FRI

UNIVERSITY OF LJUBLJANA | Faculty of Computer and Information Science

Quantum machine learning

by

Bojan Žunkovič

Faculty of computer and information science

University of Ljubljana

December 8, 2025

Contents

1 Approaches to QML

We distinguish four subareas of quantum machine learning and adopt the typology based on whether we obtain the data by a classical (C) or quantum device (Q) and if we process it with a classical (C) or quantum (Q) device/algorithm (shown in Fig. 1.1).

In the case of classical data and devices, we consider approaches that attempt to solve problems in one area using tools developed in another area. Typical examples include applying neural networks to describe complex quantum systems and utilising tensor networks for various machine learning tasks, such as classification, anomaly detection, data embedding, language modelling, and generative modelling. Another important research direction is also the development of quantum-inspired classical algorithms. A prominent example is the dequantization of quantum recommender systems.

In the case of classical data and quantum devices/algorithms, the primary goal is to replace the computationally intensive part of a classical algorithm with a more efficient quantum algorithm. The "hard"/"efficient" might concern computation, generalisation or robustness. The most prominent quantum "algorithm" is a variational quantum circuit, a quantum version of a neural network.

In the case of quantum data and classical algorithms, we typically use machine learning to assist quantum experiments. One possibility is to process the data from quantum measurements with standard machine learning tools. Another use of machine learning in quantum experiments is device calibration, a tedious, error-prone, and critical task.

The last case of quantum data and a quantum algorithm is relatively unexplored due to the lack of quantum control and relatively short coherence times of current quantum devices. One possible application in this area could be quantum cryptography.

Figure 1.1: Four approaches to quantum computing based on the data source and the algorithm used to process the data.

We will explore how quantum computers could help with classical machine learning tasks (the CQ

case). We will also mention some quantum-inspired classical algorithms (the CC case).

References

- Schuld, M., & Petruccione, F. (2021). Machine learning with quantum computers (Vol. 676, pp. 163-169). Berlin: Springer. (Chapter 1: Introduction)

2 Classical and quantum probability

To understand what quantum computing might mean for machine learning, we must distinguish between and understand the differences between classical and quantum data, as well as classical and quantum processing devices. We will begin by contrasting classical probability with quantum probability. Our approach will be less rigorous and formal, and we intend to provide as much intuition about quantum probability as possible, drawing on our knowledge and experience with classical probability theory. We introduce the main concepts of classical probability and then extend them to the quantum case.

2.1 Classical probability

For our purposes, it will be sufficient to define the probability in terms of Kolmogorov axioms summarised as follows: The probability space (Ω, F, P) is a space with a sample space Ω, event space F and a suitably normalised measure $P(\Omega) = 1$. If the sample space is finite, the probability P is a finite set of positive real numbers summing to one.

There are several interpretations of probability, such as frequencies, propensities, degrees of freedom, and plausibility. Here, we will take an operational approach and not discuss the issue of interpretation, as it does not affect the properties we will discuss in the following sections. The quantum probability inherits a similar interpretation problem, which is even more complicated due to the non-intuitive character of quantum probability.

We will use the following notation

$$
\begin{aligned}
X &- \text{random variable,} \\
x_i &- \text{simple event or a value of the random variable,} \\
N &- \text{number of simple events/values of random variable } X, \\
P(x_i) = p_i &- \text{probability for the outcome,} \\
p_i > 0 &- \text{positivity,} \\
|\vec{p}|_1 = \sum_{i=1}^{N} p_i = 1 &- \text{normalization,}
\end{aligned}
$$

Since we will restrict ourselves to the finite-dimensional setting, we can represent the probability as a vector $\vec{p} = (p_1, p_2, \ldots, p_j)$, where p_j represents probabilities for simple events. In this case, we

can represent random variables as a vector where we assume each event is associated with a real number pertaining to the random variable. We represent simple events as diagonal matrices with only one element equal to 1. The representation of a compound event (union of simple events) is equal to the sum of the projectors of simple events. The projection of a probability vector with an event projector is an unnormalized probability vector. Its L_1 norm determines the probability for the event, and its normalised version determines the probability vector after we observe the event. We call the change of the probability vector once we observe an event as the collapse of probability. In an observation/experiment, one of the considered events has to occur. Consequently, the projectors of all considered events should sum to identity $\sum_j \Pi_{e_j} = \mathbb{1}_{\mathbb{N}}$. With $\mathbb{1}_N$, we will denote an N dimensional identity matrix. The last requirement implies that the event probabilities sum to one.

We describe more than one random variable by the joint probability. In the case of two random variables X and Y described by the probabilities P_1 and P_2 there is joint probability $P_12(X = x_i, Y = y_j) = p_{ij}^{12}$. If X has N events and Y has M events, the joint probability is a set of NM positive numbers summing to one. In general, the joint probability p_{ij}^{12} is not a product of probabilities p_i^1 and p_j^2. If we can write the joint probability of K random variables as

$$P_{ij...k}^{12...K} = p_i^1 p_j^2 \ldots p_k^K \tag{2.1}$$

we say that the random variables are independent. Otherwise, they are dependent. We obtain the marginal distribution of a random variable by summing over all remaining random variables

$$p_i^1 = \sum_{j...k} p_{ij...k}^{12...K}. \tag{2.2}$$

Example 2.1. Imagine we have two coins. The first (coin A) is unbiased, and the second (coin B) has a 3/4 probability of getting heads. We first throw coin A. If we observe a head, we throw coin A again; if we get tails, we throw coin B. We want to describe the probability vector during the game. Let us denote simple events as follows $e_0 = (heads, heads), e_1 = (heads, tails), e_2 = (tails, heads), e_3 = (tails, tails)$. For the described game the probability vector is given by $\vec{p} = (\frac{1}{4}, \frac{1}{4}, \frac{3}{8}, \frac{1}{8})$ and the projectors for the simple events are

$$\Pi_{e_0} = \text{diag}(1, 0, 0, 0),$$
$$\Pi_{e_1} = \text{diag}(0, 1, 0, 0),$$
$$\Pi_{e_2} = \text{diag}(0, 0, 1, 0),$$
$$\Pi_{e_3} = \text{diag}(0, 0, 0, 1),$$

the diag denotes a diagonal matrix with provided elements on the diagonal. Now, imagine we throw the first coin. We formally write the compound events as $e_0 \cup e_1$ and $e_2 \cup e_3$. Their projectors sum to identity and are given by

$$\Pi_{e_0 \cup e_1} = \text{diag}(1, 1, 0, 0),$$
$$\Pi_{e_2 \cup e_3} = \text{diag}(0, 0, 1, 1).$$

The unnormalised projections of the probability vector with the event projectors are

$$\vec{p}_{\mathrm{H}} = \vec{p} \cdot \Pi_{e_0 \cup e_1} = \mathrm{diag}(\frac{1}{4}, \frac{1}{4}, 0, 0), \quad p(e_0 \cup e_1) = |\vec{p}_{\mathrm{H}}|_1 = \frac{1}{2},$$

$$\vec{p}_{\mathrm{T}} = \vec{p} \cdot \Pi_{e_2 \cup e_3} = \mathrm{diag}(0, 0, \frac{3}{8}, \frac{1}{8}), \quad p(e_0 \cup e_1) = |p_{\mathrm{T}}|_1 = \frac{1}{2}.$$

As expected, both events are equally likely. If we observe heads, the probability vector is updated to

$$\vec{p} \rightarrow \vec{p}_{\mathrm{H}}/|\vec{p}_{\mathrm{H}}|_1 = \mathrm{diag}(\frac{1}{2}, \frac{1}{2}, 0, 0),$$

and if we observe tails, the probability vector becomes

$$\vec{p} \rightarrow \vec{p}_{\mathrm{T}}/|\vec{p}_{\mathrm{T}}|_1 = \mathrm{diag}(0, 0, \frac{3}{4}, \frac{1}{4}).$$

We describe the marginal probabilities for the second throw to get heads or tails by the following marginal probability vectors

$$\text{we observed heads} \rightarrow \vec{p}_{\mathrm{H}}^2 = (\frac{1}{2}, \frac{1}{2}),$$

$$\text{we observed tails} \rightarrow \vec{p}_{\mathrm{T}}^2 = (\frac{3}{4}, \frac{1}{4}).$$

The outcome of the first throw determines the marginal probability vector for the second throw. However, we need to know which outcome occurred when throwing either coin A or coin B. A similar situation will appear in quantum mechanics, only that the collapse of the probability will be independent of the knowledge of the first observation, which is a distinct feature of quantum probabilities.

Probability simplex In the following, we will not be strict and sometimes use probability to denote a probability vector, not its L_1 norm. Let us now consider the space of all possible probabilities with N outcomes. We define pure probabilities (or pure probability vectors) as probabilities with only one specific simple event. The space of simple events is thus isomorphic to the space of pure probabilities. We can obtain any other probability as a convex combination of pure probabilities. The space of all possible probabilities with N outcomes is a simplex. In Fig. 2.1, we show the probability simplex for the cases $N = 2$ and $N = 3$.

We want to compare the probabilities and find simple functionals describing their properties. The most common concepts are majorisation, Shanon entropy, relative entropy, and Fisher-Rao metric. We shall introduce them in the following.

Majorisation The simplest concept with which we can compare two probability distributions is majorisation. Consider two vectors with N real elements $\vec{x}$ and $\vec{y}$. The vector $\vec{x}$ majorises the
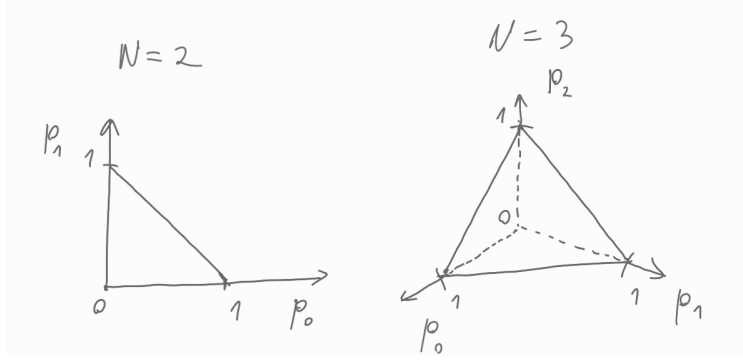
Figure 2.1: The space of all probabilities. In the case of $N = 2$ (left), it is a line; in the case of $n = 3$, it is a triangle.

vector $\vec{y}$ (written as $\vec{x} \succeq \vec{y}$) iff

$$\sum_{i=1}^{k} x_i^{\downarrow} \geq \sum_{i=1}^{k} y_i^{\downarrow}, \quad \text{for all} \quad k = 1, 2, \ldots N. \tag{2.3}$$

The vectors $x^{\downarrow}$ and $x^{\downarrow}$ are such permutations of the original vectors that the entries appear in decreasing order (e.g. $x_1^{\downarrow} \geq x_1^{\downarrow} \geq \ldots \geq x_N^{\downarrow}$). Majorisation is transitive, i.e. $\vec{x} \succeq \vec{y}$ and $\vec{y} \succeq \vec{z}$ implies $\vec{x} \succeq \vec{z}$. All pure probabilities majorise all probabilities. Similarly, the uniform probability is majorised by all probabilities. Therefore, we can think of majorisation as determining which probability is more uniform. There can be probabilities that are unrelated by majorisation as shown in Fig. 2.2 for the case $N = 3$.
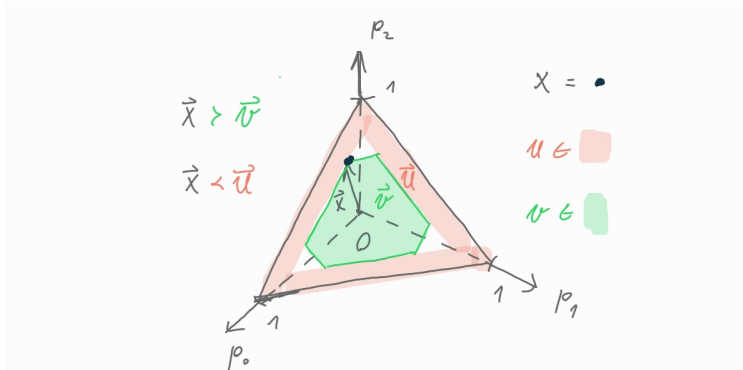
Figure 2.2: Majorisation relation in the $N = 3$ probability simplex. The vectors in the red region majorise the vector $\vec{x}$, and the vectors in the green region are majorised by the vector $\vec{x}$.

Example 2.2. Let $\vec{x} = (4, 1, 1)$, $\vec{y} = (2, 1, 2)$ and $\vec{z} = (3, 1, 3)$

k	$\sum_{i=1}^{k} x_i^{\downarrow}$	$\sum_{i=1}^{k} y_i^{\downarrow}$	$\sum_{i=1}^{k} z_i^{\downarrow}$
1	4	2	3
2	5	4	6
3	6	5	7

We have $\vec{x} \succeq \vec{y}$ and $\vec{z} \succeq \vec{y}$, but there is no majorisation relation between $\vec{x}$ and $\vec{z}$.

Shanon entropy   The most well-known probability functional is the Shanon entropy defined as

$$S(P) = -\sum_{i=1}^{N} p_i \log(p_i). \tag{2.4}$$

Although the entropy is associated with a random variable, we write $S = S(P)$ since the probability is the only property we need from the random variable. We interpret and use the Shanon entropy in many different ways. We can interpret it as the uncertainty about an outcome according to a known probability distribution P. It can quantify the amount of information we need to specify an outcome. It can quantify the compression limits of a long string. It can quantify the probabilities of long sequences of independent identically distributed (IID) random variables. And more. In the following, we provide three examples that elucidate the use of Shanon entropy in different contexts.

Example 2.3. First, we provide without a proof (which is simple) a fundamental theorem, namely the asymptotic equipartition property (AEP) theorem. If $X_1, X_2, \ldots$ are IID with $P(X)$, then

$$-\frac{1}{n} \log p(X_1, X_2, \ldots X_n) \to S(P), \quad \text{in probability.} \tag{2.5}$$

Defining the typical set as a set of sequences $x_1, x_2, \ldots x_n$ such that

$$2^{-n(S(P)+\epsilon)} \le p(x_1, x_2, \ldots x_n) \le 2^{-n(S(P)-\epsilon)} \tag{2.6}$$

we have, as a consequence of the AEP, the following properties. The typical set has a probability close to 1, all elements of the typical set are almost equally likely, and the number of elements in the typical set grows as $\exp(n\, S(p))$.

Example 2.4. Imagine we want to find the smallest expected number of binary questions to determine an event if we know the underlying probability P. For concreteness, we consider that we have four events associated with the probability vector $\vec{p} = (\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8})$. Our task is to find the sequence of YES/NO questions that result in the shortest average number of questions needed to determine the event. In our case, we obtain the shortest average number of questions by the following sequence: 1) "Is the event e_0?", 2) if NO "Is the event e_1?", 3) if NO, "Is the event e_3?" We calculate the expected number of questions as follows

$$L = 1p(e_0) + 2p(e_1) + 3p(e_2) + 3 * p(e_3) = \frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \frac{3}{8} = 1\frac{3}{4}$$

It is not difficult to see that the answer is the Shanon entropy $S(p)$ if calculated with the logarithm of base 2. Therefore, we can regard the $-\log(p_i)$ of the probability of simple events as the amount of information we obtain by measuring the event if the underlying probability distribution is P.
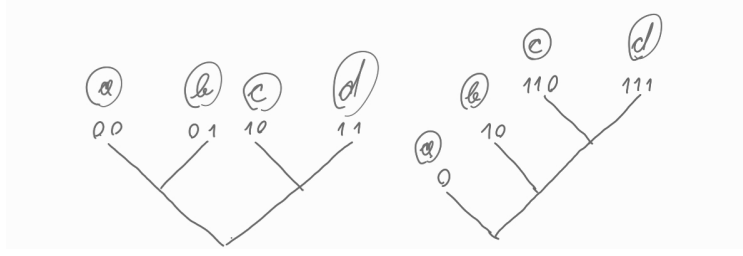
Figure 2.3: Binary code trees for four characters. The right tree has a shorter expected length if $p_1 > p_2 + p_3$.

Example 2.5. Let us now consider the problem of compression. We assume that we have a long text where the four possible characters "a", "b", "c", and "d" appear according to the probability $\vec{p} = (\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8})$. Our task is to specify a binary prefix code so that the expected length of the text will be the shortest. We calculate the expected length of the code as

$$L = l(a)p(a) + l(c)p(c) + l(b)p(b) + l(d)p(d). \tag{2.7}$$

Let us consider the codes given by the left and the right code tree in Fig. 2.3. In the left case, we get

$$L_{\text{left}} = 2\frac{1}{2} + 2\frac{1}{4} + 2\frac{1}{8} + 2\frac{1}{8} = 2$$

and in the right case, we get

$$L_{\text{right}} = \frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \frac{3}{8} = 1\frac{3}{4}.$$

One can show that the right case is the optimal one. Also, the Shanon entropy again gives the shortest expected length, which is, in this context, related to data compression.

Relative entropy   The second important probability functional that we will discuss is the relative entropy between two random variables X and Y with probabilities $P(X)$ and $Q(Y)$ defined as

$$S(P||Q) = \sum_{i=1}^{N} p_i \log\left(\frac{p_i}{q_i}\right). \tag{2.8}$$

Relative entropy also has many applications and interpretations. It describes the convergence of samples to the asymptotic probability. We can also use it as an inefficiency measure in our compression, question strategy,..., if assuming the wrong distributions. We further elaborate on both meanings with some examples

Example 2.6. Sanov theorem (informal): Assume we are drawing $\mathcal{N}$ samples from a probability distribution Q. The probability to get the frequency count P^* is close asymptotically given

by

$$\mathcal{P} \sim e^{-\mathcal{N}S(P^*||Q)}. \tag{2.9}$$

Example 2.7. Let us continue with the code trees example from the discussion of the Shanon entropy (Example. 2.5) and assume that we think all characters are equally likely. Namely, we assume that the probability of the characters is $q = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ whereas the real probability is still $\vec{p} = (\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8})$. The best encoding for our assumed probability q is the left code tree in Fig. 2.3. The difference between the assumed optimal length and the real optimal length (L_{left} - L_{right} from the example Example. 2.5) is equal to the relative entropy

$$S(p||q) = \frac{1}{2}\log(\frac{4}{2}) + \frac{1}{4}\log(\frac{4}{4}) + \frac{1}{8}\log(\frac{4}{8}) + \frac{1}{8}\log(\frac{4}{8}) = \frac{1}{2} + 0 - \frac{1}{8} - \frac{1}{8} = \frac{1}{4}$$

Fisher-Rao metric The Relative entropy determines a probabilistic "distance" between two probabilities. However, it is not a true distance in a mathematical sense since it is not symmetric. If we expand the relative entropy around the first or the second argument, we get

$$S(P||P + dP) \approx \sum_{i=1}^{N} p_i \log\left(\frac{p_i}{p_i + dp_i}\right) = \frac{1}{2}\sum_{i=1}^{N}\frac{dp^i dp^i}{p^i}. \tag{2.10}$$

We get the same result if we expand $S(P + dP||P)$. The infinitesimal relative entropy is symmetric. The metric determining the infinitesimal distance between probability distributions is the Fisher-Rao metric defined as

$$g_{ij} = \frac{1}{4}\frac{\delta_{ij}}{p_i}. \tag{2.11}$$

We can introduce new coordinates to make the metric flat

$$X^i = \sqrt{p^i} \Rightarrow dX^i = \frac{dp^i}{\sqrt{p^i}}. \tag{2.12}$$

It follows that

$$ds^2 = \frac{1}{4}\sum_{i=1}^{N}\frac{dp^i dp^i}{p^i} = \sum_{i=1}^{N} dX^i dX^i. \tag{2.13}$$

In the new flat space, the ellipsoids around the probability vector become circles, and the normalisation condition becomes $|\vec{p}|_1 = |\vec{X}|_2 = 1$. In the newly introduced coordinates with a flat metric, the probabilities live on a positive part of a sphere. The geodesic distance between probabilities is the angle between the square roots of the probabilities (called the Battacharayya angle θ_{B}) determined by the scalar product between square roots of the probabilities

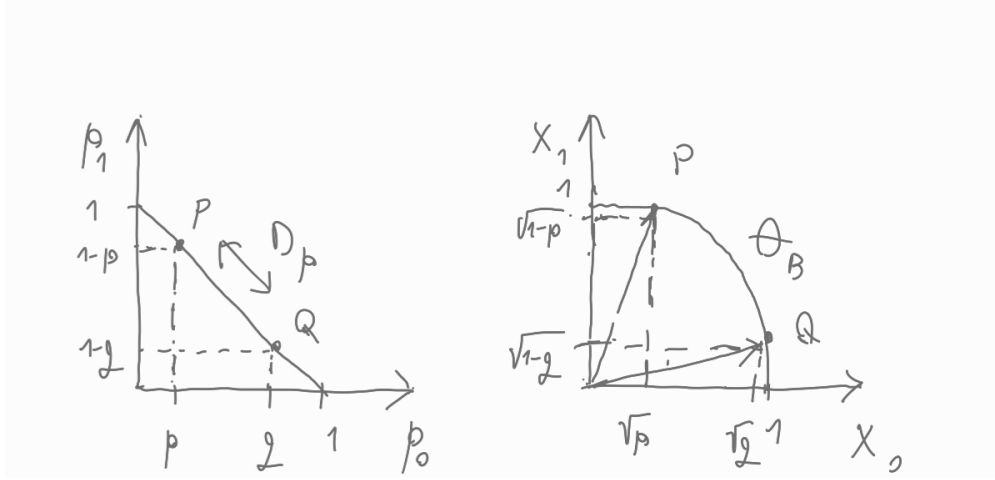$$\cos\theta_{\text{B}} = \sum_{i=1}^{N}\sqrt{p^i q^i}. \tag{2.14}$$

Figure 2.4: The left plot represents the standard probability simplex for $N = 2$ and the L_1 distance between the probabilities P and Q. The right plot shows the deformed simplex in new coordinates with a flat Fisher-Rao metric and a geometric distance between the probabilities given by the angle between the square root probability vectors.

The Fig. 2.4 provides a simple visualization in the case $N = 2$. The Fisher-Rao metric also has a notable operational and practical meaning. To see that, we first consider a submanifold of the probability space parametrised by coordinates θ^i. The Fisher-Rao metric induces a metric in this manifold as

$$g_{ab} = \sum_{i,j=1}^{N} g_{ij} \frac{\partial p^i}{\partial \theta^a} \frac{\partial p^j}{\partial \theta^b}. \tag{2.15}$$

The above metric provides a fundamental constraint on the estimation uncertainty of the parameters θ^i from the samples of some random variable. If we assume that we have an unbiased estimator ξ^i, i.e. $\langle \xi^i \rangle = \theta^i$, we have the following covariance bound

$$\langle \xi^a \xi^b \rangle - \langle \xi^a \rangle \langle \xi^b \rangle \geq \frac{1}{4} g_{ab}. \tag{2.16}$$

More pragmatically, the Fisher-Rao metric also defines the natural gradient. If efficiently calculable, the natural gradient improves the training with stochastic gradient descent since it considers the curvature of the probability space.

Here, we conclude our short introduction to classical probability. Introduced concepts will be a starting point for discussing quantum probability in the following sections.

References

- Bengtsson, I., & Życzkowski, K. (2017). Geometry of quantum states: an introduction to quantum entanglement. Cambridge University Press. (Chapter 2: Geometry of probability distributions – 2.1, 2.2, 2.3, 2.4, 2.5)

- Cover, T. M. (2006 - second edition). Elements of information theory. John Wiley & Sons. (Chapter 2: Entropy, relative entropy and mutual information)

2.2 Quantum probability

This section introduces quantum probability with minimal changes to the classical probability formalism. The goal is to preserve most of the classical intuition/interpretation described in the previous section. We will introduce quantum probability in two steps:

- Extension of the formalism for classical probability from vectors to (diagonal) matrices

- Removing the restriction to diagonal matrices

The following table serves as a dictionary of the extended formalism of classical probability described in the previous section. We describe the probability in the extended formalism by a diagonal matrix and not by a vector.

