Riesz-Fischer Theorem)**.** *Let $f : [-\pi, \pi] \to \mathbb{R}$ be an L^2 function with Fourier coefficients $a, \{b_n\}, \{c_n\}$. Then $\{b_n\}$ and $\{c_n\}$ are ℓ^2 sequences, and the Fourier series*

$$a + \sum_{n=1}^{\infty} (b_n \cos nx + c_n \sin nx) \tag{B.49}$$

converges to f in L^2.

Proof. This follows from Theorem 14 and the coefficient formula (Corollary 7). □

Corollary 6 (Parseval's Theorem)**.** *Let $f : [-\pi, \pi] \to \mathbb{R}$ be an L^2 function with Fourier coefficients $a, \{b_n\}, \{c_n\}$, and let $g : [-\pi, \pi] \to \mathbb{R}$ be an L^2 function with Fourier coefficients $A, \{B_n\}, \{C_n\}$. Then*

$$\frac{1}{\pi}\int_{[-\pi,\pi]} fg\, dm = 2aA + \sum_{n=1}^{\infty} (b_n B_n + c_n C_n)\,. \tag{B.50}$$

Proof. By the inner product formula (Proposition 5), we have

$$\langle f, g\rangle = (a\sqrt{2\pi})(A\sqrt{2\pi}) + \sum_{n=1}^{\infty} \left((b_n\sqrt{\pi})(B_n\sqrt{\pi}) + (c_n\sqrt{\pi})(C_n\sqrt{\pi})\right), \tag{B.51}$$

and dividing through by π gives the desired formula. □

In the case where $g = f$, this theorem yields Parseval's identity:

$$\frac{1}{\pi}\int_{[-\pi,\pi]} f^2\, dm = 2a^2 + \sum_{n=1}^{\infty}\left(b_n^2 + c_n^2\right). \tag{B.52}$$

Corollary 7. *If $a < b$, then $L^2([a,b])$ and ℓ^2 are isometrically isomorphic.*

Proof. Since

$$\frac{1}{\sqrt{2\pi}},\quad \frac{\cos x}{\sqrt{\pi}},\quad \frac{\sin x}{\sqrt{\pi}},\quad \frac{\cos 2x}{\sqrt{\pi}},\quad \frac{\sin 2x}{\sqrt{\pi}},\quad \frac{\cos 3x}{\sqrt{\pi}},\quad \frac{\sin 3x}{\sqrt{\pi}},\quad \cdots \tag{B.53}$$

is a Hilbert basis for $L^2([-\pi,\pi])$, it follows from Proposition 11 that the linear transformation $T : \ell^2 \to L^2([-\pi,\pi])$ defined by

$$T(a_1, a_2, a_3, \ldots) = \frac{a_1}{\sqrt{2\pi}} + \frac{a_2\cos x}{\sqrt{\pi}} + \frac{a_3 \sin x}{\sqrt{\pi}} + \frac{a_4 \cos 2x}{\sqrt{\pi}} + \frac{a_5 \sin 2x}{\sqrt{\pi}} + \cdots \tag{B.54}$$

is an isometric isomorphism. □

Other Orthogonal Systems

The Fourier basis is not the only Hilbert basis for $L^2([a,b])$. Indeed, many such families of orthogonal functions are known. In this section, we derive an orthonormal sequence of polynomials that is a Hilbert basis for $L^2([a,b])$.

Consider the sequence of functions

$$1,\quad x,\quad x^2,\quad x^3, \ldots \tag{B.55}$$

on the interval $[-1,1]$. These functions are not a Hilbert basis for $L^2([-1,1])$, since they are not orthonormal. However, it is possible to use these functions to make a Hilbert basis of polynomials via the Gram-Schmidt process. We start by making the constant function 1 into a unit vector:

$$p_0(x) = \frac{1}{\|1\|_2} = \frac{1}{\sqrt{2}}. \tag{B.56}$$

The function x is already orthogonal to p_0 on the interval $[-1,1]$, so we normalize x as well:

$$p_1(x) = \frac{x}{\|x\|_2} = x\sqrt{\frac{3}{2}}. \tag{B.57}$$

Now we want a quadratic polynomial orthogonal to p_0 and p_1. The function x^2 is already orthogonal to p_1, but not to p_0. However, if we subtract from x^2 the projection of x^2 onto p_0, then we get a quadratic polynomial orthogonal to p_0:

$$x^2 - \langle p_0, x^2 \rangle p_0(x) = x^2 - \frac{1}{3}. \tag{B.58}$$

Normalizing gives:

$$p_2(x) = \frac{3\sqrt{5}}{2\sqrt{2}} \left(x^2 - \frac{1}{3} \right) \tag{B.59}$$

Continuing in this fashion, we obtain an orthonormal sequence $\{p_n\}$ of polynomials, where each p_n is obtained from x^n by subtracting the projections of x^n onto $p_0, \ldots, p_{n-1}$ and then normalizing.

Definition: Legendre Polynomials

The normalized Legendre polynomials are the sequence of polynomial functions $p_n : [-1, 1] \to \mathbb{R}$ defined recursively by $p_0(x) = 1/\sqrt{2}$ and

$$p_n(x) = c_n \left(x^n - \sum_{k=0}^{n-1} \langle p_k, x^n \rangle p_k(x) \right) \tag{B.60}$$

for $n \geq 1$, where the constant $c_n > 0$ is chosen so that $\|p_n\|_2 = 1$.

By design, each normalized Legendre polynomial $p_n(x)$ has degree n, and the sequence $\{p_n\}_{n \geq 0}$ is orthonormal. The next few such polynomials are

$$p_3(x) = \frac{5\sqrt{7}}{2\sqrt{2}} \left(x^3 - \frac{3}{5} x \right), \quad p_4(x) = \frac{105}{8\sqrt{2}} \left(x^4 - \frac{6}{7} x^2 + \frac{3}{35} \right), \quad \ldots \tag{B.61}$$

Theorem 4. *The sequence $p_0, p_1, p_2, \ldots$ of normalized Legendre polynomials is a Hilbert basis for $L^2([-1, 1])$.*

Proof. Let S be the linear span of $p_0, p_1, p_2, \ldots$. Since

$$x^n = \frac{p_n(x)}{c_n} + \sum_{k=0}^{n-1} \langle p_k, x^n \rangle p_k(x), \tag{B.62}$$

the subspace S contains each x^n, and hence contains all polynomials. By the Weierstrass approximation theorem, every continuous function on $[-1, 1]$ is a uniform limit (and hence an L^2 limit) of a sequence of polynomials. It follows that the closure of S contains all the

continuous functions, and hence contains all L^2 functions by Theorem 13. □

Thus every L^2 function f on $[-1, 1]$ can be written as the sum of an infinite Legendre series

$$f = \sum_{n=0}^{\infty} \langle p_n, f \rangle p_n. \tag{B.63}$$

These behave much like Fourier series, with analogs of Parseval's theorem and Parseval's identity.

Legendre polynomials are important in partial differential equations. For the following definition, recall that a harmonic function on a closed region in $\mathbb{R}^3$ is any continuous function that satisfies Laplace's equation $\nabla^2 f = 0$ on the interior of the region.

Definition: Dirichlet Problem on a Ball

Let B^3 denote the closed unit ball on $\mathbb{R}^3$, and let S^2 denote the unit sphere. The Dirichlet problem on B^3 can be stated as follows: Given a continuous function $f : S^2 \to \mathbb{R}$, find a harmonic function $F : B^3 \to \mathbb{R}$ that agrees with f on S^2.

Since we are working on the ball, it makes sense to use spherical coordinates (ρ, θ, ϕ), which are defined by the formulas

$$x = \rho \cos\theta \sin\phi, \quad y = \rho \sin\theta \sin\phi, \quad z = \rho \cos\phi. \tag{B.64}$$

Using spherical coordinates, one family of solutions to Laplace's equation on the ball can be written as follows:

$$F(\rho, \theta, \phi) = \rho^n p_n(\cos\phi) \tag{B.65}$$

where p_n is the nth Legendre polynomial. These solutions are all axially symmetric around the z-axis, meaning that they have no explicit dependence on θ.

Since the Legendre polynomials are a Hilbert basis, we can use these solutions to solve the Dirichlet problem for any axially symmetric function $f : S^2 \to \mathbb{R}$. All we do is write f as the sum of a Legendre series

$$f(\theta, \phi) = \sum_{n=0}^{\infty} a_n p_n(\cos\phi), \tag{B.66}$$

and then the corresponding harmonic function F will be defined by the formula

$$F(\rho, \theta, \phi) = \sum_{n=0}^{\infty} a_n \rho^n p_n(\cos \phi). \tag{B.67}$$

The functions $p_n(\cos \phi)$ on the unit sphere can be generalized to the family of spherical harmonics $Y_{\ell,m}(\theta, \phi)$, which are a Hilbert basis for $L^2(S^2)$. The Legendre polynomials defined above correspond to the $m = 0$ case:

$$Y_{\ell,0}(\theta, \phi) = \frac{1}{\sqrt{2\pi}} p_\ell(\cos \phi). \tag{B.68}$$

Every L^2 function f on the sphere has a Fourier decomposition in terms of spherical harmonics:

$$f(\theta, \phi) = \sum_{\ell=0}^{\infty} \sum_{m=-\ell}^{\ell} a_{\ell,m} Y_{\ell,m}(\theta, \phi). \tag{B.69}$$

In quantum mechanics, these spherical harmonics give rise to the eigenfunctions of the square of the angular momentum operator. These are known as atomic orbitals, and can be used to describe the quantum wave functions of electrons in an atom.