Object	Classical (vector)	Extended (matrix)	Example ($N = 2$)
Probability	$\vec{p} = (p_1, p_2, \ldots p_N)$	$\rho = \mathrm{diag}(p_1, p_2, \ldots p_N)$	$\rho = \begin{pmatrix} p_1 & 0 \\ 0 & p_2 \end{pmatrix}$
random variables	$\vec{a} = (a_1, a_2, \ldots a_N)$	$A = \mathrm{diag}(a_1, a_2, \ldots a_N)$	$A = \begin{pmatrix} a_1 & 0 \\ 0 & a_2 \end{pmatrix}$
simple events	$\Pi_j = \mathrm{diag}(0_1, \ldots 1_j \ldots 0_N)$	Π_j (remains the same)	$\Pi_1 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$
real probabilities	$p_j \in \mathbb{R}$	$\rho^\dagger = \rho$	$p_1 = p_1^*, \, p_2 = p_2^*$
real random variables	$a_j \in \mathbb{R}$	$A^\dagger = A$	$a_1 = a_1^*, \, a_2 = a_2^*$
positivity	$p_j > 0$	$\rho > 0$	$p_1 > 0, \, p_2 > 0$
normalisation	$\sum_{i=1}^{N} p_i = 1$	$\mathrm{Tr}\, \rho = 1$	$p_1 + p_2 = 1$
sum of simple events	$\sum_{i=1}^{N} \Pi_i = \mathbb{1}_N$	$\sum_{i=1}^{N} \Pi_i = \mathbb{1}_N$	$\Pi_1 + \Pi_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
event probability	$p_j = |\vec{p} \cdot \Pi_j|_1$	$p_j = \mathrm{Tr}\,(\Pi_j \rho \Pi_j)$	$\mathrm{Tr}\,(\Pi_1 \rho \Pi_1) = p_1$
averages	$\langle a \rangle = \sum_{ij=1}^{N} p_i a_i b_i$	$\langle A \rangle = \mathrm{Tr}\, \rho A$	$\mathrm{Tr}\, \rho A = a_1 p_1 + a_2 p_2$
Independent P and Q	$\vec{p}^{12} = \vec{p}^1 \otimes \vec{q}^2$	$\rho^{12} = \rho^1 \otimes \rho^2$	$\rho^{12} = \begin{pmatrix} p_1 q_1 & 0 & 0 & 0 \\ 0 & p_1 q_2 & 0 & 0 \\ 0 & 0 & p_2 q_1 & 0 \\ 0 & 0 & 0 & p_2 q_2 \end{pmatrix}$
marginals	$p_i^1 = \sum_{j=1}^{N} p_{ij}^{12}$	$\rho^1 = \mathrm{Tr}_2\, \rho$	$\rho^1 = \mathrm{Tr}_2 \rho^{12} = \begin{pmatrix} p_1 & 0 \\ 0 & p_2 \end{pmatrix}$
prob. update	$\vec{p}_{\Pi_j}^{\mathrm{UN}} = \Pi_j \vec{p}$	$\rho_{\Pi_j}^{\mathrm{UN}} = \Pi_j \rho \Pi_j$	$\rho_{\Pi_1}^{\mathrm{UN}} = \begin{pmatrix} p_1 & 0 \\ 0 & 0 \end{pmatrix}$

With $\vec{p}^{UN}$ and ρ^{UN} we denote the unnormalized probabilities. With the restriction to the diagonal matrices given in the first three rows of the above table, the standard and the extended columns are equivalent. We arrive at quantum probability if we remove this restriction and allow all objects that satisfy the rest of the equations in the extended column. It is standard to call the quantum probability a density matrix. In the following, we shall describe, similarly to the classical case, the essential properties of the quantum probability/density matrix.

Example 2.8. Qubit Let us start with the simplest case $N = 2$ (qubit). In this case, we can parametrise the most general density matrix satisfying the hermiticity and normalisation as

$$\rho = \frac{1}{2} \begin{pmatrix} 1+z & x-\mathrm{i}y \\ x+\mathrm{i}y & 1-z \end{pmatrix}. \tag{2.17}$$

The positivity condition is equivalent to the positivity of the eigenvalues of the above density matrix. After simplification, we obtain

$$x^2 + y^2 + z^2 \leq 1 \tag{2.18}$$

The probabilities to get $a0$ or $a1$ are

$$p_0 = \mathrm{Tr}\,\Pi_0 \rho \Pi_0 = \frac{1}{2}(1+z), \quad \text{and} \quad p_1 = \mathrm{Tr}\,\Pi_1 \rho \Pi_1 = \frac{1}{2}(1-z), \tag{2.19}$$

where

$$\Pi_0 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \quad \text{and} \quad \Pi_1 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}. \tag{2.20}$$

Is the mapping to quantum probability unique? We now consider the following problem. Given a classical probability $\vec{p}$ and a set of "simple" projectors $\{\Pi_j\}$, find a corresponding quantum probability. One possibility is just the diagonal matrix with p_j on the diagonal. However, this is not the only possibility. Any density matrix having the same diagonal will produce the same classical probability for the diagonal projectors.

Example 2.9. In our qubit example, all density matrices of the form

$$\eta = \begin{pmatrix} p_0 & \sqrt{p_0 p_1}\, e^{\mathrm{i}\phi} \\ e^{-\mathrm{i}\phi}\sqrt{p_0 p_1} & p_1 \end{pmatrix}$$

reproduce the same classical probability given by the projectors Π_0, Π_1 defined in Eq. 2.20.

Determines the quantum probability as a unique classical probability? We have seen that many quantum probabilities are consistent with one classical probability and a set of projectors. The converse is also true. A single quantum probability (density matrix) ρ determines many classical probabilities associated with different sets of projectors. Since we do not have the restriction to diagonal matrices, we can transform one set of projectors $\{\Pi_j\}$ into another valid set of projectors by a unitary transformation $\{U\Pi_j U^\dagger\}$. The second set will typically result in different classical probabilities for different unitaries.

Example 2.10. In our qubit example, we define the following set of projectors

$$\Pi_0(\varphi) = \begin{pmatrix} \cos\varphi \\ \sin\varphi \end{pmatrix} \begin{pmatrix} \cos\varphi & \sin\varphi \end{pmatrix} = \begin{pmatrix} \cos\varphi^2 & \cos\varphi\sin\varphi \\ \cos\varphi\sin\varphi & \sin\varphi^2 \end{pmatrix}$$

$$\Pi_1(\varphi) = \begin{pmatrix} -\sin\varphi \\ \cos\varphi \end{pmatrix} \begin{pmatrix} -\sin\varphi & \cos\varphi \end{pmatrix} = \begin{pmatrix} \sin\varphi^2 & -\cos\varphi\sin\varphi \\ -\cos\varphi\sin\varphi & \cos\varphi^2 \end{pmatrix}$$

It is clear that $\Pi_0(\varphi) + \Pi_1(\varphi) = \mathbb{1}_2$. Since we have $||(\cos\varphi, \sin\varphi)||_2 = ||(-\sin\varphi, \cos\varphi)||_2 = 1$ we also have $\Pi_j(\varphi)^2 = \Pi_j(\varphi)$. Finally, we get the following classical probabilities associated with the qubit density matrix ρ from Example. 2.8 and the set of projectors $\{\Pi_j(\varphi)\}$

$$p_0 = \operatorname{Tr}\Pi_0(\varphi)\rho\Pi_0(\varphi) = \frac{1}{2}(1 + z(\cos\varphi - \sin\varphi) + x\cos\varphi\sin\varphi),$$

$$p_1 = \operatorname{Tr}\Pi_1(\varphi)\rho\Pi_1(\varphi) = \frac{1}{2}(1 - z(\cos\varphi - \sin\varphi) - x\cos\varphi\sin\varphi).$$

Pure and mixed states We have seen that we do not have a one-to-one mapping between quantum and classical probabilities. The set of quantum probabilities seems larger than the set of classical probabilities. In the following, we will make the above statement more precise.

The quantum probability (density matrix) is a Hermitian matrix. Hence, we can diagonalise it by a unitary transformation. Due to positivity and normalisation, its eigenvalues are positive real numbers that sum to one. We thus have

$$\rho = U\Lambda U^\dagger, \quad U^\dagger U = \mathbb{1}_N, \quad \Lambda = \operatorname{diag}(\lambda_1, \dots \lambda_N), \quad \lambda_j \geq 0, \quad \sum_{i=1}^{N} \lambda_i = 1 \tag{2.21}$$

We use the convention that the eigenvalues are ordered $\lambda_1 \geq \lambda_2 \geq \dots \lambda_N$. Density matrices with only one non-zero eigenvalue $\lambda_1 = 1$ are called pure quantum probabilities. If the quantum probability is not pure, we call it a mixed quantum probability. We have

$$\rho^2 = \rho, \quad \text{pure quantum probability} \tag{2.22}$$

$$\rho^2 < \rho, \quad \text{mixed quantum probability.} \tag{2.23}$$

Pure quantum probabilities are analogues of pure classical probabilities. Every pure classical probability is also a pure quantum probability.

Pure quantum probabilities are the central object in quantum computation and machine learning. Therefore, we introduce a new notation. We write $\rho_{\text{pure}} = |\psi\rangle\langle\psi|$. We denote the "square root" of pure quantum probability with a *ket* $|\psi\rangle$, which in our case represents a column vector. We represent the row vector a *bra* $\langle\psi|$, where we also take complex conjugation. *Bra* and ket represent adjoint vectors. We can combine them and construct operators (matrices) or scalars (complex numbers). Pure states are related to the eigenvectors of the density matrix with the eigenvalue 1. The averages over pure quantum probabilities can be calculated as $\langle A\rangle_\psi = \operatorname{Tr} A\rho = \langle\psi| A |\psi\rangle$. The normalisation of quantum probabilities implies the L_2 normalisation of the "square root" of pure quantum probabilities, namely $||\psi||_2 = \langle\psi|\psi\rangle = 1$.

To summarise, pure quantum probabilities are vectors in a complex Hilbert space, i.e. $\psi \in \mathcal{H}$, and the random variables (observables) are operators in that space.

In the following, we will use the standard terminology and refer to the "square root" of the pure quantum probability as a quantum state or a pure quantum state. We will imply that we can write it as a matrix or a vector. We will refer to mixed quantum probabilities as mixed quantum states. The term density matrix refers to the matrix representation of a pure or mixed quantum state.

We observe that $\vec{p} \in \mathbb{R}^N, ||\vec{p}||_1 = 1$ while $|\psi\rangle \in \mathbb{C}^N, ||\psi||_2 = 1$. We also see that while there is a finite number of pure classical probabilities, we have infinitely many pure quantum states.

Example 2.11. (Qubit) Let us revisit the qubit example Example. 2.8 and write the density matrix in terms of the ubiquitous Pauli matrices $\sigma^x, \sigma^y, \sigma^z$:

$$\rho = \frac{1}{2}(\mathbb{1} + \vec{\tau} \cdot \vec{\sigma}), \quad \vec{\tau} = (x, y, z), \quad \vec{\sigma} = (\sigma^x, \sigma^y, \sigma^z). \tag{2.24}$$

The qubit is in a pure state iff $||\vec{\tau}||_2 = 1$. In this case, we can write

$$\rho = |\psi\rangle\langle\psi|, \quad |\psi\rangle = \begin{pmatrix} \cos\theta/2 \\ e^{-i\varphi}\sin\theta/2, \end{pmatrix} \tag{2.25}$$

The angles θ and φ parametrise a general pure qubit state. The density matrix of this state is given by

$$\rho = \begin{pmatrix} \cos^2\theta/2 & e^{-i\varphi}\cos\theta/2\sin\theta/2 \\ e^{i\varphi}\cos\theta/2\sin\theta/2 & \sin^2\theta/2, \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 + \cos\theta & \sin\theta(\cos\varphi - i\sin\varphi) \\ \sin\theta(\cos\varphi - i\sin\varphi) & 1 - \cos\theta. \end{pmatrix} \tag{2.26}$$

We can now express the elements of the vector $\vec{\tau}$ in terms of the angles θ and φ

$$x = \sin\theta\cos\varphi, \tag{2.27}$$
$$y = \sin\theta\sin\varphi, \tag{2.28}$$
$$z = \cos\theta. \tag{2.29}$$

As shown in the Fig. 2.5, the $N = p$ probability manifold is a unit ball called the Bloch ball. The pure states live on its surface, known as the Bloch sphere. This simple example illustrates the geometric difference between quantum and classical probability. In the $N = 2$ case, the classical probability is simply a line with pure states being its boundary points. The quantum case is more complicated as all quantum states live in a unit ball, and there are infinitely many pure states.
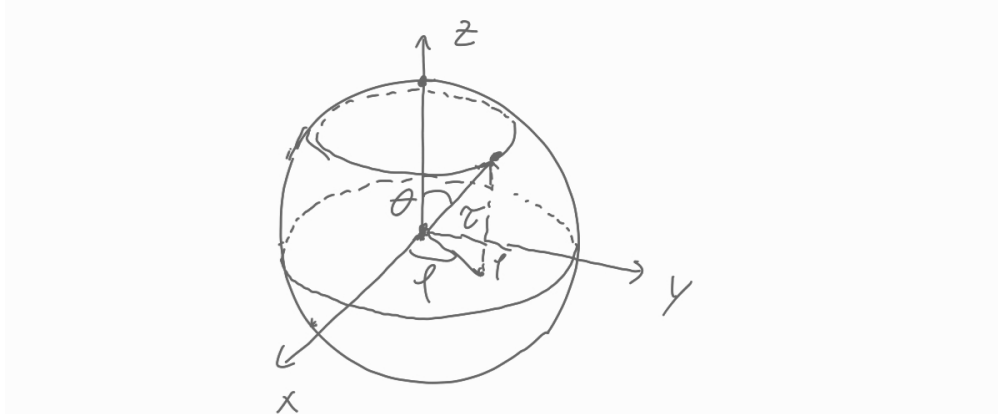
Figure 2.5: We show a representation of the Bloch ball. Pure quantum states live on the surface of the unit ball (Bloch sphere), whereas mixed quantum states live inside the sphere. The uniform distribution is at the origin.

Superposition and $|Schroedinger\rangle$ For a pure quantum state, one can always choose a set of projectors Π_j such that the corresponding classical probability is pure. In this case, the outcome is deterministic, and we know the quantum state exactly. However, in contrast to the classical case, we can also perform observations for any other set of projectors. In most other cases, the obtained classical probability will not

be pure, and the outcome will be random. The ambiguity of classical measurements for a pure quantum state is the origin of the Schrödinger cat thought experiment, where we "evolve" a cat into a state of being alive and dead. Here, we interpret a pure quantum state as a deterministic object, the same way as we can with a pure classical state. We only go this far with interpreting the superposition, which is operationally only a statement that a pure quantum state expressed in a chosen basis can have more than one non-zero component.

Example 2.12. Let us consider the following density matrix

$$\rho = \frac{1}{2} \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}.$$

Together with projectors Eq. 2.20 this defines a uniform classical probability $\vec{p} = (\frac{1}{2}, \frac{1}{2})$. However, we also notice that the density matrix is a pure quantum state given by

$$|\psi\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{2}(|0\rangle + |1\rangle).$$

Since a pure quantum state is also a projector, any set of projectors containing the state will have a deterministic outcome for our chosen state. For example

$$\Pi_0 = \frac{1}{2} \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, \quad \Pi_1 = \frac{1}{2} \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}.$$

The above projectors determine a pure classical probability with $p_0 = 1$ and $p_1 = 0$.

Collapse of the probability   What happens with the quantum state when we observe an event? The answer to this question is the same as in the classical case. If we throw a fair coin initially, the probability is $\vec{p} = \frac{1}{2}(1, 1)$. After we observe either 0 or 1, we update the probability to $\vec{p} = (1, 0)$ or $\vec{p} = (0, 1)$. In the quantum case, it is the same. The quantum states become states we obtain if we first project the observed state onto the projector associated with the outcome and then normalise it. We express this in equations as follows

$$\rho \xrightarrow{\Pi_j} \frac{\Pi_j \rho \Pi_j}{\mathrm{Tr} \Pi_j \rho \Pi_j}. \tag{2.30}$$

Majorisation of quantum probabilities   In the classical case, we introduced the majorisation order with non-increasing permuted probability vectors. As we have seen, a quantum probability ρ determines many classical probabilities through projective measurements $\{\Pi_j\}$. However, we can use a unique probability to generalise majorisation, namely the probability determined by the eigenvectors of the quantum probability ρ. Therefore, we define the majorisation of quantum probabilities by

$$\rho \succeq \sigma \quad \Leftrightarrow \vec{\lambda}(\rho) \succeq \vec{\lambda}(\sigma). \tag{2.31}$$

We can show that the probability vector $\vec{\lambda}(\rho)$ majorises any other probability vector determined by a projective measurement $\{\Pi_j\}$. In other words, we determine the least uniform (most pure) probability by the projectors on the eigenspace of the quantum probability ρ.

Von Neumann entropy   The most important probability functional in the classical case is the Shanon entropy. Its generalisation to the quantum case should recover the same result in the limit of diagonal density
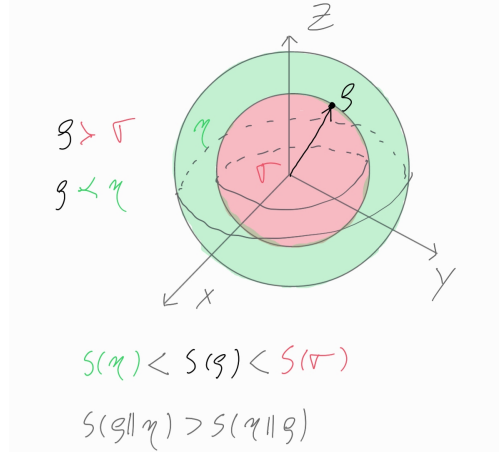
Figure 2.6: Majorization and entropies in a qubit case. The green quantum probabilities majorize the red quantum probabilities. Related inequalities hold for von Neumann entropy and quantum relative entropy.

matrices. Namely

$$S(\rho_c) = \sum_{i=1}^{N} \lambda_i \log \lambda_i, \rho_c = \text{diag}(\lambda_1, \ldots, \lambda_N). \tag{2.32}$$

We can write this expression in a basis-independent form

$$S(\rho) = \text{Tr}\rho \log \rho, \tag{2.33}$$

which is valid for any density matrix. It already represents the correct quantum analogue of the Shanon entropy, called von Neumann entropy. However, this is not the only choice we could make. For example, every complete set of projectors defines a classical probability, which then has an associated Shanon entropy

$$S(\vec{p}) = \sum_{i=1}^{N} p_i \log p_i, \quad p_i = \text{Tr}\,\rho\Pi_i. \tag{2.34}$$

One can show that von Neumann's entropy determines the minimum of the above entropies over all possible measurements. In this sense, the von Neumann entropy determines the minimum amount of uncertainty we can have when observing a quantum state, and it is a measure of purity. This agrees with the generalisation of majorisation discussed above and is explicitly shown in Fig. 2.6.

Quantum relative entropy   Similarly as before, we define the quantum relative entropy by classical analogy

$$D(\rho||\eta) = \text{Tr}\rho(\log \rho - \log \eta). \tag{2.35}$$

The relative entropy quantifies the distinguishability of two quantum states by the best possible observation/measurement.

In the qubit case, we explicitly calculate the relative entropy $S(\rho_a||\rho_b)$ as

$$S(\rho_a||\rho_b) = \frac{1}{2}\ln\left(\frac{1-\tau_a^2}{1-\tau_b^2}\right) + \frac{\tau_a}{2}\ln\left(\frac{1-\tau_a}{1+\tau_a}\right) - +\frac{\tau_a}{2}\ln\left(\frac{1-\tau_b}{1+\tau_b}\right), \tag{2.36}$$

where $\rho_{a,b} = \frac{1}{2}(\mathbb{1} + \vec{\tau}_{a,b} \cdot \vec{\sigma})$ and $\tau_{a,b} = \|\vec{\tau}_{a,b}\|_2$. From the Eq. 2.36 we explicitly see that $S(\rho_a||\rho_b) > S(\rho_b||\rho_a)$ iff $\tau_a < \tau_b$ (see also Fig. 2.6).

Quantum statistical distance (Fisher-Study distance)   Let us consider pure states $|\psi\rangle$ and $|\varphi\rangle$ and a set of projectors $\{\Pi_1, \Pi_2 \ldots \Pi_N\}$. They define two classical probabilities $\vec{p} = (p_1, p_2 \ldots p_N)$ and $\vec{q} = (q_1, q_2 \ldots q_N)$ with

$$p_j = \langle\psi|\,\Pi_j\,|\psi\rangle \quad \text{and} \quad q_j = \langle\varphi|\,\Pi_j\,|\varphi\rangle. \tag{2.37}$$

We also write $\Pi_j = |j\rangle\langle j|$. The geodesic distance between the two classical probabilities $\vec{p}$ and $\vec{q}$ is given by the Battacharayya angle θ_{B}

$$\cos\theta_{\mathrm{B}} = \sum_{i=1}^{N} \sqrt{p_i q_i} = \sum_{i=1}^{N} |\langle\psi|j\rangle|\,|\langle\varphi|j\rangle| \tag{2.38}$$

We want to find the maximum possible distance for all allowed sets of measurements. Therefore, we must minimise the right-hand side of Eq. 2.38. By using the inequality

$$\sum_{i=1}^{N} |\langle\psi|j\rangle|\,|\langle\varphi|j\rangle| \geq |\langle\psi|\varphi\rangle| = \cos\theta_{\mathrm{FS}} \tag{2.39}$$

we obtain the probabilistic distance between two quantum states. The distance between two quantum states is the largest possible distance between two classical probabilities defined by the states and a set of projectors $\{\Pi_j\}$.

Multiple quantum random variables: Entanglement   We will now discuss the properties of quantum probabilities with many random variables. Consider two uncorrelated pure quantum states with states $|\psi_1\rangle$ and $|\psi_2\rangle$. A pure quantum state

$$|\psi\rangle = |\psi_2\rangle \otimes |\psi_2\rangle. \tag{2.40}$$

describes the joint quantum probability of the states

Example 2.13. Let us consider a simple example of a two-qubit state. The first qubit is described by the state $|\psi_1\rangle = |0\rangle$ and the second by the state $|\psi_2\rangle = |1\rangle$. We write the joint probability distribution as $|\psi\rangle = |0\rangle \otimes |1\rangle$ or simply $|\psi\rangle = |01\rangle$. If we choose the basis $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, $|0\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ we can express the quantum states as vectors

$$|\psi_1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |\psi_2\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad \text{and} \quad |\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}. \tag{2.41}$$

However, we can not write all states as a tensor product of two states (as in Eq. 2.40). We call such states entangled states. A principal example of an entangled state is a Bell state

$$\left|\psi^{+}\right\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{2.42}$$

We can express the state in a standard basis as

$$\left|\phi^{+}\right\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}. \tag{2.43}$$

To show that the state $|\phi^+\rangle$ is entangled, we consider the most general two-qubit state that we can write as a tensor product of one qubit states

$$|\psi_1\rangle = \begin{pmatrix} a \\ b \end{pmatrix}, \quad |\psi_1\rangle = \begin{pmatrix} c \\ d \end{pmatrix}. \tag{2.44}$$

We express the joint state in a standard basis as

$$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle = \begin{pmatrix} ac \\ ad \\ bc \\ bd \end{pmatrix}. \tag{2.45}$$

The equation $|\psi\rangle = |\phi\rangle$ shows that neither a, b, c, d can be zero. At the same time, one of a and d and one of b and c should be zero. Consequently, we can not write the state $|\phi^+\rangle$ as a product of two one-qubit states, i.e., the state is entangled. The marginal probabilities of entangled states are mixed states, and the marginal probabilities of non-entangled states are pure states.

Example 2.14. Let us calculate the marginal probabilities of the Bell state $|\phi^+\rangle$. We calculate them by the partial trace operation as

$$\rho = \frac{1}{2} \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}, \quad \rho_1 = \text{Tr}_2\, \rho = \frac{1}{2} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \rho_2 = \text{Tr}_1\, \rho = \frac{1}{2} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{2.46}$$

Schmidt decomposition An important tool to understand entangled states is the Schmidt decomposition

$$|\psi\rangle = \sum_{i=1}^{K} \sqrt{\lambda_i}\, |\varphi_i\rangle\, |\theta_i\rangle, \tag{2.47}$$

where $|\varphi_i\rangle$ and $|\theta_j\rangle$ denote orthonormal bases of the two parts of the quantum state, $\lambda_i > 0$, and $\sum_{i=1}^{K} \lambda_i = 1$.

Proof. In general we can a twopartite quantum state as $|\psi\rangle = \sum_{i,j} C_{i,j} |i\rangle\, |j\rangle$, where $i = 1, \ldots N$ and $j = 1, \ldots M$ and $|i\rangle, |j\rangle$ are orthonormal basis states in the respective subsystems. We can view the coefficients C as an $N \times M$ dimensional complex matrix and expand it via the singular value (SVD) decomposition

$$C = U \text{diag}\left(\sqrt{\lambda_1}, \sqrt{\lambda_2}, \ldots \sqrt{\lambda_K}\right) V, \quad U^\dagger U = V V^\dagger = \mathbb{1}_K, \quad K = \min(N, M). \tag{2.48}$$

Inserting the SVD decomposition in the general expression for the state, we obtain

$$|\psi\rangle = \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{i=1}^{K} U_{ik} \sqrt{\lambda_k} V_{k,j} |i\rangle\, |j\rangle. \tag{2.49}$$

We now define new basis states $|\varphi_i\rangle$ and $|\theta_j\rangle$ as

$$|\varphi_k\rangle = \sum_{i=1}^{N} U_{i,k} |i\rangle, \quad |\theta_k\rangle = \sum_{j=1}^{M} V_{k,j} |j\rangle. \tag{2.50}$$

Finally, we must show that the new basis states $|\varphi_i\rangle$ and $|\theta_j\rangle$ are orthonormal. We see this by using the unitarity of U and V

$$\langle\varphi_{k'}|\varphi_k\rangle = \sum_{i',i=1}^{N} \overline{U}_{i',k'}U_{i,k}\,\langle i'|i\rangle = \sum_{i=1}^{N} \overline{U}_{i,k'}U_{i,k} = \delta_{k',k}, \tag{2.51}$$

$$\langle\theta_{k'}|\theta_k\rangle = \sum_{j',j=1}^{N} \overline{V}_{k',j'}V_{k,j}\,\langle j'|j\rangle = \sum_{j=1}^{M} \overline{V}_{k',j'}V_{k,j} = \delta_{k',k}. \tag{2.52}$$

$\square$

An important consequence of the Schmidt decomposition is that we can write any quantum state (pure or mixed) as a marginal of a larger pure state.

$$\rho_{12} = |\psi\rangle\,\langle\psi| = \sum_{k',k=1}^{K} \sqrt{\lambda_{k'}\lambda_k}\,|\varphi_k\theta_k\rangle\,\langle\varphi_{k'}\theta_{k'}|\,, \tag{2.53}$$

$$\rho_1 = \mathrm{Tr}_2\,\rho_{12} = \sum_{k=1}^{K} \langle\theta_k|\psi\rangle\,\langle\psi|\theta_k\rangle = \sum_{k=1}^{K} \lambda_k\,|\varphi_k\rangle\,\langle\varphi_k|\,. \tag{2.54}$$

Since in the above equations, $|\psi\rangle$ is an arbitrary bipartite quantum state, also $\lambda_k > 0$, and $|\varphi_k\rangle$ can take any allowed values. Hence, any ρ_1 can be written as a marginal of a larger pure state $|\psi\rangle$.

Completeness of the reduced density matrix   We will now consider a probability $\vec{p} = (p_1, p_2, \ldots p_N)$ determined by projectors $\{\Pi_j \otimes \mathbb{1}_M\}$ acting nontrivially only on the first part of the system

$$p_j = \mathrm{Tr}\,(\rho_{12}\Pi_j \otimes \mathbb{1}_M) \tag{2.55}$$

Since the projectors act nontrivially only on the first part of the system, we can write

$$
\begin{aligned}
p_j &= \sum_{i=1}^{N}\sum_{j=1}^{M} \langle i|\,\langle j|\,\rho_{12}\Pi_j \otimes \mathbb{1}_M\,|i\rangle\,|j\rangle \\
&= \sum_{i=1}^{N}\sum_{j=1}^{M} \langle i|\,\langle j|\,\rho_{12}\,|j\rangle\,\Pi_j\,|i\rangle \\
&= \sum_{i=1}^{N} \langle i|\sum_{j=1}^{M} \langle j|\,\rho_{12}\,|j\rangle\,\Pi_j\,|i\rangle \\
&= \sum_{i=1}^{N} \langle i|\,\rho_1\Pi_j\,|i\rangle \\
&= \mathrm{Tr}\,\rho_1\Pi_j
\end{aligned}
\tag{2.56}
$$

The first line in the above derivation is just an explicit representation of the trace. To get the second line, we used the fact that the projectors are trivial on the second space; hence, they do not change the state $|j\rangle$, and we can pull it through and leave out the tensor product since the remaining projector acts only on the first part of the system. In the third line, we commuted the second (finite) sum with the "bra" of the first part of the system $\langle i|$. In the fourth line, we introduced the reduced density matrix as a partial trace over the second system $\rho_1 = \mathrm{Tr}_2\rho_{12}$. In the last line, we compactly wrote the equation as a trace of the first system. The meaning of the obtained equation is that to determine any probability on the part of the system, we only need to know the reduced density matrix of the considered part.

Reduced density matrix of a classical probability In classical probability, marginal distributions do not have any information about the joint distribution. On the other hand, quantum marginal probabilities (or reduced density matrices) still preserve just enough information to reconstruct the joint pure state. To see this, we first use the Schmidt decomposition of the joint distribution and write the reduced densities with the Schmidt vectors $|\varphi_k\rangle$ and $|\theta_k\rangle$

$$\rho_{12} = |\psi\rangle\langle\psi| = \sum_{k',k=1}^{K} \sqrt{\lambda_{k'}\lambda_k} \, |\varphi_k\theta_k\rangle\langle\varphi_{k'}\theta_{k'}| \,, \tag{2.57}$$

$$\rho_1 = \text{Tr}_2\,\rho_{12} = \sum_{k=1}^{K} \langle\theta_k|\psi\rangle\langle\psi|\theta_k\rangle = \sum_{k=1}^{K} \lambda_k \, |\varphi_k\rangle\langle\varphi_k| \,, \tag{2.58}$$

$$\rho_2 = \text{Tr}_1\,\rho_{12} = \sum_{k=1}^{K} \langle\varphi_k|\psi\rangle\langle\psi|\varphi_k\rangle = \sum_{k=1}^{K} \lambda_k \, |\theta_k\rangle\langle\theta_k| \,. \tag{2.59}$$

Since the $|\varphi_k\rangle$ and $|\theta_k\rangle$ are orthonormal in the respective Hilbert spaces, they are eigenvectors of the reduced density matrices ρ_1 and ρ_2, respectively. We also see that the reduced density matrices have equal spectra, which are given by $\lambda_1, \lambda_2, \ldots \lambda_N$. Therefore, we can calculate the vectors $|\varphi_k\rangle$, $|\theta_k\rangle$ and the values λ_k by diagonalising the reduced densities ρ_1 and ρ_2. Then, we can reconstruct the joint probability with the help of the Schmidt decomposition. This construction only works if the joint quantum probability is a pure state.

Example 2.15. Let us consider a small example and study two quantum probabilities which determine the same classical probability by the projectors to the computational basis $|00\rangle, |01\rangle, |10\rangle, |11\rangle$. The classical probability should be $\vec{p} = (\frac{1}{2}, 0, 0, \frac{1}{2})$.

Classical probability The classical probability can be written as a density matrix as

$$\rho_{12} = \begin{pmatrix} \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} \end{pmatrix}. \tag{2.60}$$

Then the marginals are

$$\rho_1 = \text{Tr}_1\,\rho_{12} = \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \rho_2 = \text{Tr}_2\,\rho_{12} = \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{2.61}$$

Besides ρ_{12}, there are many density matrices with the above marginals ρ_1 and ρ_2. For example

$$\eta = \rho_1 \otimes \rho_2 = \frac{1}{4}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{2.62}$$

In other words, classical marginal probabilities have no information about the correlations in the joint probability.

Quantum probability Another possibility to encode a classical probability with a quantum probability is to use the pure state

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{2.63}$$

$$\rho_{12} = |\psi\rangle\langle\psi| = \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}. \tag{2.64}$$

We reproduce the correct classical probability since we have diag $(\rho) = \vec{p}$. The marginals are now given by

$$\rho_1 = \frac{1}{2} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \rho_2 = \frac{1}{2} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{2.65}$$

The marginals are the same as in the classical case! How can they now have more information about the joint probability? The difference is that, in this case, the joint probability is a pure state. Which we can reconstruct with the prescription from the main text. From ρ_1 and ρ_2 we first get the the eigenvectors $|\varphi_k\rangle$, θ_k, and the eigenvalues λ_k. Since both reduced densities are diagonal we have $\lambda_1 = \lambda_2 = \frac{1}{2}$, and $|\varphi_1\rangle = |0\rangle$, $|\varphi_2\rangle = |1\rangle$, $|\theta_1\rangle = |0\rangle$, $|\theta_2\rangle = |1\rangle$. Now we reconstruct the initial joint probability via the Schmidt decomposition

$$\sum_{k=1,2} \sqrt{\lambda_k} |\varphi_k\rangle |\theta_k\rangle = \frac{1}{\sqrt{2}} |0\rangle |0\rangle + \frac{1}{\sqrt{2}} |1\rangle |1\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) |\psi\rangle. \tag{2.66}$$

Note that even this procedure is not unique. However, if we know that the initial joint probability describes a classical distribution and has positive phases, then the reconstructed joint probability is unique.

Spooky action at a distance In the example Example. 2.1, we considered correlated coin flips. We showed that the marginal probability for the second coin flip changes when we observe the outcome of the first coin flip. We observe a similar phenomenon with quantum probability, with a crucial difference that no information transfer between the system's first and second parts is necessary to reproduce the correlations. We sometimes refer to this change in the marginal probability of a distant system upon observing the first system as "spooky action at a distance". For concreteness, we consider the standard maximally entangled Bell state in Example. 2.16.

Example 2.16. We start our experiment with a maximally entangled Bell state $|\phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. We will observe the change in the marginal probabilities of the second cubit upon observing the first qubit. Before any observation, the marginal probability of the second qubit is

$$\rho_2 = \frac{1}{2} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{2.67}$$

Then we perform an observation of the first qubit and observe with probability $\frac{1}{2}$ the state $|0\rangle$ and with probability $\frac{1}{2}$ the state $|1\rangle$. We express the updated joint probability after the collapse as

$$|\phi^+\rangle \xrightarrow{\text{observe } 0} |00\rangle, \tag{2.68}$$

$$|\phi^+\rangle \xrightarrow{\text{observe } 1} |11\rangle. \tag{2.69}$$

The marginal probability changes accordingly

$$\rho_2 \xrightarrow{\text{observe } 0} |0\rangle \langle 0|, \tag{2.70}$$

$$\rho_2 \xrightarrow{\text{observe } 1} |1\rangle \langle 1|. \tag{2.71}$$

In other words, if we observed 0 for the first qubit, we will observe 0 also for the second qubit, and if we observed 1 for the first qubit, we will observe 1 for the second qubit. In the classical example, we had to transfer some information about the outcome of the first observation, but this is not the case here.

Transformations of quantum states   We transform quantum states by linear transformation[1]. In the finite-dimensional case, we represent linear transformations with matrices

$$|\psi\rangle \rightarrow M |\psi\rangle. \tag{2.72}$$

Since the transformed state $M |\psi\rangle$ has to be a quantum state as well, it needs to be normalised

$$|\psi|_2 = 1, \quad |M\psi|_2 = 1 \tag{2.73}$$

Norm-preserving matrices are unitary matrices. Hence, we represent transformations of quantum states by unitary matrices

$$|\phi\rangle \rightarrow U |\phi\rangle, \quad U^\dagger U = UU^\dagger = \mathbb{1}_N, \tag{2.74}$$

and in the language of density matrices

$$\rho \rightarrow U\rho U^\dagger. \tag{2.75}$$

Therefore, we write the most general transformation of a reduced density matrix as

$$\rho_1 \rightarrow \text{Tr}_2 U \rho_1 \otimes |\nu\rangle \langle \nu| U^\dagger \tag{2.76}$$

$$= \sum_{\mu=1}^{K} \langle \mu| U |\nu\rangle \rho_1 \langle \nu| U^\dagger |\mu\rangle \tag{2.77}$$

$$= \sum_{\mu=1}^{K} A_\mu \rho_1 A_\mu, \quad \text{where} \quad A_\mu = \langle \mu| U |\nu\rangle. \tag{2.78}$$

Since U is a unitary matrix, we have

$$\sum_{k=1}^{K} A^\dagger A = \sum_{\nu=1}^{K} \langle \nu| U |\mu\rangle \langle \mu| U^\dagger |\nu\rangle = \mathbb{1}_N \tag{2.79}$$

Therefore, the most general transformation of a density matrix can be written with arbitrary operators A_ν that satisfy the condition Eq. 2.79 and is called the Kraus map.

Example 2.17. Imagine we have two unitaries U_1 and U_2 to transform our quantum state ρ. We apply the unitary U_1 with probability p_1 and the unitary U_2 with probability $p_2 = 1 - p_1$. Our statistical description of the transformed density matrix is then

$$\rho \rightarrow \rho' = p_1 U_1 \rho U_1^\dagger + p_2 U_2 \rho U_2^\dagger. \tag{2.80}$$

The transformed matrix is still a valid density matrix since it is a convex combination of density matrices and $p_1 + p_2 = 1$. By introducing $A_j = \sqrt{p_j} U_j$, we write the transformed density matrix with a Kraus map

$$\rho' = \sum_{j=1,2} A_j \rho A_j^\dagger. \tag{2.81}$$

The operators A_j satisfy the condition Eq. 2.79

$$\sum_{j=1,2} A_j^\dagger A_j = \sum_{j=1,2} p_j U_j^\dagger U_j = \sum_{j=1,2} p_j \mathbb{1}_N = \mathbb{1}_N. \tag{2.82}$$

We can interpret the Kraus map as describing quantum operations with classical noise.

---

[1]Non-linear transformations lead to non-physical effects, e.g. superluminal information transfer.

General measurements/events We introduced quantum events by analogy with the classical description. They are described by a set of projectors $\{\Pi_j\}$ that sum to the identity $\sum_j \Pi_j = \mathbb{1}_N$. The probability p_j to observe a particular event Π_j is given by the trace $p_j = \operatorname{Tr} \rho \Pi_j$. We determine the state after the observation by the collapse rule

$$\rho \xrightarrow{\Pi_j} \frac{\Pi_j \rho \Pi_j}{\operatorname{Tr} \Pi_j \rho \Pi_j}. \tag{2.83}$$

Similarly, as we did in the case of the transformations, we can study how the measurement generalises by studying a joint probability

$$\rho_1 \xrightarrow{\Pi_j} \frac{\operatorname{Tr}_2 \left(\Pi_j \rho_1 \otimes |\nu\rangle \langle \nu| \Pi_j\right)}{\operatorname{Tr} \left(\Pi_j \rho_1 \otimes |\nu\rangle \langle \nu| \Pi_j\right)}. \tag{2.84}$$

We find that

$$\operatorname{Tr}_2 \left(\Pi_j \rho_1 \otimes |\nu\rangle \langle \nu| \Pi_j\right) = \sum_{\mu=1}^K \langle \mu| \Pi_j |\nu\rangle \rho_1 \langle \nu| \Pi_k |\mu\rangle = \sum_{\mu=1}^K A_{\mu,j} \rho_1 A_{\mu,j}^\dagger, \quad \text{where} \quad A_{\mu,j} = \langle \mu| \Pi_j |\nu\rangle. \tag{2.85}$$

Since the projectors sum to identity, we have

$$\sum_{\nu,j} A^\dagger A = \sum_{\mu,j} \langle \nu| \Pi_j |\mu\rangle \langle \mu| \Pi_j |\nu\rangle = \mathbb{1}_N \tag{2.86}$$

Example 2.18. We have seen that a measurement determined by a set of projectors $\{\Pi_j\}$ does not uniquely determine the quantum state. However, there exists a generalised measurement described by operators $\{A_k\}$, which determines ρ uniquely. We call such a measurement an informationally complete (IC) measurement. In the case of a qubit, an important IC measurement is given by projectors $\Pi_j = |\psi_j\rangle \langle \psi_j|$ where

$$|\psi_1\rangle = |0\rangle, \tag{2.87}$$

$$|\psi_2\rangle = \frac{1}{\sqrt{3}} |0\rangle + \sqrt{\frac{2}{3}} |1\rangle, \tag{2.88}$$

$$|\psi_3\rangle = \frac{1}{\sqrt{3}} |0\rangle + \sqrt{\frac{2}{3}} e^{i\frac{2\pi}{3}} |1\rangle, \tag{2.89}$$

$$|\psi_4\rangle = \frac{1}{\sqrt{3}} |0\rangle + \sqrt{\frac{2}{3}} e^{i\frac{4\pi}{3}} |1\rangle. \tag{2.90}$$

The projector Π_j also has the property that

$$\operatorname{Tr} \Pi_i \Pi_j = \frac{\delta_{i,j} N + 1}{N + 1}, \tag{2.91}$$

where $N = 2$ is the dimension of the Hilbert space. IC measurements that satisfy the condition Eq. 2.91 are called symmetric IC measurements and are important in quantum tomography and cryptography. [2]

References

- Bengtsson, I., & Życzkowski, K. (2017). Geometry of quantum states: an introduction to quantum entanglement. Cambridge University Press. (Chapter 2: Geometry of probability distributions – 2.1, 2.2, 2.3, 2.4, 2.5, Chapter 5: Outline of quantum mechanics)

- Nielsen, M. A., & Chuang, I. L. (2010). Quantum computation and quantum information. Cambridge University Press. (Chapter 2: introduction to quantum mechanics, Chapter 11: Entropy and information)

3 Quantum computation

3.1 Quantum many-body problem

In this section, we will connect the structure of quantum probability with the physical descriptions of a system. Then, we will discuss the quantum many-body problem and the prospects of quantum machine learning.

Dynamical and statistical description of a quantum system So far, we have seen that we represent transformations of quantum states by unitary matrices. However, we still do not know how we can generate those unitaries. To see this, we must introduce the main object determining a quantum system's behaviour, the Hamiltonian (typically denoted by H). In general, the Hamiltonian is a Hermitian matrix and is connected to the unitary transformations of quantum states by the Schrödinger equation

$$\frac{\mathrm{d}\,|\psi\rangle}{\mathrm{d}t} = -\mathrm{i}H\,|\psi\rangle. \tag{3.1}$$

If the Hamiltonian H is time-independent, then we have

$$|\psi(t)\rangle = \mathrm{e}^{-\mathrm{i}Ht}\,|\psi\rangle = U\,|\psi\rangle \tag{3.2}$$

For a given unitary, we can always find the generating Hamiltonian.

The Hamiltonian is also important for the statistical description of the system. If we have a quantum system at a certain temperature T, we write the density matrix that describes its statistical properties as

$$\rho_T = \mathrm{e}^{-H/T}/Z = \frac{1}{Z}\sum_{j=1}^{N}\mathrm{e}^{-E_k/T}\,|\xi_k\rangle\,\langle\xi_k|, \tag{3.3}$$

where we introduced the normalization $Z = \mathrm{Tr}\,\mathrm{e}^{-H/T}$, and the Hamiltonian eigenvector decomposition $H = \sum_{k=1}^{N} E_k\,|\xi_k\rangle\,\langle\xi_k|$. We observe that the eigenvectors follow an exponential (softmax) distribution. If we lower the temperature, the states with the lowest energies become exponentially more likely, and at zero temperature, the system is in the ground state, i.e. the state with the smallest energy. We describe the quantum system as being in a mixed state at higher temperatures.

Example 3.1. To get an intuition about the Hamiltonian, we will first consider a classical problem. Then, we will rewrite it using the quantum formalism. Finally, we will generalise it to the quantum case.

Imagine we have N magnets arranged on a line. Let us assume that we can place the magnets either with the north pole up or the south pole up. We associate a scalar $\sigma_j \in \{-1,+1\}$ to each of the states. If the first magnet is in the state $\sigma_1 = +1$, the next magnet prefers to be in the $\sigma_2 = -1$. We can describe this interaction by a scalar product $H = \sum_j \sigma_j\sigma_{j+1}$. Further, we can tune the strength of the interaction between the magnets by controlling the size of the wall between two magnets. In this case, we write $H = \sum_j J_j\sigma_j\sigma_{j+1}$. Finally, we can add a bigger magnet below each magnet, which always has the north pole up. The final Hamiltonian describing the energy of the system is

$$H_{\mathrm{Ising}} = \sum_{j=1}^{n-1} J_j\sigma_j\sigma_{j+1} + \sum_{j=1}^{n} h_j\sigma_j. \tag{3.4}$$

We now rewrite the classical Ising Hamiltonian Eq. 3.4 with the quantum formalism by introducing the standard embedding $\Phi(1) = |0\rangle$ and $\Phi(-1) = |1\rangle$. The classical Ising Hamiltonian can then be written with the σ^z Pauli matrix Since we have $\sigma^z |0\rangle = +1 |0\rangle$. We have $\sigma^z |1\rangle = -1 |1\rangle$, and

$$H_{\text{Ising}}^{\text{q}} = \sum_{\langle i,j \rangle} J_{i,j} \sigma_i^z \sigma_j^z + \sum_j h_j \sigma_j^z. \tag{3.5}$$

The notation $\langle i, j \rangle$ in the sum means we have a sum over the nearest neighbours. The above equation is the quantum formulation of the classical Ising model since we have $H_{\text{Ising}}(\sigma_1, \sigma_2, \ldots \sigma_n) = \Phi^\dagger(\sigma_1) \otimes \Phi^\dagger(\sigma_2) \otimes \ldots \Phi^\dagger(\sigma_n) H_{\text{Ising}}^{\text{q}} \Phi(\sigma_1) \otimes \Phi(\sigma_2) \otimes \ldots \Phi(\sigma_n)$, where $\Psi^\dagger(1) = \langle 0|$ and $\Phi^\dagger(-1) = \langle 1|$. We see that operators acting on a single qubit encode an external field, and the operators acting on two or more qubits encode the interaction between the qubits.

Finding the ground state, i.e. the state with the lowest energy, of the Ising model directly corresponds to quadratic unconstrained binary optimisation (QUBO)

$$\min \sum_{\langle i,j \rangle} w_{i,j} x_i x_j + \sum_j b_j x_j \tag{3.6}$$

where $x_j \in \{0, 1\}$. The QUBO problem is NP-hard, translating into the Ising model's long relaxation times. We already see an intimate connection between interesting computational problems and paradigmatic physical models. Therefore, we can use observable properties of actual physical systems to solve interesting computational problems.

The model $H_{\text{Ising}}^{\text{q}}$ is classical since all operators in the sums commute. We must add a term that does not commute with σ_j^z to introduce quantum effects. The standard choice is a local field in the x direction modelled by the Pauli matrix σ_j^x. The resulting model is called the transverse field Ising model (TFIM)

$$H_{\text{TFIM}} = \sum_{\langle i,j \rangle} J_{i,j} \sigma_i^z \sigma_j^z + \sum_{j=1} (h_j^z \sigma_j^z + h_j^x \sigma_j^x). \tag{3.7}$$

As we will see later, the TFIM is still insufficient to perform universal quantum computation, but it is sufficient to solve interesting NP-hard problems, e.g., the travelling salesman problem.

Why is quantum computing hard? Exact solutions to quantum problems are rare. The reason is that the size of the Hilbert space of a many-body quantum problem grows exponentially with the number of qubits, i.e., 2^n. Therefore, even numerically, we can find exact solutions for problems up to 40 qubits. Nevertheless, we can describe much larger systems in many important special cases. We typically exploit some of the models' symmetry or unique properties in these cases, e.g., low entanglement, local conservation laws, or small parameters. Accordingly, we developed many different methods to describe these special cases:

- **Analytical methods**: perturbation theory, renormalisation group, integrability, quadratic systems,
- **Numerical methods**: Monte Carlo, tensor networks, numerical renormalisation group.

Special cases are important since they often contain essential information about more realistic physical systems. However, they cover a negligible part of the Hilbert space, representing two opportunities from the quantum machine learning perspective. First, we can use standard machine learning tools to describe many-body quantum systems and cover more Hilbert space with new numerical methods. Second, we can use the classically inaccessible part of the Hilbert space to improve machine learning algorithms. Here, we will focus mainly on the latter.

Quantum advantage   A natural framework to study how quantum computation can assist machine learning and quantify the quantum advantage is complexity theory. We will distinguish computational complexity, sample complexity, and model complexity.

Computational (quantum) complexity is the most common approach to finding the benefits of quantum computing. The main tool is the asymptotic runtime complexity, which denotes the asymptotic functional dependence of the runtime with respect to the input size. A quantum speedup means the quantum algorithm is faster than the classical one in asymptotic complexity. We distinguish five different classes of quantum speedup

1. *Provable quantum speedup:* we have proof that no classical algorithm can perform better than the quantum algorithm, Grover's search (quadratic speedup)

2. *Strong quantum speedup:* the quantum algorithm is better than the best classical algorithm, Shor's factoring (exponential speedup)

3. *Common quantum speedup:* the quantum algorithm is better than the best available classical algorithm

4. *Potential quantum speedup:* the quantum algorithm is better than a specific classical algorithm

5. *Limited quantum speedup:* quantum algorithm is better than the corresponding classical algorithm (useful when comparing quantum and classical annealing)

Sample complexity is related to generalisation. It determines how many samples we need to generalise from data. Most evidence considering quantum and classical sample complexity suggests they are polynomially equivalent. However, upon introducing noise, the separation becomes more drastic. The quantum algorithm can still generalise in some cases, while the classical problem is considered unlearnable.

Finally, model complexity refers to the expressiveness of the models, i.e. the classes of functions we can express with a given number of parameters. Due to overfitting, expressive models are not necessarily better than "slim" models. In this context, the main avenue of research is experimental since it is not apparent which quantum models have a suitable bias to capture patterns in specific datasets.

References

- Schuld, M., & Petruccione, F. (2021). Machine learning with quantum computers (Vol. 676, pp. 163-169). Berlin: Springer. (Chapter: Quantum Computing, Chapter: Potential Quantum Advantages)

3.2   Models of quantum computation

There are several competing models of quantum computation: gate model, adiabatic quantum computation, measurement-based quantum computation and continuous variable quantum computation. They are equivalent to a polynomial overhead.

Gate model of quantum computation   The most common approach to quantum computing is the gate model of quantum computation. Here, we generate complex quantum transformations by composing smaller, simple-to-implement quantum transformations called quantum gates. The two main components that ensure the scalability of the gate model are fault tolerance and universality. Fault tolerance means we can perform logical transformations even if some of our physical qubits are corrupted. It was shown that it is possible to implement fault-tolerant schemes for a finite set of gates. Conveniently, universality states

that it is possible to implement any quantum transformation with a finite set of gates. More formally, the Solovay-Kitaev theorem shows that for any gate U on a single qubit, and given any $\epsilon > 0$, it is possible to approximate U to a precision ϵ using $\mathcal{O}(\log c(1/\epsilon))$ gates from a fixed finite set, where c is a small constant approximately equal to 2. Combining those two results, we can deduce that we can implement any algorithm with a finite set of universal quantum gates. Importantly, the number of gates we must use for fault-tolerant computation is only polylogarithmically larger than the number of simple transformations in the original algorithm. Finally, there are infinitely many universal sets of quantum gates. The most common universal set is: CNOT, Hadamard gate H, phase gate S and the T gate:

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \qquad H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \qquad T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$$

Example 3.2. We typically represent quantum circuits with diagrams as shown in Fig. 3.1. We generally draw the diagrams from left to right. We start by writing the initial states of the qubits $|0\rangle$. To each of the states, we associate a wire. If we see only a wire, the qubit will not change during the computation. We draw the unitary transformations as boxes with names inside that denote the type of the transformation. Connecting the corresponding wires to the box indicates the qubits on which the unitary acts. Finally, on the right, we draw a special box to denote a measurement of one or more qubits.
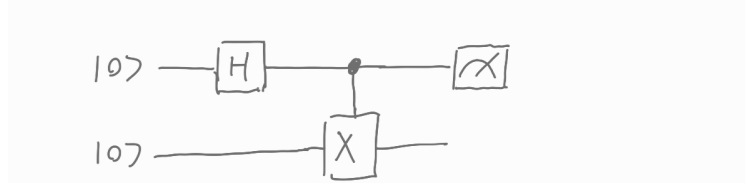

Figure 3.1: Example quantum circuit. We start in a state $|00\rangle$, then transform it with a Hadamard gate and a CNOT gate to the state $|\phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, and finally we measure the first qubit.

Practically, several challenges limit the applicability of the gate model. The main problem is that the current implementation of gates is noisy and that we can apply only up to several 100 transformations. Another critical issue that we should consider in this setting is that not all qubits are connected. Sometimes, we have to use more than two qubits and additional gates to apply a desired interaction between the two qubits.

Adiabatic quantum computation   Adiabatic quantum computing builds on two observations/statements. I) Optimisation problems can be formulated such that a quantum state with minimum energy solves the problem determined by a Hamiltonian H_1. II) A quantum system remains in its instantaneous eigenstate if a given perturbation is acting on it slowly enough. The idea of adiabatic quantum computing is then the following. Start with a ground state of a simple Hamiltonian (e.g. $H_0 = \sum_{i=1}^{N} \sigma_i^x$) and then gradually change it to the final Hamiltonian (e.g. $H_1 = \sum_{\langle i,j \rangle} J_{i,j} \sigma_i^z \sigma_j^z + \sum_i h_i \sigma_i^z$) which encodes the solution of a desired problem in its lowest energy state. The caveat is that there is a speed limit on how fast the transition from one H_0 to H_1 can be. The transition time is lower bounded by the square of the inverse of the difference between the two lowest energy states.

We can avoid the speed limit problem of adiabatic quantum computing by relaxing the condition of being in the best possible state and doing many tries of the same protocol. If the probability of staying in the ground state is large enough, we will eventually get the best possible solution to the problem. This is the quantum annealing approach to quantum computation.

As in the gate model, there are also some practical obstacles. The main ones are finite temperature and interference from measurement devices. Since they disturb the system, they increase the transition probability to a higher energy state (i.e. a sub-optimal solution).

Measurement-based and continuous variable quantum computation   Measurement-based quantum computing consists of two steps. First, we prepare a highly entangled quantum state (a cluster state). Then, we perform conditional single-qubit measurements. We use the results of the measurements as conditions for the following measurements. The result of the computation is either the state of unmeasured qubits or the result of the last measurement. Though we have translated many important algorithms into this framework, it is unclear how they can present an advantage for NISQ devices.

So far, we have covered quantum computation with finite-dimensional systems. Continuous-variable quantum computing, on the other hand, manipulates quantum states whose matrix representations are infinite-dimensional. Contrary to measurement-based quantum computing, continuous-variable quantum computing seems beneficial for machine learning algorithms due to a large local Hilbert space that we can use to store and manipulate information.

Implementations   There are several approaches to building a quantum computer. Their detailed description is out of the scope of the lectures. We only list a few important properties and obstacles:

Superconducting circuits

- They are silicon-based; hence, we can use a lot of existing infrastructure
- Typical operating temperature is $\sim 10mK$
- Short control time $\sim 10ns$
- 2D connectivity. Therefore, we can not implement all possible two-body interactions.

Trapped ions

- Implemented by particles in magnetic traps
- Interactions are controlled by lasers
- Long coherence time
- Slow control time

Photonic-based systems

- Based on light, hence work at room temperature
- The main implementation problem is photon loss
- Difficult to store
- It is not clear how they can scale

References

- Schuld, M., & Petruccione, F. (2021). Machine learning with quantum computers (Vol. 676, pp. 163-169). Berlin: Springer. (Chapter: Quantum Computing)

- Nielsen, M. A., & Chuang, I. L. (2010). Quantum computation and quantum information. Cambridge University Press. (Chapter 7: Quantum computers: physical realization)

3.3 Fault-tolerant quantum computing: basic algorithms

In this section, we consider the gate model of quantum computation. We will introduce it by translating essential quantum experimental protocols into quantum circuits. Then, we will provide some examples that show quantum speedup. Finally, we will discuss several types of algorithms for fault-tolerant quantum computers.

Quantum gates As we already discussed, the main building block of the gate model of quantum computing is a finite set of quantum gates that act only on a few qubits. Typically, we extend the universal set of quantum gates given in the previous section to be able to implement quantum protocols with a lesser number of gates. The following tables provide the standard set of gates implemented on current NISQ devices.

Gate	Circuit	Matrix	Dirac
X	$-\boxed{X}-$	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$\lvert 0 \rangle \langle 1 \rvert + \lvert 1 \rangle \langle 0 \rvert$
Y	$-\boxed{Y}-$	$\begin{pmatrix} 0 & -\mathrm{i} \\ \mathrm{i} & 0 \end{pmatrix}$	$\mathrm{i}(\lvert 1 \rangle \langle 0 \rvert - \lvert 0 \rangle \langle 1 \rvert)$
Z	$-\boxed{Z}-$	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	$\lvert 0 \rangle \langle 0 \rvert - \lvert 0 \rangle \langle 0 \rvert$
H	$-\boxed{H}-$	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$	$\frac{1}{\sqrt{2}}(\lvert 0 \rangle \langle 0 \rvert + \lvert 0 \rangle \langle 1 \rvert + \lvert 1 \rangle \langle 0 \rvert - \lvert 1 \rangle \langle 1 \rvert)$
S	$-\boxed{S}-$	$\begin{pmatrix} 1 & 0 \\ 0 & \mathrm{i} \end{pmatrix}$	$\lvert 0 \rangle \langle 0 \rvert + \mathrm{i}\lvert 1 \rangle \langle 1 \rvert$
T	$-\boxed{T}-$	$\begin{pmatrix} 1 & 0 \\ 0 & \mathrm{e}^{\mathrm{i}\frac{\pi}{4}} \end{pmatrix}$	$\lvert 0 \rangle \langle 0 \rvert + \mathrm{e}^{\mathrm{i}\frac{\pi}{4}}\lvert 1 \rangle \langle 1 \rvert$

Table 1: Common one-qubit quantum gates

Example 3.3. No cloning theorem While copying classical bits is not a problem, we can not copy quantum states. More precisely, we can rule out the existence of a unitary transformation U which would produce a state $\lvert \psi \rangle \lvert \psi \rangle$ when acting on a state $\lvert \psi \rangle \lvert 0 \rangle$ for any $\lvert \psi \rangle$. This is easy to show by contradiction. Imagine we have such a unitary. Then, we can write

$$U(\lvert \psi, 0 \rangle) = \lvert \psi, \psi \rangle, \quad U(\lvert \phi, 0 \rangle) = \lvert \phi, \phi \rangle. \tag{3.8}$$

It follows that

$$\langle \psi, 0 \vert \phi, 0 \rangle = \langle \psi \vert \phi \rangle \tag{3.9}$$

$$\langle \psi, 0 \vert \phi, 0 \rangle = \langle \psi, 0 \vert U^{\dagger} U \lvert \phi, 0 \rangle = \langle \psi, \psi \vert \phi, \phi \rangle = (\langle \psi \vert \phi \rangle)^2 \tag{3.10}$$

Gate	Circuit	Matrix	Dirac
CNOT	-[X]-	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$	$\lvert 00\rangle\langle 00\rvert + \lvert 01\rangle\langle 01\rvert + \lvert 10\rangle\langle 11\rvert + \lvert 11\rangle\langle 10\rvert$
SWAP	-[y]-	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$	$\lvert 00\rangle\langle 00\rvert + \lvert 10\rangle\langle 01\rvert + \lvert 01\rangle\langle 10\rvert + \lvert 11\rangle\langle 11\rvert$

Table 2: Common two-qubit quantum gates

Gate	Circuit	Description
qubit	——	wire with one qubit
n qubits	⧸n	wire with n qubits
C-U	-[u]-	one qubit control of one qubit unitary
measurement	-[⦜]-	projection on one of the states $\lvert 0\rangle, \lvert 1\rangle$

Table 3: Other common operations.

We find the equation $x^2 = x$, which has only two solutions, namely $x = 0$ and $x = 1$. This means that we can copy only one set of orthogonal basis states. We will not exactly copy any other state.

Bell states We always initialise the circuit-based quantum computers in a state with all qubits in the state $\lvert 0\rangle$. As our first protocol, we will discuss in detail how we can transform the two-qubit, product state $\psi_0 = \lvert 0,0\rangle$ into entangled Bell states

$$\left\lvert \phi^{\pm}\right\rangle = \frac{1}{\sqrt{2}}(\lvert 0,0\rangle \pm \lvert 1,1\rangle), \quad \left\lvert \psi^{\pm}\right\rangle = \frac{1}{\sqrt{2}}(\lvert 0,1\rangle \pm \lvert 1,0\rangle). \tag{3.11}$$

We describe the circuit constructing the state $\lvert \phi^{+}\rangle$ shown in Fig. 3.2 a. We first transform the initial state by a Hadamard gate acting on the first qubit, which transforms the state of the first qubit as shown in table 1, while the second qubit is unchanged. We get the state ψ_1.

$$\lvert \psi_1\rangle = \frac{1}{\sqrt{2}}(\lvert 0\rangle + \lvert 1\rangle)\lvert 0\rangle = \frac{1}{\sqrt{2}}(\lvert 0,0\rangle + \lvert 1,0\rangle). \tag{3.12}$$

Then, we transform ψ_1 by a controlled-NOT operation with the first qubit as a control. This means that all states where the first qubit is in a state $\lvert 0\rangle$ remain unchanged. All states where the first qubit is in a state $\lvert 1\rangle$ will have the second qubit flipped after the transformation. By applying this rule to the state ψ_1, we get the final result

$$\lvert \psi_2\rangle = \frac{1}{\sqrt{2}}(\lvert 0,0\rangle + \lvert 1,1\rangle), \tag{3.13}$$

To get the state ψ^{+}, we have to flip one of the qubits. We do this with the X gate, which we should apply after the CNOT gate (see Fig. 3.2 b). We must add a phase to the states with the first qubit in the state $\lvert 1\rangle$ to get the remaining states. We can achieve this by a controlled Z gate at the end of the circuit or a bit flip on the first qubit applied before the Hadamard transformation (see Fig. 3.2 c) and Fig. 3.2 d)).
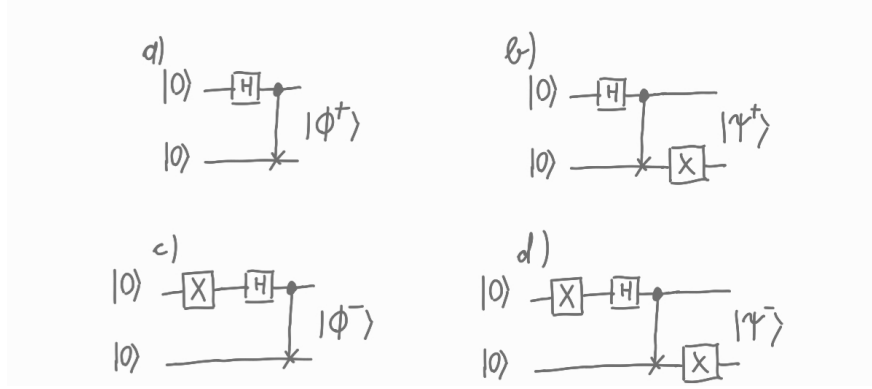
Figure 3.2: Quantum protocols for preparation of the single qubit Bell basis.

Teleportation One of the most interesting quantum experiments is teleportation. Here, we will consider a qubit teleportation. In this experiment, we have an entangled qubit in a state $|\psi^+\rangle = \frac{1}{\sqrt{2}}(|0,0\rangle + |1,1\rangle)$ and an additional unknown qubit in a state $|\psi\rangle = a\,|0\rangle + b\,|1\rangle$. By performing a series of transformations, we can transfer the state of the additional qubit to one of the entangled qubits. This interesting point is that the entangled qubit holding the state can be far away from the additional qubit. We show the circuit describing the gates performing this operation in Fig. 3.3. The system's initial state is $|\psi_0\rangle$. It gets transformed during the protocol as follows

$$|\psi_0\rangle = |\psi, \phi\rangle = (a\,|0\rangle + b\,|1\rangle)\frac{1}{\sqrt{2}}(|0,0\rangle + |1,1\rangle) \tag{3.14}$$

$$|\psi_1\rangle = \frac{1}{\sqrt{2}}(a\,|0\rangle\,(|0,0\rangle + |1,1\rangle) + b\,|1\rangle\,(|1,0\rangle + |0,1\rangle)) \tag{3.15}$$

$$|\psi_2\rangle = \frac{1}{2}\,(|0,0\rangle\,(a\,|0\rangle + b\,|1\rangle) + |0,1\rangle\,(b\,|0\rangle + a\,|1\rangle)) + |1,0\rangle\,(a\,|0\rangle - b\,|1\rangle) + |1,1\rangle\,(b\,|1\rangle - a\,|0\rangle) \tag{3.16}$$

$$|\psi_3\rangle = \begin{cases} a\,|0\rangle + b\,|1\rangle & (M_1 = 0, \quad M_2 = 0) \\ b\,|0\rangle + a\,|1\rangle & (M_1 = 0, \quad M_2 = 1) \\ a\,|0\rangle - b\,|1\rangle & (M_1 = 1, \quad M_2 = 0) \\ b\,|0\rangle - a\,|1\rangle & (M_1 = 1, \quad M_2 = 1) \end{cases} \tag{3.17}$$

$$\tag{3.18}$$

Based on the measurements M_1 and M_2, we can transform the state of the last qubit into the state ψ by applying controlled bit flip (gate X) and phase flip (gate Z).

The quantum teleportation protocol is important for quantum computation since it provides a way to transform different types of resources: 1 qubit = 2 bits + 1 entangled pair. It is also used as a part of error-correcting protocols and can be interpreted (with few modifications) as a simple example of a measurement-based quantum computation.

Quantum parallelism We will now discuss a quantum circuit that evaluates the value of a function $f(x)$. We will illustrate another essential feature of quantum computation: quantum parallelism.

We consider a binary function

$$f(x) : \{0, 1\} \to \{0, 1\}. \tag{3.19}$$

There are four such functions: identity, bit flip, constant 0, and constant 1. The idea is to construct a unitary U_f to implement these functions. It can not be a one-qubit unitary since the constant functions can
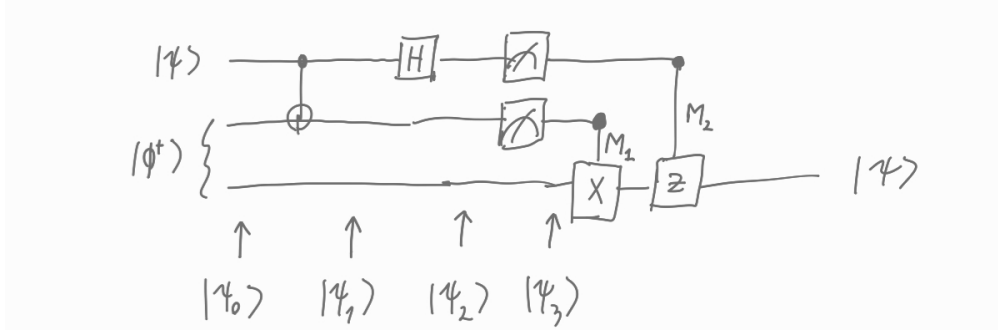
Figure 3.3: Single qubit teleportation protocol

not be uniquely inverted. Hence we define the unitary U_f as

$$(x, y) \xrightarrow{U_f} (x, y \oplus f(x)). \tag{3.20}$$

The $\oplus$ denotes mod 2 addition, namely $0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$, $1 \oplus 1 = 0$. In the dirac notation we write $U_f |x, y\rangle = |x, y \oplus f(x)\rangle$. That this is a unitary transformation, it is not difficult to see since

$$U_f^2 |x, y\rangle = U_f |x, y \oplus f(x)\rangle = |x, (y \oplus f(x) \oplus f(x))\rangle = |x, y\rangle. \tag{3.21}$$

Applying the operation on a state $|x, 0\rangle$ we get $U_f |x, 0\rangle = |x, f(x)\rangle$. If we now prepare the first qubit in the state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and apply the unitary transformation we get

$$U_f \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) |0\rangle = \frac{1}{\sqrt{2}}(U_f |0, 0\rangle + U_f |1, 0\rangle) = \frac{1}{\sqrt{2}}(|0, f(0)\rangle + |1, f(1)\rangle). \tag{3.22}$$

One application of U_f produced the results of $f(0)$ and $f(1)$ in two orthogonal states. However, it is important to note that we can not access this information immediately since the results of measurements are probabilistic, and we can not choose which state of the first qubit to observe.

Deutsch algorithm   This section will show how we can extract useful information from a quantum state of the form Eq. 3.22. We consider a problem to determine if the function f is constant or balanced. A constant function has $f(0) = f(1)$, and a balanced has $f(0) \neq f(1)$. We need two applications of f to determine if f is constant or balanced. A quantum algorithm (called Deutsch algorithm) shown in Fig. 3.4 can do that with only one application of U_f, namely

$$(H \otimes \mathbb{1})U_f(H \otimes H)(\mathbb{1} \otimes X) |0, 0\rangle. \tag{3.23}$$

The transformations of the Deutsch algorithm are as follows (see Fig. 3.4)

$$|\psi_0\rangle = |0, 0\rangle \tag{3.24}$$

$$|\psi_1\rangle = (\mathbb{1} \otimes X) |\psi_0\rangle \quad = |0, 1\rangle \tag{3.25}$$

$$|\psi_2\rangle = (H \otimes H) |\psi_1\rangle \quad = \frac{1}{2}(|0\rangle + |1\rangle)(|0\rangle - |1\rangle) = \frac{1}{2}\left(\sum_{x=0,1} |x\rangle\right)(|0\rangle - |1\rangle) \tag{3.26}$$

$$|\psi_3\rangle = U_f |\psi_2\rangle \quad = \frac{1}{2}\left(\sum_{x=0,1} (-1)^{f(x)} |x\rangle\right)(|0\rangle - |1\rangle) \tag{3.27}$$

$$|\psi_4\rangle = (H \otimes \mathbb{1}) |\psi_3\rangle \quad = \frac{1}{2}\left(\left((-1)^{f(0)} + (-1)^{f(1)}\right) |0\rangle + \left((-1)^{f(0)} - (-1)^{f(1)}\right) |1\rangle\right)(|0\rangle - |1\rangle) \tag{3.28}$$
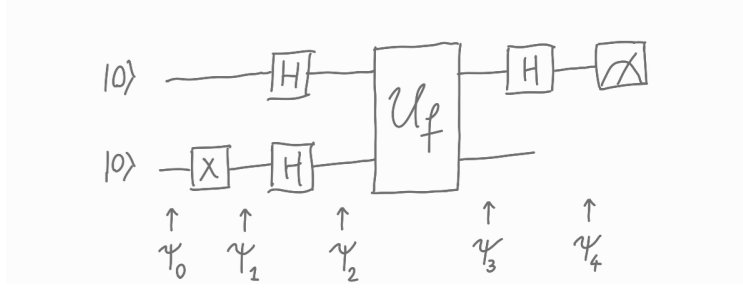
33

Figure 3.4: Diagramatic representation of the Deutsch algorithm to determine if the function is balanced or constant.

If we now measure the first qubit and observe the state $|0\rangle$, we know that $f(0) = f(1)$, and if we observe the state $|1\rangle$, we know that $f(0) \neq f(1)$. Hence, we evaluated a global property of a function without explicitly calculating its values.

Deutsch-Josza algorithm   We can generalise the Deutsch algorithm to n-qubits. In this case, we consider a function

$$f(x) : \{0,1\}^n \to 0, 1. \tag{3.29}$$

The function f is constant if all 2^n inputs have the same result, and it is balanced if half of the inputs result in 1 and the other half in 0. To evaluate whether the function is balanced or constant classically, we need to evaluate on average $\mathcal{O}(2^N)$ different inputs. The quantum Deutsch-Josza algorithm shown in Fig. 3.5 does this with one application of U_f defined as

$$U_f |x, z\rangle = |x, z \oplus f(x)\rangle, \quad x = (x_1, x_2 \ldots x_n), \quad z \in \{0, 1\}. \tag{3.30}$$

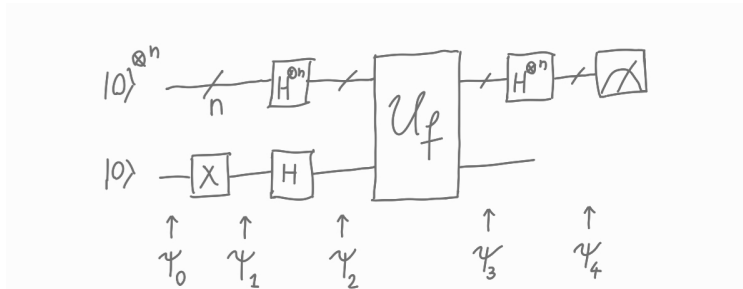Figure 3.5: Diagramatic representation of the Deutsch-Jozsa algorithm to determine if a function f is balanced or constant.

Also, the algorithm is a straightforward generalisation of the Deutsch algorithm to n qubits (plus one auxiliary qubit).

$$(\mathbb{H}^{\otimes n} \otimes \mathbb{1}_2) U_f (\mathbb{H}^{\otimes n} \otimes H)(\mathbb{1}_{2^n} \otimes X) |0\rangle^{\otimes n} |0\rangle. \tag{3.31}$$

Similarly, as in the one-qubit case, we write the state after each transformation

$$|\psi_0\rangle = |0^{\otimes n}, 0\rangle \tag{3.32}$$

$$|\psi_1\rangle = (\mathbb{1}_{2^n} \otimes X) |\psi_0\rangle \qquad = |0^{\otimes n}, 1\rangle \tag{3.33}$$

$$|\psi_2\rangle = (H^{\otimes n} \otimes H) |\psi_1\rangle \qquad = \frac{1}{\sqrt{2^{n+1}}} \left(\sum_{s_1=0,1} |s_1\rangle \right) \left(\sum_{s_2=0,1} |s_2\rangle \right) \dots \left(\sum_{s_n=0,1} |s_n\rangle \right) (|0\rangle - |1\rangle) \tag{3.34}$$

$$= \frac{1}{\sqrt{2^{n+1}}} \left(\sum_{x=0}^{2^n-1} |x\rangle \right) (|0\rangle - |1\rangle) \tag{3.35}$$

$$|\psi_3\rangle = U_f |\psi_2\rangle \qquad = \frac{1}{\sqrt{2^{n+1}}} \left(\sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle \right) (|0\rangle - |1\rangle) \tag{3.36}$$

$$|\psi_4\rangle = (H^{\otimes n} \otimes \mathbb{1}) |\psi_3\rangle \qquad = \frac{1}{\sqrt{2^{n+1}}} \left(\sum_{x,z=0}^{2^n-1} (-1)^{x \cdot z + f(x)} |z\rangle \right) (|0\rangle - |1\rangle). \tag{3.37}$$

To find the last equality, we used the identity

$$H^{\otimes n} |x_1, x_2, \dots x_n\rangle = \frac{1}{\sqrt{2^n}} \sum_{z=0}^{2^n-1} (-1)^{z_1 x_1 + z_2 x_2 + \dots z_n x_n} |z_1, z_2, \dots z_n\rangle. \tag{3.38}$$

The probability of observing a state corresponding to a constant function is

$$p(0, 0, \dots 0) = \left| \frac{1}{2^n} \sum_{x=0}^{2^n-1} (-1)^{f(x)} \right|^2. \tag{3.39}$$

If the function is constant, the probability is one, and if the function is balanced, the probability is 0. The presented Deutsch-Jozsa algorithm is the simplest case where we can show exponential speedup with respect to the best-known classical algorithm.

Categories of quantum algorithms The Deutsch-Jozsa solves a problem which is not practically relevant. However, similar problems are practically important and can be solved efficiently on a quantum computer but not on a classical computer. This is a class of problems related to the quantum Fourier transform, which we will discuss in the next section.

An entirely different class of algorithms are the quantum search algorithms. A prime example is Grover's algorithm. These general algorithms solve unstructured problems; we can not expect exponential speedup as in the Fourier transformation class. For example, Grover's search is quadratically better than the best classical version. However, we can make stronger theoretical claims. While the Fourier algorithms are compared to the best-known classical algorithms, we can prove that Grover's search is the best possible quantum algorithm, quadratically better than the best possible classical algorithm.

The last class of algorithms for fault-tolerant quantum devices are the quantum simulation algorithms. They also achieve exponential speedups with respect to the best-known classical algorithms but do not rely on the Fourier transformation. They are simply simulating a many-body quantum system. The exponential speedup comes from the fact that quantum simulation is classically challenging.

References

- Nielsen, M. A., & Chuang, I. L. (2010). Quantum computation and quantum information. Cambridge University Press. (Chapter 1.3: Quantum computation, Chapter 1.4: Quantum algorithms, Chapter 4: Quantum circuits)

3.4 Quantum Fourier transformation

In this section, we will look at the quantum Fourier transformation (QFT) and then discuss a set of routines that can be used to speed up many classical machine learning algorithms.

Algorithm The QFT algorithm shown in Fig. 3.6 is a quantum version of the classical discrete Fourier transformation. The classical discrete Fourier transformation maps a vector of complex numbers $x_1, x_2 \ldots x_{2^n}$ to a new vector $y_1, y_2 \ldots y_{2^n}$ defined as

$$y_k = \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} e^{\frac{2\pi i j k}{2^n}} x_j. \tag{3.40}$$

The quantum version is then defined by analogy. The basis state $|j\rangle$ is transformed as

$$|j\rangle \xrightarrow{\text{QFT}} \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi i j k}{2^n}} |k\rangle. \tag{3.41}$$

The action on an arbitrary state is then

$$\sum_{j=0}^{2^n-1} x_j |j\rangle \xrightarrow{\text{QFT}} \sum_{k=0}^{2^n-1} y_k |k\rangle, \tag{3.42}$$

where y_k are the discrete Fourier transformed amplitudes defined in Eq. 3.40. The presented quantum transformation is a unitary transformation, which we will show by explicitly constructing the circuit for its implementation.

First, we define some useful conventions/notation, which we will use throughout the lecture notes. A natural number j can be written in a binary representation $j = j_1 j_2, \ldots j_n$, which more formally means $j = j_1 2^{n-1} + j_2 2^{n-2} + \ldots j_n 2^0$. Similarly, we can represent a binary fraction as $0.j_l j_{l+1} \ldots j_m = j_l/2 + j_{l+1}/2^2 + \ldots j_m/2^{m-l+1}$.

To obtain a useful representation for a QFT, we rewrite the definition Eq. 3.41 as follows

$$|j\rangle \xrightarrow{\text{QFT}} \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi i j k}{2^n}} |k\rangle \tag{3.43}$$

$$= \frac{1}{\sqrt{2^n}} \sum_{k_1=0}^{1} \ldots \sum_{k_n=0}^{1} e^{2\pi i j \left(\sum_{l=1}^{n} k_l 2^{-l} \right)} |k_1, k_2, \ldots, k_n\rangle \tag{3.44}$$

$$= \frac{1}{\sqrt{2^n}} \sum_{k_1=0}^{1} \ldots \sum_{k_n=0}^{1} \prod_{l=1}^{n} e^{2\pi i j k_l 2^{-l}} |k_1, k_2, \ldots, k_n\rangle \tag{3.45}$$

$$= \frac{1}{\sqrt{2^n}} \bigotimes_{l=1}^{n} \left[\sum_{k_l=0}^{1} e^{2\pi i j k_l 2^{-l}} |k_l\rangle \right] \tag{3.46}$$

$$= \frac{1}{\sqrt{2^n}} \bigotimes_{l=1}^{n} \left[|0\rangle + e^{2\pi i j 2^{-l}} |1\rangle \right] \tag{3.47}$$

$$= \frac{1}{\sqrt{2^n}} (|0\rangle + e^{2\pi i 0.j_n} |1\rangle)(|0\rangle + e^{2\pi i 0.j_{n-1} j_n} |1\rangle) \ldots (|0\rangle + e^{2\pi i 0.j_1 j_2 \ldots j_n} |1\rangle) \tag{3.48}$$

The final representation of the QFT Eq. 3.48 is convenient to derive an efficient circuit for its implementation presented in Fig. 3.6. The gate R_k denotes a single qubit phase shift

$$R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix}.$$ (3.49)
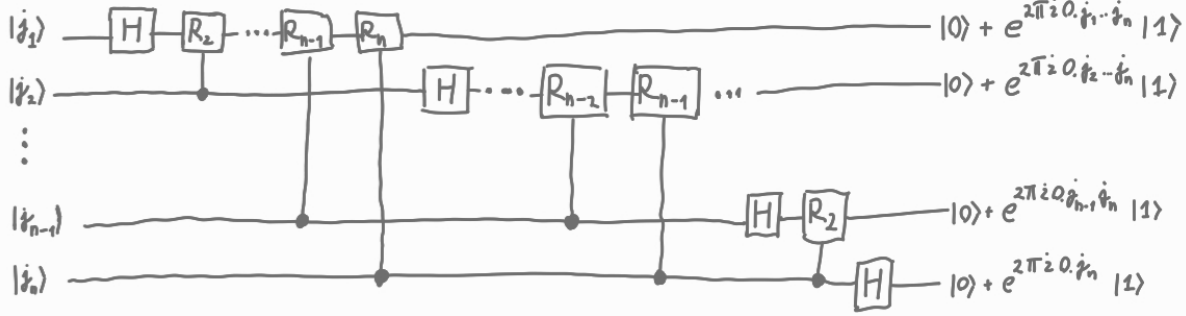

Figure 3.6: The main part of the QFT algorithm.

To show that the algorithm Fig. 3.6 represents the QFT, we focus on the transformation of the first qubit of the state $|j\rangle = |j_1, j_2, \ldots j_n\rangle$ (top wire in Fig. 3.6). Application of the Hadamard on the first qubit produces the state

$$\frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi i 0.j_1} |1\rangle \right) |j_2 \ldots j_n\rangle.$$ (3.50)

Above follows from the fact that $e^{2\pi i 0.j_1} = -1$ if $j_1 = 1$ and $e^{2\pi i 0.j_1} = 1$ if $j_1 = 0$. Then, we apply the R_2 gate and again shift the phase of the state $|1\rangle$ as

$$\frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi i 0.j_1 j_2} |1\rangle \right) |j_2 \ldots j_n\rangle.$$ (3.51)

This follows directly from the definition of the R_k gate. Then we apply the controlled rotations $R_3, \ldots R_n$ and get the state

$$\frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi i 0.j_1 j_2 \ldots j_n} |1\rangle \right) |j_2 \ldots j_n\rangle.$$ (3.52)

Transformations of other qubits follow the same pattern with a decreasing number of conditional rotations, resulting in fewer digits in the phase shift. Comparing the representation Eq. 3.48 of the QFT and the output of the circuit Fig. 3.6, we see that the outputs are precisely reversed. Therefore, we have to reverse the order of the states obtained by the presented algorithm by applying $n-1$ swap gates.

The presented algorithm is efficient in the number of qubits since it requires $\mathcal{O}(n^2)$ simple two-qubit gates. In contrast, the classical algorithm needs to apply $\mathcal{O}(n2^n)$ elementary operations. The main caveat is that there is no known efficient way of extracting all the information stored in the amplitudes. Therefore, we use QFT as a part of algorithms to extract some global properties of functions, which would otherwise require the computation of their values for all possible inputs. A prime example is the Deutsch-Jozsa algorithm discussed in the previous section.

Quantum phase estimation One of the most used quantum procedures based on the QFT is the phase estimation protocol, which encodes the phase φ of an amplitude $\alpha = |\alpha| e^{i\varphi}$ into a state of an auxiliary

quantum system. Formally, we define the problem as follows. Given a unitary U and an eigenstate $|\phi\rangle$ such that

$$U |\phi\rangle = e^{i\varphi} |\phi\rangle \tag{3.53}$$

we want to estimate the phase φ. We show the protocol, which efficiently estimates the phase φ, in Fig. 3.7 and uses the inverse quantum Fourier transformation.
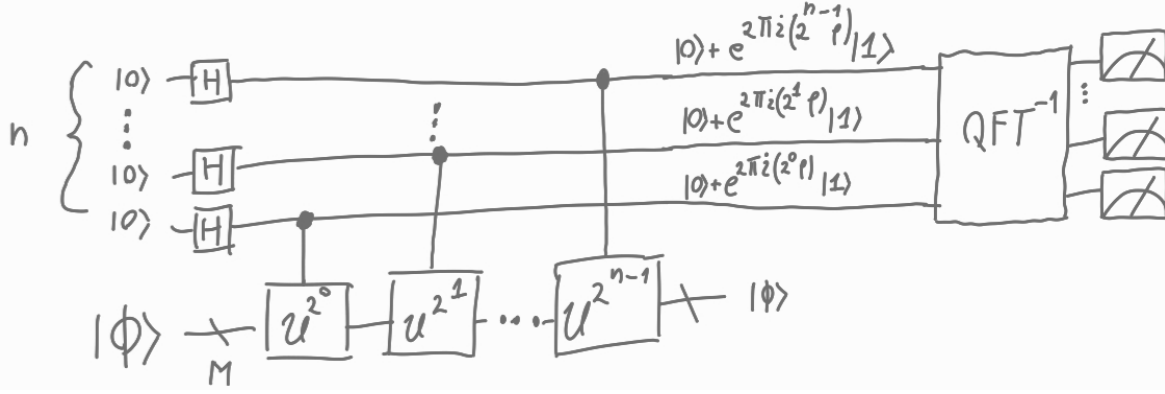

Figure 3.7: The circuit implementing the quantum phase estimation protocol

Here, we will focus only on the first part, which consists of n Hadamard transformations and n controlled unitary transformations with exponentially increasing duration (or power). The first controlled unitary after the Hadamard transformation transforms the state to

$$\frac{1}{\sqrt{2^n}}(|0\rangle + |1\rangle)^{\otimes n-1}(|0\rangle + e^{2\pi i\varphi} |1\rangle). \tag{3.54}$$

We used that $|\phi\rangle$ is an eigenstate of U, i.e. Eq. 3.53. After the second controlled unitary, we have the state

$$\frac{1}{\sqrt{2^n}}(|0\rangle + |1\rangle)^{\otimes n-2}(|0\rangle + e^{2\pi i 2^1 \varphi} |1\rangle)(|0\rangle + e^{2\pi i\varphi} |1\rangle). \tag{3.55}$$

The rest of the controlled unitary transformations add the phases in an exponentially increasing fashion, such that the state before the application of the inverse QFT is

$$\frac{1}{\sqrt{2^n}}(|0\rangle + e^{2\pi i 2^{n-1}\varphi} |1\rangle)(|0\rangle + e^{2\pi i 2^{n-2}\varphi} |1\rangle)\dots(|0\rangle + e^{2\pi i\varphi} |1\rangle) = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi i\varphi k} |k\rangle \tag{3.56}$$

Comparing Eq. 3.42 with the result of the Fourier transformation Eq. 3.42, we observe that the obtained state Eq. 3.42 encodes a Fourier transformation of the phase φ. To obtain an approximation of the phase, we must apply an inverse Fourier transformation and read out the bits in the auxiliary system. If $\varphi = b = 0.b_1 \dots b_n$ our final state will be $|b\rangle = |b_1 b_2 \dots b_n\rangle$ with probability 1.

To calculate the efficiency of the protocol in general, we have to estimate the probability of observing a state with an error larger than e. The state of the ancillary register after the inverse QFT is

$$\frac{1}{2^n} \sum_{k,l=0}^{2^n-1} e^{-2\pi i kl/2^n} e^{2\pi i\varphi k} |l\rangle . \tag{3.57}$$

Now consider the amplitude α_l of the state $|(b+l) \mod 2^n\rangle$

$$\alpha_l = \frac{1}{2^n} \sum_{k=0}^{2^n-1} \left(e^{2\pi i(\varphi-(b+l)/(2^n))} \right)^k . \tag{3.58}$$

38

The sum in Eq. 3.58 is a geometric series, hence

$$\alpha_l = \frac{1}{2^n} \frac{1 - e^{2\pi i (2^n \varphi - (b+l))}}{1 - e^{2\pi i (\varphi - (b+l)/2^n)}} = \frac{1}{2^n} \frac{1 - e^{2\pi i (2^n \delta - l)}}{1 - e^{2\pi i (\delta - l/2^n)}}, \tag{3.59}$$

where $\delta = \varphi - 0.b_1 b_2 \ldots b_n$. We will estimate the probability of obtaining the value m such that $|m - b| > e$, where e is a positive integer characterising our desired tolerance to error. The probability of observing such an m is given by

$$p(|m - b| > e) = \sum_{-2^{n-1} < l \le -(e+1)} |\alpha_l|^2 + \sum_{e+1 \le l \le 2^{n-1}} |\alpha_l|^2 \tag{3.60}$$

For any real θ, we have $|1 - \exp i\theta| \le 2$ so

$$|\alpha_l| \le \frac{1}{2^n} \frac{2}{|1 - e^{2\pi i (\delta - l/2^n)}|} \le \frac{1}{2^{n+1} |\delta - l/2^n|}. \tag{3.61}$$

We obtain the last inequality using $|1 - \exp(i\theta)| \ge 2\theta/\pi$. Combining the last two expressions, we get the bound

$$p(|m - b| > e) \le \frac{1}{4} \left[\sum_{l=-2^{n-1}+1}^{-(e+1)} \frac{1}{(2^n \delta - l)^2} + \sum_{l=e+1}^{2^{n-1}} \frac{1}{(2^n \delta - l)^2} \right] \tag{3.62}$$

Since $0 < 2^n \delta < 1$ we have

$$p(|m - b| > e) \le \frac{1}{4} \left[\sum_{l=-2^{n-1}+1}^{-(e+1)} \frac{1}{l^2} + \sum_{l=e+1}^{2^{n-1}+} \frac{1}{(l-1)^2} \right] \tag{3.63}$$

$$\le \frac{1}{2} \sum_{l=e}^{2^{n-1}-1} \frac{1}{l^2} \tag{3.64}$$

$$\le \frac{1}{2} \int_{e-1}^{2^{n-1}-1} dl \frac{1}{l^2} \tag{3.65}$$

$$= \frac{1}{2(e-1)}. \tag{3.66}$$

Suppose we want to estimate φ up to 2^{-t}, which translates to $e = 2^{n-t} - 1$. The probability of getting such a state is at least $1 - \frac{1}{2(2^{n-t}-2)}$. Thus, to successfully obtain φ to t bits with a probability of success at least $1 - \epsilon$, we need to use $n = t + \lfloor \log(2 + 1/2\epsilon) \rfloor$ qubits.

As noted above, many exponentially fast algorithms use the quantum phase estimation protocol. The most famous are order finding and factoring. The second is essential since most of the current internet security is based on the 2048 RSA key encryption, which can be decoded efficiently using the factoring algorithm. However, classical security protocols exist which can not be efficiently decrypted even with a quantum algorithm. A field studying these types of problems is post-quantum cryptography.

Besides the mentioned algorithms, the QPE can perform many linear algebra transformations efficiently. These can then accelerate many classical machine learning algorithms.

Matrix multiplication The first linear algebra routine we can speed up using the QPA protocol is vector-matrix multiplication. This protocol applies to Hermitian matrices A with eigenvalues λ_r smaller than one. Classically, we can write

$$A v_r = \lambda_r v_r, \quad x = \sum_{r=1}^{R} (v_r^T x) v_r \Rightarrow A x = \sum_{r=1}^{R} \lambda_r (v_r^T x) v_r. \tag{3.67}$$
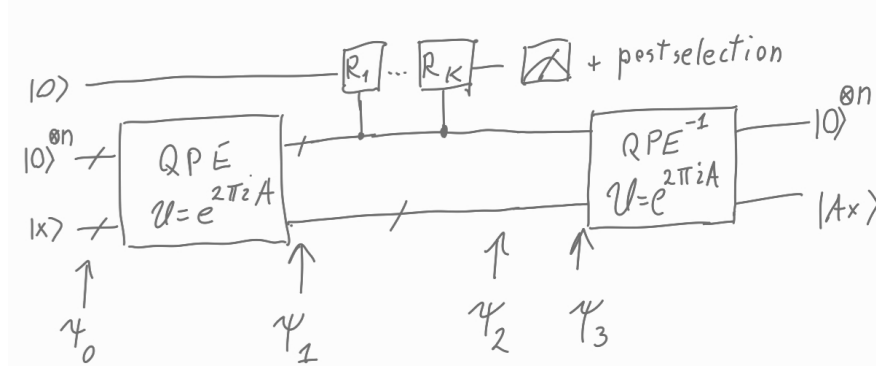
Figure 3.8: Matrix multiplication and matrix inversion protocols. The difference between the protocols is the choice of the rotations $R_1 \dots R_K$.

The quantum formulation of the problem is as follows. Given a quantum state $|x\rangle$, we want to calculate the state $|Ax\rangle$. More formally, we want to find the transformation

$$|x\rangle = \sum_{r=1}^{R} \langle v_r | x \rangle |v_r\rangle \rightarrow \sum_{r=1}^{R} \lambda_r \langle v_r | x \rangle |v_r\rangle . \tag{3.68}$$

The corresponding quantum matrix multiplication protocol is shown in Fig. 3.8. We start the protocol in a state $|\psi_0\rangle = |0\rangle |0\rangle^n |x\rangle$ and transform it with a quantum phase transformation protocol related to the unitary transformation $U = \mathrm{e}^{2\pi \mathrm{i}A}$. To determine the state $|\psi_1\rangle$ we recall the action of the QPA protocol on an eigenstate QPA $\left(U = \mathrm{e}^{2\pi \mathrm{i}A}\right) |0\rangle^{\otimes n} |v_r\rangle \sim |\lambda_r\rangle |v_r\rangle$. Therefore we have

$$|\psi_1\rangle \sim \sum_{r=1}^{R} |0\rangle |\lambda_r\rangle \langle v_r | x \rangle |v_r\rangle . \tag{3.69}$$

In the next step, we use conditional rotations to rotate the first auxiliary qubit to a state $\sqrt{1 - \lambda_r^2} |0\rangle + \lambda_r |1\rangle$. This can be done by applying $K = n$ conditional rotations, which successively move the state $|0\rangle$ to the state $|1\rangle$. For example, the k-th rotation would move the current state of the auxiliary qubit by $1/2^k$ if $q_k = 1$, where $\lambda_r = 0.q_1 q_2, \dots q_n$. After the rotations, we get the state

$$|\psi_2\rangle \sim \sum_{r=1}^{R} (\sqrt{1 - \lambda_r^2} |0\rangle + \lambda_r |1\rangle) |\lambda_r\rangle \langle v_r | x \rangle |v_r\rangle . \tag{3.70}$$

Next, we measure the first auxiliary qubit. If we observe the state $|1\rangle$, we proceed with the calculation; otherwise, we start from the beginning. This process is called post-selection. The successful state after the post-selection is

$$|\psi_3\rangle \sim \sum_{r=1}^{R} \lambda_r |1\rangle |\lambda_r\rangle \langle v_r | x \rangle |v_r\rangle \tag{3.71}$$

This state is almost what we need to get. The problem is that the auxiliary qubits are entangled with the main qubits. We can disentangle them by using the inverse QPA algorithm. Finally, we get

$$|1\rangle |0\rangle^{\otimes n} \sum_{r=1}^{R} \lambda_r \langle v_r | x \rangle |v_r\rangle , \tag{3.72}$$

which is exactly the desired result.

40

Matrix inverse The matrix inverse differs from the matrix multiplication only in the rotation part. Instead of the transformation leading to Eq. 3.70, we use the rotations that implement the "inverse" transformation

$$|0\rangle\,|\lambda\rangle \to (\sqrt{1 - C/\lambda_r}\,|0\rangle + C/\lambda_r\,|1\rangle). \tag{3.73}$$

Since $|\lambda_r| < 1$, we have to use a constant essentially determined by the smallest eigenvalue of A.

The postselection procedure is a non-unitary operation which requires repeating the experiment $\mathcal{O}(1/p_{\mathrm{acc}})$ times. For matrix multiplication the acceptance probability is $p_{\mathrm{acc}} = \sum_r \lambda_r^2 |\langle v_r|x\rangle|^2$. For the matrix inversion the acceptance probability is $p_{\mathrm{acc}} = \sum_r \frac{C^2}{\lambda_r^2} |\langle v_r|x\rangle|^2 \leq \kappa^{-2}$, where κ is the conditional number of the matrix. The quantum algorithm will use many tries to succeed for a badly conditioned matrix. We can circumvent this by using preconditioning techniques.

qBlas Besides matrix multiplication and matrix inversion, several important linear algebra routines have been translated to the quantum domain. Most important are the singular value decomposition and density matrix exponentiation. These routines have logarithmic requirements regarding data size and number of examples. However, they can typically be used only with subtle restrictions. We will put the limitations aside and focus on the machine learning implications. We can not immediately transform many machine learning algorithms into their quantum versions. We often have to reformulate them to a less efficient version on which we can then apply quantum linear algebra routines, such that the final quantum protocol still represents a significant speedup with respect to the original algorithm. Prominent machine learning algorithms that achieve a possible exponential speed-up are principal component analysis, support vector machines, cluster finding, topological data analysis, and Gaussian processes.

References

- Schuld, M., & Petruccione, F. (2021). Machine learning with quantum computers (Vol. 676, pp. 163-169). Berlin: Springer. (Chapter: Fault-Tolerant Quantum Machine Learning)
- Nielsen, M. A., & Chuang, I. L. (2010). Quantum computation and quantum information. Cambridge University Press. (Chapter 5.1: The quantum Fourier transform, Chapter 5.2: Phase estimation)

3.5 Quantum search

The second branch of algorithms, which achieves less dramatic but more rigorous/strict speedups, is the quantum search algorithm. Finding one or more examples in an unstructured database with N samples can classically be solved in time $\mathcal{O}(N)$. Remarkably, a quantum algorithm (Grover's search) can do that in time $\mathcal{O}(\sqrt{N})$.

Suppose you have a quantum unitary/oracle O implementing the classical function $f(x) : \{0,1\}^{\otimes n} \to \{0,1\}$

$$|x\rangle\,|y\rangle \xrightarrow{O} |x\rangle\,|y \oplus f(x)\rangle. \tag{3.74}$$

The task is to find an example x for which $f(x) = 1$. We can achieve this in $\mathcal{O}(\sqrt{N/M})$ steps by performing the algorithm presented in Fig. 3.9. The initial state of the algorithm is $|\phi_0\rangle = |0\rangle^n\,|0\rangle$. Like many quantum protocols, the algorithm starts with applying the n-fold tensor product of Hadamard gates on the first n qubits and a bit flip, followed by a Hadamard gate on the last qubit. These transformations result in the
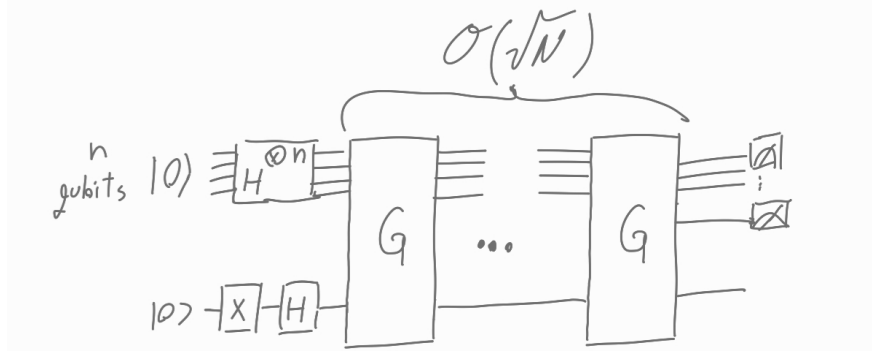
Figure 3.9: Circuit implementing the Grover search algorithm.

state

$$|\psi_1\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \tag{3.75}$$

Then we apply several Grover rotations $G = ((2|\psi\rangle\langle\psi| - \mathbb{1}_{2^n}) \otimes \mathbb{1}_2)O$, where $|\psi\rangle = \frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} |x\rangle$ and O is the unitary defined in Eq. 3.74. We can understand the Grover rotation G geometrically. First, we write the state $|\psi\rangle$ as

$$|\psi\rangle = \sqrt{\frac{N-M}{N}} |\alpha\rangle + \sqrt{\frac{M}{N}} |\beta\rangle \tag{3.76}$$

$$= \cos\frac{\theta}{2} |\alpha\rangle + \sin\frac{\theta}{2} |\beta\rangle . \tag{3.77}$$

The state $|\alpha\rangle$ is given by a superposition of $N - M$ states for which $f(x) = 0$. Similarly, the state $|\beta\rangle$ represents a superposition of M states for which $f(x) = 1$. We now consider the action of the Grover rotation in the basis spanned by the vectors $|\alpha\rangle, |\beta\rangle$. The application of the oracle unitary O results in a reflection around the α axis (see Fig. 3.10), namely

$$O(a|\alpha\rangle + b|\beta\rangle)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = a|\alpha\rangle\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) + b|\beta\rangle\frac{1}{\sqrt{2}}(|1\rangle - |0\rangle) = (a|\alpha\rangle - b|\beta\rangle)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \tag{3.78}$$

Similarly the application of $2|\psi\rangle\langle\psi| - \mathbb{1}_{2^n}$ results in a reflection around the state $|\psi\rangle$. The two actions do not leave the manifold spanned by the states $|\alpha\rangle$ and $|\beta\rangle$ and constitute a rotation. From Fig. 3.10 it is clear that their combination implements a rotation

$$G|\psi\rangle = \cos\frac{3\theta}{2}\alpha + \sin\frac{3\theta}{2} |\beta\rangle . \tag{3.79}$$

The state after K application of G is

$$G^K |\psi\rangle = \cos\frac{2K+1}{2}\theta |\alpha\rangle + \sin\frac{2K+1}{2}\theta |\beta\rangle . \tag{3.80}$$

Finally, we measure the first n qubits. If the angle $\frac{2K+1}{2}\theta$ is equal to $\pi/2$, we will likely observe one of the states in $|\beta\rangle$, which is the desired result. It follows that we have to perform $R \approx \left\lfloor \frac{\pi}{4}\sqrt{\frac{N}{M}} \right\rfloor$ operations to get the desired states with a high probability.
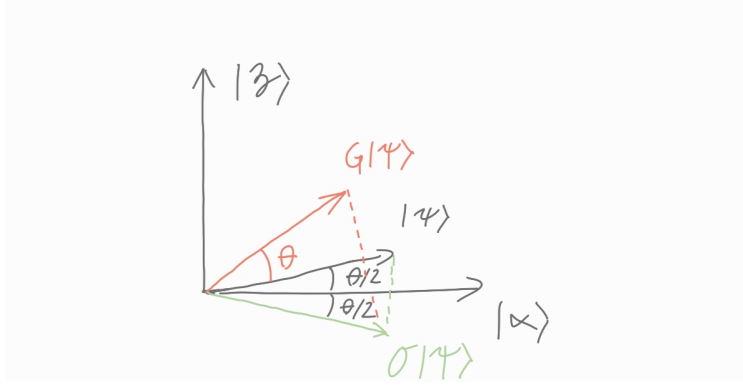
Figure 3.10: Geometric view of the Grover rotation.

References

- Schuld, M., & Petruccione, F. (2021). Machine learning with quantum computers (Vol. 676, pp. 163-169). Berlin: Springer. (Chapter: Quantum computing)

- Nielsen, M. A., & Chuang, I. L. (2010). Quantum computation and quantum information. Cambridge University Press. (Chapter 6.1: The quantum search algorithm)

3.6 Deterministic Quantum Computation with 1 Qubit

In the previous example, we used a pure quantum state of n qubits to achieve a quantum speedup. This is the case for most quantum algorithms. A notable exception is the deterministic Quantum Computation with One Clean Qubit (DQC1) model, introduced by Knill and Laflamme (1998). The DQC1 model of quantum computation challenges the conventional belief that quantum computation requires highly entangled pure states.

The standard setup of the DQC1 model is shown in Fig. 3.11 and consists of one pure qubit in a known quantum state $|0\rangle$ and $n-1$ qubits in a completely mixed state $\rho = \mathbb{1}/2^{n-1}$. We then perform a unitary on all n qubits and measure the expectation value of the observable σ^z on the first clean qubit. In the most basic example, the unitary implements the Hadamard test, which is given by Hadamard on the first qubit, followed by a controlled unitary (the first qubit is the control) and another Hadamard on the first qubit, namely

$$U = (H \otimes \mathbb{1}_{n-1})CU(H \otimes \mathbb{1}_{n-1}), \tag{3.81}$$

where CU is a controlled unitary, the first qubit is the control, and the unitary is applied to the rest of the $n-1$ qubits. To calculate the action of the unitary U on a mixed state, we first calculate its action on a
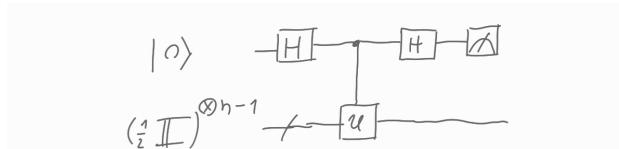

Figure 3.11: A circuit representation of the DQC1 computational model; a Hadamard test on a mixed state.

pure basis state and then invoke the linearity of the map. The action of the unitary U, also known as the Hadamard test, on a pure state $|0\rangle|\psi\rangle$ is

$$U|0\rangle|\psi\rangle = \frac{1}{\sqrt{2}}(H \otimes \mathbb{1}_{n-1})CU(|0\rangle + |1\rangle)|\psi\rangle \tag{3.82}$$

$$= \frac{1}{\sqrt{2}}\left((H \otimes \mathbb{1}_{n-1})(|0\rangle|\psi\rangle + |1\rangle U|\psi\rangle)\right)$$

$$= \frac{1}{2}\left((|0\rangle + |1\rangle)|\psi\rangle + (|0\rangle - |1\rangle)U|\psi\rangle\right)$$

$$= \frac{1}{2}\left(|0\rangle(1+U)|\psi\rangle + |1\rangle(1-U)|\psi\rangle\right)$$

The probabilities P_j of observing the first qubit in state $|j\rangle$ for $j = 0, 1$ are

$$P_0 = \frac{1}{2}(1 + \mathrm{Re}(\langle\psi|U|\psi\rangle)), \tag{3.83}$$

$$P_1 = \frac{1}{2}(1 - \mathrm{Re}(\langle\psi|U|\psi\rangle)).$$

The expectation value $\langle\sigma^z\rangle$ in the Hadamard test with a pure initial state is then

$$\langle\sigma^z\rangle = \mathrm{Re}(\langle\psi|U|\psi\rangle) \tag{3.84}$$

By linearity, we have in the case of the DQC1 circuit (a Hadamard test of a completely mixed initial state)

$$\langle\sigma^z\rangle = \frac{1}{2^{n-1}}\sum_{k=0}^{2^{n-1}-1}\mathrm{Re}(\langle k|U|k\rangle) = \frac{1}{2^{n-1}}\mathrm{Re}(\mathrm{Tr}\,U). \tag{3.85}$$

Since we can calculate the trace of an exponentially large matrix with only a few gates, we have demonstrated that a quantum computer with only one "clean" qubit (fully quantum-coherent) and all others in maximally mixed states can still efficiently solve problems believed to be classically intractable. DQC1 thus serves as a key model for studying the boundary between classical and quantum computation, offering insights into how minimal quantum resources can still yield a computational advantage.

References

- Knill, E., & Laflamme, R. (1998). Power of one bit of quantum information. Physical Review Letters, 81(25), 5672.

3.7 Adiabatic quantum computation

Adiabatic Quantum Computation (AQC) is a distinct quantum computing model that differs from the circuit model that we have discussed so far by evolving a quantum system from an initial Hamiltonian with an easily prepared ground state to a final Hamiltonian whose ground state encodes the solution. The adiabatic theorem ensures the system remains in its ground state if the evolution is slow enough. AQC has strong ties to condensed matter physics, computational complexity, and heuristic algorithms.

The concept originated in the late 1980s as quantum stochastic optimisation, later renamed quantum annealing, which was inspired by simulated annealing. Early research suggested quantum annealing could outperform classical methods, leading to experimental implementations and the development of the quantum adiabatic algorithm. While optimisation problems typically use stoquastic Hamiltonians, non-stoquastic AQC is as powerful as the standard circuit-based quantum model, making AQC a general framework beyond optimisation.

Quantum Adiabatic Theorem In this section, we present an elementary argument for the quantum adiabatic theorem, which serves as the cornerstone of AQC.

Consider a time-dependent Hamiltonian $H(t)$ with an initial state $|\psi(0)\rangle = |\phi_k(0)\rangle$, where $|\phi_k(0)\rangle$ is an instantaneous eigenstate of the Hamiltonian at time $t = 0$. The adiabatic theorem states that if the Hamiltonian evolves slowly compared to the energy gaps, the time-evolved state remains approximately proportional to the instantaneous eigenstate. More precisely, the deviation satisfies

$$\||\psi(t)\rangle - |\phi_k(t)\rangle\|_2 = \mathcal{O}\left(\frac{1}{T}\right), \quad 0 < t < T, \tag{3.86}$$

where T is the annealing time.

To formalise this, we define the instantaneous eigenstates of the time-dependent Hamiltonian:

$$H(t)\,|\psi_l(t)\rangle = E_l(t)\,|\psi_l(t)\rangle. \tag{3.87}$$

Expanding the state $|\psi(t)\rangle$ in terms of these eigenstates, we write

$$|\psi(t)\rangle = \sum_k c_k(t)\,|\psi_k(t)\rangle. \tag{3.88}$$

Substituting this into the Schrödinger equation

$$i\partial_t\,|\psi(t)\rangle = H(t)\,|\psi(t)\rangle, \tag{3.89}$$

we obtain

$$i\sum_k \dot{c}_k(t)\,|\psi_k(t)\rangle + c_k(t)\,\big|\dot{\psi}_k(t)\big\rangle = \sum_k c_k(t) E_k(t)\,|\psi_k(t)\rangle. \tag{3.90}$$

Projecting onto $\langle\psi_l(t)|$, we derive

$$i\dot{c}_l(t) = E_l(t)c_l(t) - i\sum_k \big\langle\psi_l(t)\big|\dot{\psi}_k(t)\big\rangle c_k(t) \tag{3.91}$$

$$= \big[E_l(t) - i\big\langle\psi_l(t)\big|\dot{\psi}_l(t)\big\rangle\big] c_l(t) - i\sum_{k\neq l} \big\langle\psi_l(t)\big|\dot{\psi}_k(t)\big\rangle c_k(t).$$

Neglecting the last term under the adiabatic approximation, we arrive at

$$i\dot{c}_l(t) = \big[E_l(t) - i\big\langle\psi_l(t)\big|\dot{\psi}_l(t)\big\rangle\big] c_l(t). \tag{3.92}$$

This differential equation has the solution

$$c_l(t) = e^{-i\int_0^t \big[E_l(u) - i\big\langle\psi_l(u)\big|\dot{\psi}_l(u)\big\rangle\big]\mathrm{d}u} c_l(0). \tag{3.93}$$

Now, assume the system starts in the state defined by $c_l(0) = 1$ with all other coefficients initially zero. To justify neglecting the term in (3.93), we differentiate the eigenvalue equation (3.87) with respect to time:

$$\dot{H}\,|\psi_k\rangle + H\,\big|\dot{\psi}_k\big\rangle = \dot{E}_k\,|\psi_k\rangle + E_k\,\big|\dot{\psi}_k\big\rangle. \tag{3.94}$$

Taking the inner product with $\langle\psi_l|$ for $l \neq k$ yields

$$\langle\psi_l|\,H\,\big|\dot{\psi}_k\big\rangle + E_k\,\big\langle\psi_l\big|\dot{\psi}_k\big\rangle = E_k\,\big\langle\psi_l\big|\dot{\psi}_l\big\rangle. \tag{3.95}$$

From this, we extract

$$\big\langle\psi_l\big|\dot{\psi}_l\big\rangle = \frac{\langle\psi_l|\,\dot{H}\,|\psi_k\rangle}{E_k - E_l} = \frac{\dot{H}_{lk}}{E_k - E_l}. \tag{3.96}$$

For a time-dependent Hamiltonian of the form

$$H(t) = H_0 + \frac{t}{T}V, \quad 0 < t < T, \tag{3.97}$$

where H_0 is the initial Hamiltonian, V is an interaction term, and T is the annealing time, we find

$$\dot{H}_{lk} = \frac{1}{T}V_{lk}. \tag{3.98}$$

Thus, if the change is sufficiently slow (large T), the neglected term in (3.93) remains small, ensuring that the time-evolved state remains close to the instantaneous eigenstate.

Example 3.4. Consider the time-dependent Hamiltonian:

$$H(t) = \begin{bmatrix} \frac{\alpha t}{2} & \Delta \\ \Delta & -\frac{\alpha t}{2} \end{bmatrix}, \tag{3.99}$$

where α is the sweep rate and Δ is the coupling between the two states. The instantaneous eigenvalues (energy levels) of the Hamiltonian are given by:

$$E_{\pm}(t) = \sqrt{\Delta^2 + \frac{\alpha^2 t^2}{4}}. \tag{3.100}$$

Figure 3.12 illustrates these instantaneous energy levels and the relevant time scales where the eigenvector rotation occurs. The characteristic time scale

$$\tau_d = \frac{\Delta}{\alpha} \tag{3.101}$$

determines the duration over which the eigenvector transformation takes place. For an adiabatic transition, this time scale must be much larger than the inverse of the energy gap $\left(\frac{1}{2\Delta}\right)$, leading to the adiabatic condition:

$$\frac{\Delta^2}{\alpha} \gg 1. \tag{3.102}$$

We now focus on the probability of a *non-adiabatic* transition from the ground state to the excited state. While the full derivation is lengthy, we state the well-known result. The probability of a non-adiabatic transition is given by:

$$P_{\text{n.ad.}} = \exp\left(-2\pi \frac{\Delta^2}{\alpha}\right). \tag{3.103}$$

This result is consistent with our heuristic adiabatic condition: as the time scale τ_d increases while keeping the energy gap 2Δ constant, the transition probability decreases exponentially, ensuring adiabatic evolution.

References

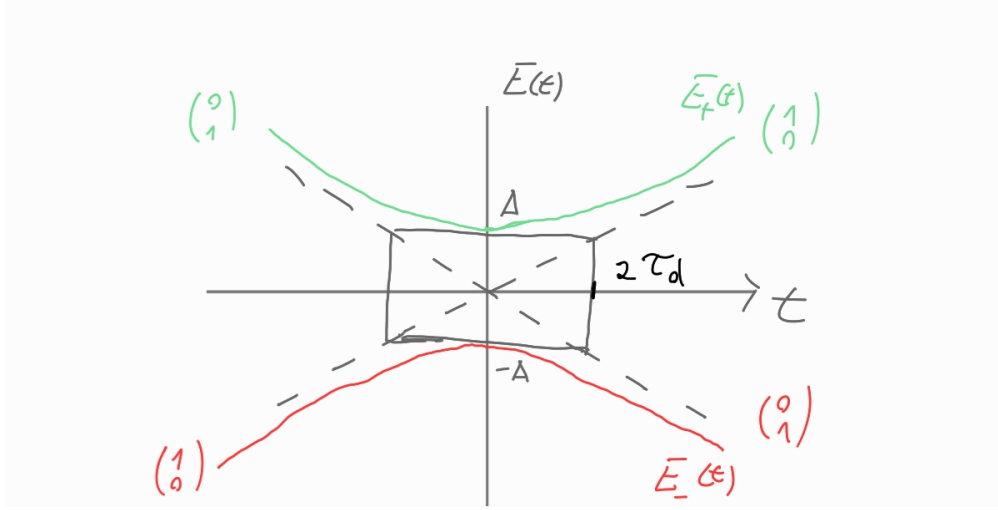- MITOpenCourseWare: Chapter 6: Adiabatic approximation

Figure 3.12: The Landau-Zenner energy diagram. The vectors at the ends of the energy lines $E_\pm$ correspond to asymptotic eigenvectors of the Hamiltonian $H(t)$ at times $t = \pm\infty$.

4 Quantum computing with data

The previous section covered essential quantum algorithms applicable to universal fault-tolerant quantum devices with at least 10^6 qubits. The best current quantum devices are noisy and have around a few hundred qubits. Hence, the name Noisy Intermediate Scale Quantum (NISQ) devices. It is also unclear how fast we can scale up quantum devices and make them fault-tolerant. Therefore, most of the current research is focused on algorithms that apply to NISQ devices and offer significant benefits with respect to the best classical algorithms. A new hybrid, variational, quantum-classical algorithm is emerging in this respect. The limitations of NISQ devices also force us to consider the data encoding schemes, i.e., how to encode classical information on a quantum computer.

This section will first discuss the data encoding strategies and then introduce the variational quantum circuits as the leading candidates for a practical algorithm on NISQ devices.

4.1 Data encoding

We divide the encoding strategies into four groups: *basis encoding, amplitude encoding, qusample encoding,* and *Hamiltonian encoding.* Each of the strategies applies to certain types of problems. They also differ in storage capacity and computational complexity. Table 4 shows an overview of the runtime complexity of the various encoding schemes discussed in the following.

Encoding	Number of qubits	Runtime of state prep	Input features
Basis	N	$\mathcal{O}(MN)$	Binary
Amplitude	$\log N$	$\mathcal{O}(MN)/\mathcal{O}(\log(MN))$	Continuous
Qusample	N	$\mathcal{O}(2^N)/\mathcal{O}(N)$	Binary
Hamiltonian	$\log N$	$\mathcal{O}(MN)/\mathcal{O}(\log(MN))$	Continuous

Table 4: Storage capacity and encoding runtime complexity of the encoding strategies for the dataset of M samples with N features.

Basis encoding Basis encoding is the backbone of many algorithms for universal quantum devices. Assume we have a dataset $\mathcal{D}$ with M samples. Each sample is an N dimensional binary vector $x^m = (b_1^m, b_2^m, \ldots, b_N^m)$, where $b_j^m \in \{0, 1\}$, $j = 1, 2 \ldots, N$, and $m = 1, 2, \ldots, M$. We can encode the dataset as a quantum superposition as

$$|\mathcal{D}\rangle = \frac{1}{\sqrt{M}} \sum_{m=1}^{M} |b_1^m, b_2^m, \ldots, b_N^m\rangle. \tag{4.1}$$

If the features are not binary, we must first convert them to binary codes and then use them as binary features.

Example 4.1. Assume we have a dataset $\mathcal{D} = \{x^1, x^2\}$ with vectors $x^1 = (2, 0)$ and $x^2 = (1, 3)$. We first convert the features to binary representations and then concatenate the representations of the first and the second feature of the vectors

$$(2, 0) \rightarrow ((0, 1), (0, 0)) \rightarrow (0, 1, 0, 0) \tag{4.2}$$
$$(1, 3) \rightarrow ((1, 0), (1, 1)) \rightarrow (1, 0, 1, 1) \tag{4.3}$$

We can now use the basis encoding for binary strings defined in Eq. 4.1. The quantum state representing the two vectors is then

$$|\mathcal{D}\rangle = \frac{1}{\sqrt{2}} (|0, 1, 0, 0\rangle + |1, 0, 1, 1\rangle). \tag{4.4}$$

The benefit of encoding is that it is a simple preparation of single states since we have to apply only a local bit flip. Efficient preparation routines have also been developed for a superposition of states, i.e. a dataset. The main disadvantage of the basis encoding is that we need many qubits to represent features. For example, for one floating-point number, we need 16 qubits.

Amplitude encoding Also, amplitude encoding is important for many quantum algorithms. In particular, the algorithms that apply the qBLAS routines. Assume we have a dataset $\mathcal{D}$ with M samples of N dimensional real vectors $x^m = (x_1^m, x_2^m, \ldots, x_N^m)$, where $x_j^m \in \mathbb{R}$, $j = 1, 2 \ldots, N$, and $m = 1, 2, \ldots, M$. We encode the dataset $\mathcal{D}$ in a quantum state as

$$\mathcal{D} = \frac{1}{\sqrt{M}} \sum_{m=1}^{M} \sum_{j=1}^{N} x_j^m |j\rangle |m\rangle = \frac{1}{\sqrt{M}} \sum_{m=1}^{M} |\psi_{x^m}\rangle |m\rangle. \tag{4.5}$$

This encoding needs MN orthogonal vectors, which can be achieved with $\log MN$ qubits. Due to the normalisation of the quantum state, the states $|\psi_{x^m}\rangle$ and hence the vectors x^m need to be L_2 normalised. We can overcome this restriction by adding a qubit that encodes the norm of the entire vector. Efficient, i.e. linear in MN, schemes have been developed to encode the classical dataset in a quantum state of the form Eq. 4.5. The benefits of amplitude encoding are the efficient use of qubits and the fact that there is no restriction on the precision of the calculations. On the other hand, we can not easily access the information presented in the amplitudes. In general, we need exponentially many measurements (in the number of qubits) to determine the state $|\mathcal{D}\rangle$.

Qsample encoding Q-sample encoding is a mixture of basis and amplitude encoding. It encodes a classical probability distribution $(p_1, p_2, \ldots, p_N)$. The square-root probabilities $\sqrt{p_j}$ are encoded in an

amplitude encoding of the basis vectors $|j\rangle$ represented in the basis encoding $j \to |j\rangle = \left|j_1, j_2 \ldots, j_{\lfloor \log N \rfloor}\right\rangle$. We encountered this encoding in the first section, where we introduced the quantum probabilities as an interesting way to encode classical probability distributions. It has the benefit that the marginal distributions (reduced density matrices) encode the conditional probabilities. Hence, it is possible to reconstruct the joint probability by knowing only the marginal probabilities (see Example. 2.15). Further, it is trivial to obtain marginal probabilities and perform rejection sampling.

Example 4.2. Rejection sampling is related to branch selection (or post-selection). In a branch selection, we have a state of the form

$$\sum_{i=1}^{N} \sqrt{a_i} \, |i\rangle \, (\sqrt{1 - \sqrt{b_i}} \, |0\rangle + \sqrt{b_i} \, |1\rangle). \tag{4.6}$$

We assume a_i, b_i are real numbers in the unit interval $[0, 1]$. By measuring the register $|i\rangle$ we sample from a distribution $(a_1, a_2, \ldots, a_N)$, but we want to sample from a distribution $(a_1 b_1, a_2 b_2, \ldots, a_N b_N)$. We can do that by first sampling the auxiliary qubit. If we observe a state $|0\rangle$, we disregard the state, but if we observe $|1\rangle$, we continue with sampling the main register $|i\rangle$. The second sampling is then performed according to the desired distribution. The acceptance rate of the postselection is given by the square amplitude of the selected branch

$$p_{\text{acc}} = \sum_{i=1}^{N} b_i a_i. \tag{4.7}$$

We now identify the distribution given by $\{a_j\}$ as the easy-to-sample distribution (p_j) and the distribution given by $\{a_j b_j\}$ as the complex distribution q_j. The described branch selection is analogous to the rejection sampling of q_j, where the samples are drawn from the probability p_j.

Hamiltonian encoding   Hamiltonian encoding determines the state implicitly by specifying the unitaries with which we transform the fixed initial state. Given a Hermitian matrix $H \in \mathcal{R}^{2^n \times 2^n}$, we encode it as a unitary transformation determined by $U_H = e^{iH}$. If the matrix is unitary, we can consider it a quantum transformation. If a matrix A does not have a special form, we can still apply the Hamiltonian encoding on a matrix

$$H = \begin{pmatrix} 0 & A \\ A^\dagger & 0 \end{pmatrix}. \tag{4.8}$$

To apply Hamiltonian encoding, we need to simulate the unitaries of the form e^{iH}. The problem of simulating such unitaries is called Hamiltonian simulation and can be done efficiently with local gates (see Solovay-Kitaev theorem in Section 3.2). However, the best decomposition of a given unitary in terms of local gates can depend on a given implementation of the NISQ device.

References

- Schuld, M., & Petruccione, F. (2021). Machine learning with quantum computers (Vol. 676, pp. 163-169). Berlin: Springer. (Chapter: Representing Data on a Quantum Computer)

4.2 Block encoding and linear combination of unitaries

Block encoding is a powerful technique that unifies many quantum algorithms. It is based on two ideas:

- Any matrix (apart from a normalisation) can be embedded into the top left part of a unitary matrix

- We can selectively apply a non-unitary operator to a state by postselecting on the desired outcome of an auxiliary qubit measurement.

We can concisely write the two ideas with the following equation

$$U = \begin{bmatrix} A/\alpha & \bullet \\ \bullet & \bullet \end{bmatrix} \quad \Rightarrow \quad A = \alpha \left(\langle 0 | \otimes \mathbb{1} \right) U \left(| 0 \rangle \otimes \mathbb{1} \right) \tag{4.9}$$

More precisely, we say that the $(s + a)$-qubit unitary U is an (α, a, ε) block encoding of A if

$$\| A - \alpha \left(\langle 0 | \otimes \mathbb{1} \right) U \left(| 0 \rangle \otimes \mathbb{1} \right) \|_2 \leq \varepsilon, \tag{4.10}$$

where A acts on s qubits and U is implemented using additional a qubits for selecting the correct block of the unitary matrix.

Since there are exponentially many different matrices, we can not efficiently block encode all possible A in the number of qubits s. However, we can efficiently encode some interesting operators.

Here, we will consider the block encoding of a linear combination of unitaries. To streamline the derivation of the block encoding, we note that the space of linear operators is itself a Hilbert space with a Hilbert-Schmidt inner product

$$\langle A, B \rangle = \frac{1}{2^n} \text{Tr}\, A^\dagger B, \tag{4.11}$$

where A, B are $2^n \times 2^n$ matrices, and the factor in front of the trace is chosen for easier normalisation of Pauli operators. If we vectorise the matrices, we find that the Hilbert-Schmidt inner product is the same (apart from the particular normalisation) as the inner product of vectorised matrices. We remark that the Pauli strings, i.e. the 4^n operators of the form $\sigma^{j_1} \otimes \sigma^{j_2} \otimes \ldots \otimes^{j_n}$, where $\sigma^0 = \mathbb{1}$, $\sigma^1 = \sigma^{\mathrm{x}}$, $\sigma^2 = \sigma^{\mathrm{y}}$, $\sigma^3 = \sigma^{\mathrm{z}}$, are the orthonormal bases of operators acting on $\mathcal{C}^{2^n}$ (see Example. 4.3). Therefore, any operator A can be decomposed as a linear combination of Pauli strings,

$$A = \sum_{k_1, k_2, \ldots k_n} \langle \sigma^{k_1} \sigma^{k_2} \ldots \sigma^{k_n}, A \rangle \sigma^{k_1} \otimes \sigma^{k_2} \otimes \ldots \otimes \sigma^{k_n} \tag{4.12}$$

Since Pauli strings are also unitaries, we can express any matrix as a sum of unitaries. Therefore, block encoding of `linear combinations of unitaries` enables us to block encode any matrix.

Linear combination of unitaries (LCU): Assume we have $A = \sum_j \alpha_j U_j$ for $\alpha_j > 0$ where the matrices A, U_j act on the 2^s dimensional space. To block encode A, we need a way to specify the expansion coefficients α and the corresponding unitaries. We can do this by preparing and selecting circuits/oracles. We define the `Prepare` circuits acting on the 2^a dimensional auxiliary space as

$$Prep\, |\underline{0}\rangle = \sum_i \frac{\sqrt{\alpha_i}}{\sqrt{\alpha}} |i\rangle, \quad \alpha = \sum_i \alpha_i. \tag{4.13}$$

Note that we only defined the action on the zero vector $|\underline{0}\rangle = |0\rangle \otimes |0\rangle \otimes \ldots \otimes |0\rangle$. Only this part will be important when constructing the block encoding. Therefore, we can choose the rest of the oracle/circuit to

optimise the actual implementation on a quantum device. Notice that the *Prep* circuit provides the information about the expansion coefficients. The second circuit, *Select*, provides the complementary information about the corresponding unitaries

$$Select = \sum_i |i\rangle \langle i| \otimes U_i. \tag{4.14}$$

While *Prep* acts only on the 2^a–dimensional auxiliary space, the *Select* circuits act on the full 2^{a+s}–dimensional space. We can block encode the matrix A with the following circuit and postselection

$$BlockEncoding(A) = (\langle 0| \otimes \mathbb{1})Prep^\dagger \ Select \ Prep(|0\rangle \otimes \mathbb{1}) = \tag{4.15}$$

$$= \left(\sum_i \frac{\sqrt{\alpha_i}}{\sqrt{\alpha}} \langle i| \otimes \mathbb{1} \right) Select \left(\sum_j \frac{\sqrt{\alpha_j}}{\sqrt{\alpha}} |j\rangle \otimes \mathbb{1} \right)$$

$$= \sum_{i,j} \frac{\sqrt{\alpha_i \alpha_j}}{\alpha} (\langle i| \otimes \mathbb{1}) \left(\sum_k |k\rangle \langle k| \otimes U_k \right) (|j\rangle \otimes \mathbb{1})$$

$$= \sum_{i,j,k} \frac{\sqrt{\alpha_i \alpha_j}}{\alpha} \langle i| (|k\rangle \langle k|) |j\rangle \otimes U_k$$

$$= \frac{1}{\alpha} \sum_k \alpha_k U_k = A/\alpha$$

An explicit example of encoding a sum of unitaries is discussed in Eq. 4.4.

Example 4.3. In this example, we show that the Pauli strings are an orthonormal basis. First we note that $\operatorname{Tr} \sigma^{\alpha \geq 1} = 0$ and $(\sigma^\alpha)^2 = \mathbb{I}$. In the case of single-qubit Pauli operators, we have

$$\langle \sigma^i, \sigma^j \rangle = \delta_{i,j}. \tag{4.16}$$

Since the trace of the tensor product is the product of traces over the separate spaces, we have

$$\langle \sigma^{i_1} \otimes \sigma^{i_2} \otimes \ldots \otimes \sigma^{i_n}, \sigma^{j_1} \otimes \sigma^{j_2} \otimes \ldots \otimes \sigma^{j_n} \rangle = \langle \sigma^{i_1}, \sigma^{j_1} \rangle \langle \sigma^{i_2}, \sigma^{j_2} \rangle \ldots \langle \sigma^{i_n}, \sigma^{j_n} \rangle \tag{4.17}$$

$$= \delta_{i_1,j_1} \delta_{i_2,j_2} \ldots \delta_{i_n,j_n}$$

The above equation is the definition of orthonormal operators. At most, 4^n linearly independent operators act on $\mathcal{C}^{2^n}$, which is exactly the number of Pauli strings with length n. Therefore, Pauli strings form an orthonormal basis according to the Hilbert-Schmidt inner product Eq. 4.11

Example 4.4. Consider two unitaries U, V that block encode A and B, i.e.,

$$U = \begin{bmatrix} A/\alpha & \bullet \\ \bullet & \bullet \end{bmatrix}, , \quad V = \begin{bmatrix} B/\beta & \bullet \\ \bullet & \bullet \end{bmatrix}. \tag{4.18}$$

Given the unitaries U and V we can block encode $A/\alpha + B/\beta$ with the following unitary

$$U_{A+B} = (H \otimes \mathbb{1})(|0\rangle \langle 0| \otimes U + |1\rangle \langle 1| \otimes V)(H \otimes \mathbb{1}) \tag{4.19}$$

In the above case, the *Prep* unitary is a simple Hadamard gate, and the *Select* unitary is a sequence of two controlled unitary operations. Suppose we initialise the auxiliary qubit in the state $|0\rangle$ and that after applying the unitary U_{A+B}, we also observe the state $|0\rangle$. In this case, the top right corner of the applied transformation on the remaining qubits is described by the sum $(A/\alpha + B/\beta)/2$.

References

- Chakraborty, Shantanav, András Gilyén, and Stacey Jeffery. "The power of block-encoded matrix powers: improved regression techniques via faster Hamiltonian simulation." arXiv preprint arXiv:1804.01973 (2018).

- Quantum algorithms: block encoding

- Sünderhauf, C., Campbell, E., & Camps, J. (2024). Block-encoding structured matrices for data input in quantum computing. Quantum, 8, 1226.

4.3 Hybird quantum-classical algorithms

The most important algorithms on the available NISQ devices are variational, quantum-classical algorithms. The main idea is to perform only a small but essential part of the entire calculation on the quantum computer. This is done by formulating our problem as a minimisation problem of a certain cost function C. The cost function should then depend on the output of the quantum circuit, which we can optimise by modifying some parameters θ. After we obtain the output of the quantum device, we proceed on the classical device by calculating the cost function and the updates of the variational parameters θ. Then, we update the parameters of the quantum circuit and repeat the cycle until our objective converges. We show the described idea in Fig. 4.1
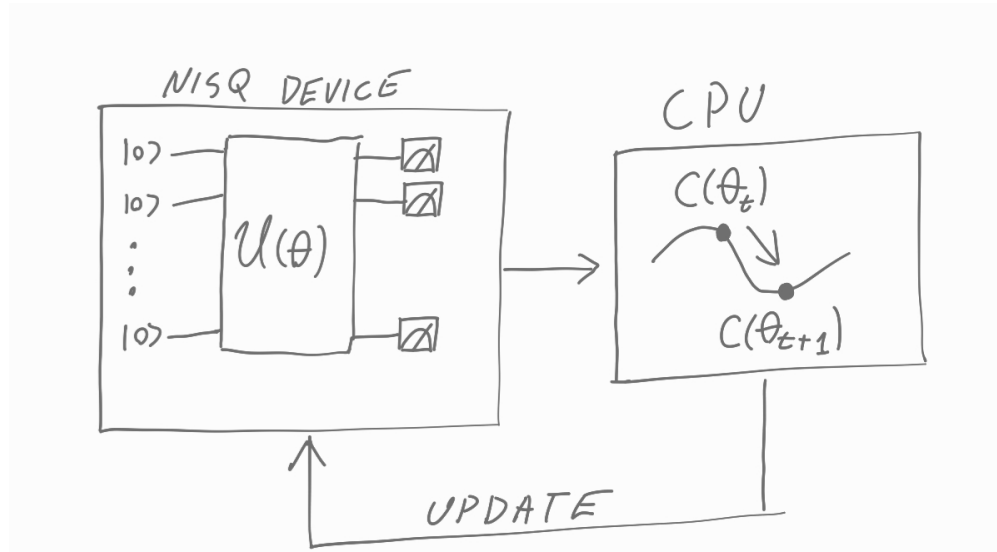
Figure 4.1: The figure shows a schematic representation of the variational, quantum-classical algorithm.

The variational quantum-classical computation paradigm already includes several important algorithms, e.g., quantum approximate optimisation algorithm, variational eigensolver and variational classifiers.

Quantum approximate optimisation algorithm The quantum approximate optimisation algorithm (QAOA) is inspired by adiabatic quantum computation. Formally, the adiabatic path is determined

by a unitary

$$U(t) = \mathrm{T}_{\leftarrow} \mathrm{e}^{\mathrm{i}\int_0^t \mathrm{d}t'\, H(t')}, \qquad (4.20)$$

where the notation $\mathrm{T}_{\leftarrow}$ denotes a time-ordered exponential. The adiabatic Hamiltonian is typically given as

$$H(t) = (1 - \lambda(t))H_0 + \lambda(t)H_1, \qquad (4.21)$$

where $\lambda(t)$ determines the adiabatic schedule and is initially 0 and at the end of the calculation 1. The solution to our problem is then encoded in the ground state of the Hamiltonian H_1.

To simplify the formal solution Eq. 4.20, we use the Baker-Campbell-Hausdorff formula

$$\mathrm{e}^{t(X+Y)=\mathrm{e}^{tX}\mathrm{e}^{tY}\mathrm{e}^{-\frac{t}{2}[X,Y]}} \ldots = \mathrm{e}^{tX}\mathrm{e}^{tY} + \mathcal{O}(t^2). \qquad (4.22)$$

A consequence of the above formula is the Trotter-Suzuki decomposition

$$\mathrm{e}^{X+Y} = \lim_{n\to\infty} \left(\mathrm{e}^{A/n}\mathrm{e}^{B/n} \right)^n. \qquad (4.23)$$

By applying the truncated (finite n) Trotter-Suzuki formula, we can approximate the formal adiabatic path as

$$U(T) \approx U(T, T-\Delta t)U(T-\Delta t, T-2\Delta t)\ldots U(\Delta t, 0) \qquad (4.24)$$

$$\approx \prod_{j=1}^{n} \mathrm{e}^{-\mathrm{i}H(j\Delta t)\Delta t} \qquad (4.25)$$

On a NISQ device, the unitary $\mathrm{e}^{\mathrm{i}H(t)\Delta t}$ is also difficult to implement. Therefore, we split it into two parts by using the Eq. 4.23 as $\mathrm{e}^{\mathrm{i}H(t)} = \mathrm{e}^{\mathrm{i}\lambda(t)H_1\Delta t}\mathrm{e}^{\mathrm{i}(1-\lambda(t))H_0\Delta t}$. Finally, we obtain an approximation of the adiabatic path

$$U(T) \approx \prod_{j=1}^{n} \mathrm{e}^{\mathrm{i}\lambda(j\Delta t)H_1\Delta t}\mathrm{e}^{\mathrm{i}(1-\lambda(j\Delta t))H_0\Delta t} \qquad (4.26)$$

Due to the discussed practical limitation, we can not use a large n. Therefore, we use the following ansatz as an approximation

$$\prod_{j=1}^{n} \mathrm{e}^{\mathrm{i}\lambda(t_j)H_1\Delta t_j}\mathrm{e}^{\mathrm{i}(1-\lambda(t_j))H_0\Delta t_j}, \qquad (4.27)$$

where the times t_j become now variational parameters. The parameters t_j are then modified at each step of the variational optimisation, as discussed in the introduction of the section.

The described quantum approximate optimisation algorithm can be applied on NISQ devices to any optimisation problem for which we can efficiently implement the unitaries in the decomposition Eq. 4.27. The QAOA can also be applied to non-optimisation problems, e.g. Gibbs sampling (see Example. 4.5).

Example 4.5. In quantum computation, we want to avoid environmental interaction. However, there are situations where this interaction can be useful, e.g. in sampling from a Gibbs distribution $p(i) = \mathrm{e}^{-E_i/T}/Z$, where $Z = \sum_i \mathrm{e}^{-E_i/T}$ and E_i are typically Ising energies. As we have seen, the Ising model has a quantum correspondence

$$H_1 = -\sum_{i,j=1}^{N} J_{i,j}\sigma_i^z\sigma_j^z - \sum_{i=1}^{N} h_i\sigma_i^z. \qquad (4.28)$$

The quantum formulation of the Gibbs distribution is then $\rho_T = \mathrm{e}^{-H_1/T}/Z$, where $Z = \mathrm{Tr}\,\mathrm{e}^{-H_1/T}$.

We can prepare the state ρ_T using a coherent algorithm. First, we add N auxiliary qubits and pair them with the main qubits. Then, we rotate each of the pairs in an entangled state

$$|\psi_{\mathrm{pair}}\rangle \frac{1}{\sqrt{Z_{\mathrm{pair}}}} \left(\mathrm{e}^{-\frac{1}{2T}} |+\rangle |+\rangle + \mathrm{e}^{\frac{1}{2T}} |-\rangle |-\rangle \right), \tag{4.29}$$

where the states $|\pm\rangle$ represent the eigenstates of the σ^{x} Hamiltonian. Next, we trace out the auxiliary qubits. The Gibbs state then describes the state of the main qubits

$$\rho_0 = \frac{1}{Z_0} \mathrm{e}^{-H_0/T}, \tag{4.30}$$

where $H_0 = \sum_i \sigma_i^{\mathrm{x}}$. Finally, the desired Gibbs state can be obtained by using the QAOA.

Alternatively, we can implement the interactions H_1 and wait for the system to equilibrate with the environment with temperature $T_{\mathrm{environment}}$. This will then automatically prepare the correct state. The caveat of the second procedure is that due to interference with the measurement devices, the environment temperature does not precisely match the desired T. Therefore, we must find efficient methods to estimate the difference between the desired and environmental temperatures.

Example 4.6. An NP-hard optimisation problem that the QAOA algorithm can solve is the max cut problem. Assume we have a graph defined by a list of edges $(i, j) \in \mathcal{E}$. We aim to separate it into two pieces by cutting as many edges as possible. This is the same as minimising the objective function

$$C(z) = \sum_{(i,j)\in\mathcal{E}} (z_i z_j - 1), \tag{4.31}$$

where $z_j \in -1, 1$. The objective function can be exactly mapped to the Ising model. Therefore, we can immediately apply the QAOA.

Variational eigensolver The variational eigensolver is an algorithm that is mostly applied in quantum chemistry. The idea is to variationally find the lowest energy states of a given Hamiltonian H. This can be formulated as an optimisation problem, where the cost function is simply the energy of the Hamiltonian

$$C(\theta) = \frac{\langle\psi(\theta)| H |\psi(\theta)\rangle}{\langle\psi(\theta)|\psi(\theta)\rangle}. \tag{4.32}$$

The parameter values θ that minimise the objective function Eq. 4.32 determine the approximate eigenstate, and the value of the objective function is the approximate energy. We have to perform exponentially many measurements to calculate the expectation values of the Hamiltonian H. However, we can do that efficiently for a restricted but important class of local Hamiltonians. These Hamiltonians can be expressed as a sum of terms that act only on a few (typically 2) qubits, e.g. $H_{\mathrm{local}} = \sum_{\langle i,j\rangle} H_{i,j}$, where $H_{i,j}$ acts nontrivially only on qubits i and j.

Variational classifiers   A generalisation of the variational eigensolver is the variational classifier. The idea is to interpret the expectation value of a variational circuit as the output of a classifier

$$f(x; \theta) = \langle \psi(x; \theta) | O | \psi(x; \theta) \rangle. \tag{4.33}$$

For M classes, the observable O needs M different eigenvalues determined by the projectors Λ_j. The logits of the variational classifier are then obtained as

$$l_j = \langle \psi(x; \theta) | \Lambda_j | \psi(x; \theta) \rangle, \quad j = 1, 2, \ldots, M. \tag{4.34}$$

Once we have the logits l_j, we can utilise standard cost functions, e.g., the cross-entropy loss, as our optimisation objective.

ML with DQC1 computational model   The hybrid quantum-classical machine learning model is not restricted to any specific type of quantum circuit. Therefore, we can also apply the same idea in the DQC1 computational paradigm, where the resource requirements are less stringent. The main idea is to represent our model as a trace of the unitary $f(x) = \frac{1}{2^n} \operatorname{Tr} U(\boldsymbol{x}, \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ denotes a vector of trainable parameters and $\boldsymbol{x}$ a vector input.

The circuit prepares the state

$$\rho = \frac{1}{2^{n+1}} \left[I_{n+1} + \alpha \left(|0\rangle \langle 1| \otimes U^\dagger + |1\rangle \langle 0| \otimes U \right) \right] \tag{4.35}$$

By measuring the expectation values of $\sigma^{\mathrm{x,y}}$, we find the real and the imaginary part of the trace

$$\langle \sigma^{\mathrm{x}} \rangle = \frac{\alpha}{2^n} \operatorname{Re} \operatorname{Tr} U(\boldsymbol{x}, \boldsymbol{\theta}), \quad \langle \sigma^{\mathrm{y}} \rangle = \frac{\alpha}{2^n} \operatorname{Im} \operatorname{Tr} U(\boldsymbol{x}, \boldsymbol{\theta}). \tag{4.36}$$

Let us consider the unitary operator in the form

$$U(\boldsymbol{x}, \boldsymbol{\theta}) = \prod_{l=1}^{L} W_l(\boldsymbol{\theta}_l) V_l(\boldsymbol{x}_l), \tag{4.37}$$

where $\boldsymbol{\theta}_l$ and $\boldsymbol{x}_l$ represent subsets of the parameter vector $\boldsymbol{\theta}$ and the data vector $\boldsymbol{x}$, respectively, which are loaded at the l-th step. We write the trainable unitary $W_l(\boldsymbol{\theta}_l)$ as

$$W_l(\boldsymbol{\theta}_l) = \prod_{k=1}^{k'} \exp(-i(\boldsymbol{\theta}_l)_k H_{lk}) T_{lk}, \tag{4.38}$$

where $(\boldsymbol{\theta}_l)_k$ represents the k-th element of $\boldsymbol{\theta}_l$, H_{lk} is an $2^n \times 2^n$ Hermitian operator and T_{lk} is an unparametrized unitary.

We further simplify the calculation and consider a special case where the trainable unitary has the simple form

$$W_l(\boldsymbol{\theta}_l) = \exp(-i \sum_k (\boldsymbol{\theta}_l)_k \sigma^{\mathrm{k}}), \tag{4.39}$$

and all σ^k operators in the summation commute with each other. In this case, the partial derivative with respect to the variational parameter reduces to

$$\frac{\partial f(\boldsymbol{x}, \boldsymbol{\theta})}{\partial (\boldsymbol{\theta}_l)_k} = \frac{-i}{2^n} \operatorname{tr} \left(\sigma^k \prod_{j=l}^{L} W_j(\boldsymbol{\theta}_j) V_j(\boldsymbol{x}) \prod_{j=1}^{l-1} W_j(\boldsymbol{\theta}_j) V_j(\boldsymbol{x}) \right). \tag{4.40}$$

This enables us to calculate all derivatives with the same DQC1 computational paradigm and train the model via standard gradient descent training methods.

The DQC1 computational model is not universal, but it still enables exponential speedup with respect to the best-known classical algorithms. Similarly, as we will see in the section about Fourier analysis of hybrid quantum-classical machine learning models, the DQC1 ML paradigm offers comparable expressivity to the standard circuit-based computational model.

References

- Schuld, M., & Petruccione, F. (2021). Machine learning with quantum computers (Vol. 676, pp. 163-169). Berlin: Springer. (Variational Circuits as Machine Learning Models)

- Quantum algorithms: variational quantum algorithms

4.4 Updating parametrised quantum circuits

In the previous section, we discussed calculating the gradients in a restricted DQC1 computational setting. Here, we will show how we can measure gradients more broadly.

The parameter update step is the most crucial in all variational quantum-classical algorithms. Broadly, we can split the parameter update methods into derivative-based and derivative-free methods. The latter can be further divided into iterative methods and one-shot methods. The most well-known iterative algorithms are the Nelder-Mead method and genetic algorithms. The one-shot methods rely on heuristic algorithms that solve the optimisation problem of determining the best parameters with a classical computer.

The gradient-based methods can also be further divided into numeric and analytic variants. Numerically, we can determine the gradient by finite difference methods as

$$\frac{\partial C(\theta)}{\partial \theta_j} = \frac{C(\theta_1, \ldots, \theta_j, \ldots, \theta_N) - C(\theta_1, \ldots, \theta_j + \Delta\theta_j, \ldots, \theta_N)}{\Delta\theta_j} + \mathcal{O}(\Delta\theta_j^2) + \mathcal{O}\left(\frac{\epsilon}{\Delta\theta_j}\right), \tag{4.41}$$

where ϵ denotes the probability of error in the circuit. The finite difference methods have limited precision due to the environmental noise.

Fortunately, we can calculate the derivative of the objective functions analytically with the same complexity as the objective function. The stochastic parameter shift rule is the general rule for calculating the analytic gradients.

Stochastic parameter shift rule We will now introduce the procedure for the analytic calculation of the parametric derivatives of the objective function

$$C(\theta) = \langle \psi_0 | U^\dagger(\theta) O U(\theta) | \psi_0 \rangle. \tag{4.42}$$

We implement the unitary with a variational circuit as

$$U(\theta) = \prod_{t=1}^{T} U_t(\theta) = U_T(\theta) U_{T-1}(\theta) \ldots U_1(\theta) \tag{4.43}$$

where

$$U_t = e^{iX_t(\theta)}. \tag{4.44}$$

The Hamiltonians X_t can be expanded in the Pauli basis $\sigma_\nu = \sigma^{\nu_1} \otimes \sigma^{\nu_2} \otimes \ldots \otimes \sigma^{\nu_N}$,

$$X_t = \sum_\nu x_{t,\nu} \sigma_\nu, \tag{4.45}$$

where $\sigma^{\nu_j} \in \{\sigma^x, \sigma^y, \sigma^z\}$. Given an operator X_t we can calculate the coefficients $x_{t,\nu}$ by using the fundamental Pauli identities

$$\operatorname{Tr}\sigma^{\nu_j} = 0, \quad (\sigma^{\nu_j})^2 = \mathbb{1}_2, \quad \operatorname{Tr}\mathbb{1}_2 = 2. \tag{4.46}$$

We find that

$$x_{t,\nu}(\theta) = \operatorname{Tr} X_t(\theta)\sigma_\nu / 2^N. \tag{4.47}$$

The immediate calculation of the cost function's parameter derivatives is impractical. Therefore, we use the chain rule to calculate and write

$$\frac{\partial C(\theta)}{\partial \theta_j} = \sum_t \sum_\nu \frac{\partial C(\theta)}{\partial x_{t,\nu}(\theta)} \frac{\partial x_{t,\nu}(\theta)}{\partial \theta_j}. \tag{4.48}$$

The strategy now is to calculate the derivatives of the cost function with respect to the expansion coefficients $x_{t,\nu}(\theta)$. The parameter derivatives can then be obtained via the chain rule Eq. 4.48.

From now on we will fix t and $\nu = (\nu_1, \nu_2, \dots, \nu_N)$ and write

$$x = x_{t,\nu}, \qquad V = \sigma_\nu, \qquad H = \sum_{\mu \neq \nu} x_{t,\mu}\sigma_\mu, \tag{4.49}$$

$$|\phi\rangle = \prod_{s=1}^{t-1} U_s |\psi_0\rangle, \qquad A = U_{t^+}^\dagger O U_{t^+}, \qquad U_{t^+} = \prod_{s=t+1}^{T} U_s = U_T(\theta)U_{T-1}(\theta)\dots U_{t+1}.$$

The objective can now be written as a function of x

$$C(x) = \langle\phi| U_t^\dagger(\theta) A U_t |\phi\rangle = \langle\phi| e^{-i(H+xV)} A e^{i(H+xV)} |\phi\rangle \tag{4.50}$$

By using the cyclic property of the trace, we find that

$$C(x) = \operatorname{Tr} A e^{i(H+xV)} \rho e^{-i(H+xV)}, \quad \rho = |\phi\rangle\langle\phi|. \tag{4.51}$$

We now introduce a commutator map that acts on the state ρ as follows

$$Z[\rho] = i\left((H+xV)\rho - \rho(H+xV)\right) \tag{4.52}$$

Since Z is a linear map, it can be considered a matrix acting on a vectorised ρ. We will also need the Baker-Campbell-Hausdorff formula

$$e^{[X,\bullet]}Y = e^X Y e^{-X} \tag{4.53}$$

where $[X, \bullet]$ represents a commutator with X. We now combine the equations Eq. 4.51, Eq. 4.52, and Eq. 4.53 and compactly write the objective as

$$C(x) = \operatorname{Tr} A e^Z [\rho]. \tag{4.54}$$

The above formula Eq. 4.54 is just a compact reformulation of the original expression Eq. 4.51, which will enable us to derive the stochastic parameter shift rule.

Next, we introduce the commutator map

$$\frac{\partial Z}{\partial x} = \mathcal{V} = i[V, \bullet]. \tag{4.55}$$

By using the property $V^2 = \mathbb{1}$ we find

$$e^{\lambda \mathcal{V}} = \rho + \sin^2(\lambda)(V\rho V - \rho) + \frac{i}{2}\sin(2\lambda)[V,\rho]. \tag{4.56}$$

We get the expression in Eq. 4.56 by using the Taylor expansion of the matrix exponential. By taking the derivative of the Eq. 4.56 on both sides, we get

$$\mathcal{V}e^{\lambda\mathcal{V}} = e^{(\lambda+\frac{\pi}{4})\mathcal{V}}[\rho] - e^{(\lambda-\frac{\pi}{4})\mathcal{V}}[\rho] \tag{4.57}$$

Using the expression in Eq. 4.57 at $\lambda = 0$ we get

$$\frac{\partial Z}{\partial x}[\rho] = e^{\frac{\pi}{4}\mathcal{V}}[\rho] - e^{-\frac{\pi}{4}\mathcal{V}}[\rho] = e^{i\frac{\pi}{4}V}\rho e^{-i\frac{\pi}{4}V} - e^{-i\frac{\pi}{4}V}\rho e^{i\frac{\pi}{4}V}. \tag{4.58}$$

To get the final expression, we have to introduce one more identity, namely

$$\frac{\partial e^Z}{\partial x} = \int_0^1 ds e^{sZ}\frac{\partial Z}{\partial x}e^{(1-s)Z}. \tag{4.59}$$

By combining the expressions Eq. 4.58 and Eq. 4.59, we can now calculate the derivative of the cost function with respect to x

$$\frac{\partial C(x)}{\partial x} = \mathrm{Tr}\, A\partial_x e^Z[\rho] \tag{4.60}$$

$$= \int_0^1 ds \left[\mathrm{Tr}\left(Ae^{-is(H+xV)}e^{i\frac{\pi}{4}V}e^{i(1-s)(H+xV)}\rho e^{-is(H+xV)}e^{-i\frac{\pi}{4}V}e^{-i(1-s)(H+xV)} \right) - \right.$$

$$\left. - \mathrm{Tr}\left(Ae^{-is(H+xV)}e^{-i\frac{\pi}{4}V}e^{i(1-s)(H+xV)}\rho e^{-is(H+xV)}e^{i\frac{\pi}{4}V}e^{-i(1-s)(H+xV)} \right) \right]$$

We can compactly write the above expression by introducing the shifted objectives

$$C_\pm(x,s) = \langle\phi| U_\pm^\dagger(x,s)AU_\pm(x,s)|\phi\rangle, \quad \text{where} \tag{4.61}$$

$$U_\pm(x,s) = e^{-is(H+xV)}e^{\pm i\frac{\pi}{4}V}e^{i(1-s)(H+xV)}. \tag{4.62}$$

We then have

$$\partial_x C(x) = \int_0^1 (C_+(x,s) - C_-(x,s))ds. \tag{4.63}$$

To calculate the derivative of the objective function, we have to implement two more local gates for each term in the expansion Eq. 4.45. Then we need to perform the integral over s, which can be done using Monte Carlo and does not significantly increase the number of measurements since we need to calculate quantum expectations using many measurements.

Below, we summarise the stochastic parameter shift rule algorithm based on the equation Eq. 4.63.

1. Sample s from the uniform distribution in [0,1];
2. Initialize the computer in the state $|\phi\rangle$
3. Apply the gate $e^{i(1-s)(H+xV)}$
4. Apply the gate $e^{i\frac{\pi}{4}V}$
5. Apply the gate $e^{is(H+xV)}$
6. Measure the observable A and call the result r_+
7. Repeat steps 2 to 5, but on point 4, apply $e^{-i\frac{\pi}{4}V}$
8. Measure A and call the result r_- the sample $g_{t,\nu} = r_+ - r_-$ is such that $\partial C/\partial x_{t,\nu} = \mathbb{E}[g_{t,\nu}]$.

References

- Banchi, L., & Crooks, G. E. (2021). Measuring analytic gradients of general quantum evolution with the stochastic parameter shift rule. Quantum, 5, 386.

4.5 Fourier analysis and exponential encoding

In this section, we explore the impact of data encoding strategies on the expressive power of parameterised quantum circuits when used as function approximators. A central theme is how the choice of encoding determines the set of functions a quantum model can represent. We introduce a framework that expresses quantum models as partial Fourier series, with the frequencies directly linked to the structure of the data encoding gates. By iterating even simple encoding gates, one can enrich the frequency spectrum accessible to the model, thereby enhancing its representational capacity. We further introduce a specific exponential encoding which enables a possible quantum speedup, provided the variational part of the quantum circuit is flexible enough.

A key tool in our analysis is the representation of quantum models as Fourier-type series. We express the output of a parameterised quantum model as

$$f_\theta(x) = \langle 0| U^\dagger(x,\theta)OU(x,\theta) |0\rangle \qquad (4.64)$$
$$= \sum_{\omega \in \Omega} c_\omega(\theta)e^{i\omega \cdot x},$$

where $\omega \cdot x$ denotes the inner product, $\Omega \subset \mathbb{R}^N$ is a set of accessible frequencies, and the coefficients $c_\omega(\theta)$ depend on the circuit parameters θ.

Importantly, we will show that the frequency spectrum Ω is entirely determined by the eigenvalues of the data-encoding Hamiltonians used in the circuit. Meanwhile, the structure of the full quantum circuit governs which coefficients c_ω can be realised.

This Fourier-type representation reveals two intertwined aspects of a quantum model's expressivity: the set of accessible basis functions $\{e^{i\omega \cdot x}\}$, and the ability to control the corresponding coefficients $\{c_\omega\}$. In many practical settings, the frequency set Ω consists of integer vectors, i.e., $\Omega \subset \mathbb{Z}^N$, and the model becomes a partial multidimensional Fourier series:

$$f_\theta(x) = \sum_{n \in \Omega} c_n(\theta)e^{in \cdot x}. \qquad (4.65)$$

Here, the functions $e^{in \cdot x}$ form an orthogonal basis, and the term "partial" emphasises that only a subset of the Fourier coefficients are non-zero. This Fourier framework enables a systematic and powerful approach to understanding the functional capacity of quantum models using well-established tools from Fourier analysis.

First, we introduce our basic tool: the natural representation of a quantum model as a partial Fourier series. For simplicity, the majority of our presentation will focus on the case of univariate functions with inputs $x \in \mathbb{R}$.

We define a (univariate) quantum model $f(x)$ as the expectation value of an observable with respect to a quantum state prepared by a parameterised circuit:

$$f(x) = \langle 0|U^\dagger(x,\theta)OU(x,\theta) |0\rangle , \qquad (4.66)$$

where $|0\rangle$ is the initial state, O is an observable, and $U(x,\theta)$ is a quantum circuit composed of data-encoding and trainable blocks. The model prediction is estimated by repeated execution and averaging over measurement outcomes.

The circuit $U(x,\theta)$ is constructed from L layers, each consisting of a fixed data-encoding unitary $S(x)$ and an arbitrary trainable unitary $W(\theta)$. We use the Hamiltonian encoding via gates of the form $G(x) = e^{-ixH}$, where H is a Hermitian operator (Hamiltonian). To isolate the effect of data encoding, we treat all trainable blocks $W(\theta)$ as general unitaries and omit the dependence on θ. The circuit thus takes the form:

$$U(x) = W^{(L+1)}S(x)W^{(L)} \cdots S(x)W^{(1)}. \qquad (4.67)$$

59

A useful simplification is obtained by diagonalising the Hamiltonian $H = V^\dagger \Lambda V$, where Λ is diagonal with eigenvalues $\lambda_1, \ldots, \lambda_d$. The encoding unitaries become $S(x) = V^\dagger e^{-ix\Lambda} V$, and we can absorb the V and $V^\dagger$ into the surrounding unitaries to assume, without loss of generality, that H is diagonal.

This decomposition allows us to isolate the data-dependent phases. The i-th component of the state $U(x)|0\rangle$ can be written as:

$$[U(x)|0\rangle]_i = \sum_{j_1, \ldots, j_L = 1}^{d} e^{-i(\lambda_{j_1} + \cdots + \lambda_{j_L})x} W_{ij_L}^{(L+1)} \cdots W_{j_2 j_1}^{(2)} W_{j_1 1}^{(1)}. \tag{4.68}$$

Introducing the multi-index $\mathbf{j} = (j_1, \ldots, j_L) \in [d]^L$, we define $\mu_{\mathbf{j}} = \lambda_{j_1} + \cdots + \lambda_{j_L}$, so that the state becomes:

$$[U(x)|0\rangle]_i = \sum_{\mathbf{j} \in [d]^L} e^{-i\mu_{\mathbf{j}} x} W_{ij_L}^{(L+1)} \cdots W_{j_2 j_1}^{(2)} W_{j_1 1}^{(1)}. \tag{4.69}$$

To compute the model output, we combine this with the measurement observable and take the real part:

$$f(x) = \sum_{\mathbf{k}, \mathbf{j} \in [d]^L} e^{i(\mu_{\mathbf{k}} - \mu_{\mathbf{j}})x} a_{\mathbf{k}, \mathbf{j}}, \tag{4.70}$$

where the coefficients $a_{\mathbf{k}, \mathbf{j}}$ depend on the unitaries and measurement:

$$a_{\mathbf{k}, \mathbf{j}} = \sum_{i, i'} W_{1k_1}^{(1)*} W_{k_1 k_2}^{(2)*} \cdots W_{k_L i}^{(L+1)*} M_{ii'} W_{i'j_L}^{(L+1)} \cdots W_{j_2 j_1}^{(2)} W_{j_1 1}^{(1)}. \tag{4.71}$$

By grouping terms with the same frequency $\omega = \mu_{\mathbf{k}} - \mu_{\mathbf{j}}$, we obtain the final Fourier representation:

$$f(x) = \sum_{\omega \in \Omega} c_\omega e^{i\omega x}, \tag{4.72}$$

where the frequency spectrum is defined by

$$\Omega = \{\mu_{\mathbf{k}} - \mu_{\mathbf{j}} \mid \mathbf{k}, \mathbf{j} \in [d]^L\}, \tag{4.73}$$

and the Fourier coefficients are given by

$$c_\omega = \sum_{\substack{\mathbf{k}, \mathbf{j} \in [d]^L \\ \mu_{\mathbf{k}} - \mu_{\mathbf{j}} = \omega}} a_{\mathbf{k}, \mathbf{j}}. \tag{4.74}$$

We note several key properties of this spectrum: $0 \in \Omega$, and if $\omega \in \Omega$, then $-\omega \in \Omega$ as well. Since $c_\omega = c_{-\omega}^*$, the function $f(x)$ is real-valued. The **degree** of the spectrum is defined as $D = \max(\Omega)$, and we denote by $K = (|\Omega| - 1)/2$ the number of independent nonzero frequencies. If the eigenvalues λ_i are integers, the frequency spectrum consists of integer values, and the model becomes a real-valued partial Fourier series.

These results show that a quantum model's expressivity is determined by two independent factors: the structure of the frequency spectrum Ω, governed by the eigenvalues of the encoding Hamiltonians, and the controllability of the coefficients c_ω, which depends on the trainable unitaries and measurement observable. In the following example, we explore how these two aspects affect the ability of quantum models to approximate functions.

Example 4.7. Single qubit encoding

As a warm-up application of the Fourier series formalism, we begin with a simple quantum model consisting of a single encoding layer ($L = 1$). Specifically, we consider a model where the input x is encoded via a single-qubit gate of the form $G(x) = e^{-ixH}$, and the full circuit is given by

$$U(x) = W^{(2)}G(x)W^{(1)}. \tag{4.75}$$

Here, H is a Hermitian generator with two distinct eigenvalues λ_1, λ_2. Without loss of generality, we may rescale the spectrum of H to $(-\gamma, \gamma)$, since global phases are unobservable in quantum mechanics. A common and important example includes Pauli rotations, where $H = \frac{1}{2}\sigma$ for $\sigma \in \sigma_x, \sigma_y, \sigma_z$, yielding $\gamma = \frac{1}{2}$. We aim to show that any model of the form in Eq. (4.75) produces a function $f(x)$ of the form:

$$f(x) = A\sin(2\gamma x + B) + C, \tag{4.76}$$

where the constants A, B, and C depend on the trainable components of the circuit but are independent of the encoded input.

To demonstrate this, we first absorb the factor γ into the input via a rescaled variable $\tilde{x} = \gamma x$. We may now assume without loss of generality that the eigenvalues of H are $\lambda_1 = -1$ and $\lambda_2 = 1$. From our general Fourier formalism, the set of frequency differences $\omega = \lambda_{k_1} - \lambda_{j_1}$ for $\lambda_{k_1}, \lambda_{j_1} \in -1, 1$ yields:

$$\Omega = -2, 0, 2. \tag{4.77}$$

Thus, the model output has the structure:

$$f(x) = c_{-2}e^{i2\tilde{x}} + c_0 + c_2 e^{-i2\tilde{x}}. \tag{4.78}$$

This expression corresponds to a real-valued cosine function:

$$f(x) = c_0 + 2|c_2|\cos(2\tilde{x} - \arg(c_2)), \tag{4.79}$$

where $\arg(c_2)$ denotes the complex phase of c_2. For Pauli rotations, $\tilde{x} = x/2$, and we recover the form of Eq. (4.76) with:

$$A = 2|c_2|, \quad B = -\pi/2 - \arg(c_2), \quad C = c_0. \tag{4.80}$$

Importantly, this result holds regardless of the number of qubits in the system, the structure of the unitaries $W^{(1)}$, $W^{(2)}$, or the form of the measurement observable O. This illustrates a key insight of our analysis: even arbitrarily deep and wide quantum circuits cannot escape the fundamental expressivity limitations imposed by the encoding strategy.

Example 4.8. Repeated Pauli encodings linearly extend the frequency spectrum

Given the severe limitations discussed in the previous section, a natural question arises: *how can we systematically extend the frequency spectrum accessible to a quantum model?* In this section, we show that the Fourier degree can be increased linearly by repeating the data-encoding gates—either in parallel (within a single layer) or sequentially (across multiple layers).

We analyse two cases:

1. **Parallel repetition**: A single-layer model ($L = 1$) where the encoding gate is repeated r times in parallel across different qubits.

2. **Sequential repetition**: A multi-layer model ($L = r$), where a single-qubit encoding gate is applied repeatedly across r layers.

Both techniques are common in practical quantum machine learning applications, and this analysis provides a theoretical foundation for their effectiveness.

1. Parallel repetition of Pauli encodings

We first consider a model with $L = 1$ and an encoding gate applied in parallel across r qubits:

$$S(x) = e^{-i\frac{x}{2}\sigma_r} \otimes \cdots \otimes e^{-i\frac{x}{2}\sigma_1}, \tag{4.81}$$

where each $\sigma_j \in \{\sigma_x, \sigma_y, \sigma_z\}$. Since the rotations act on different qubits, they commute, and the full encoding unitary can be written as:

$$S(x) = V^\dagger \exp\left(-i\frac{x}{2}\sum_{q=1}^{r}\sigma_z^{(q)}\right) V = V^\dagger e^{-ix\Lambda}V, \tag{4.82}$$

where $\sigma_z^{(q)}$ denotes a Pauli-Z operator acting on qubit q, and Λ is diagonal with eigenvalues:

$$\lambda_p = \frac{p - r}{2}, \quad p \in \{0, 1, \ldots, r\}. \tag{4.83}$$

These values arise from all possible sums of r terms $\pm\frac{1}{2}$. The spectrum of the model consists of differences between any two of these eigenvalues:

$$\Omega_{\text{par}} = \{\lambda_{k_1} - \lambda_{j_1} \mid k_1, j_1 \in \{1, \ldots, 2^r\}\} \tag{4.84}$$
$$= \{p - p' \mid p, p' \in \{0, \ldots, r\}\} \tag{4.85}$$
$$= \{-r, -r+1, \ldots, 0, \ldots, r-1, r\}. \tag{4.86}$$

Hence, a quantum model with r parallel Pauli encodings yields a **truncated Fourier series of degree r**.

2. Sequential repetition of Pauli encodings

Now consider a circuit where a single-qubit Pauli encoding gate is repeated across r layers:

$$U(x) = W^{(r+1)}e^{-i\frac{x}{2}\sigma_r}W^{(r)}\cdots W^{(2)}e^{-i\frac{x}{2}\sigma_1}W^{(1)}. \tag{4.87}$$

Assume all σ_j are equal and act on the same qubit. As before, we diagonalize each encoding gate to get $\Lambda = \frac{1}{2}\sigma_z$, and the full spectrum consists of differences of sums of r eigenvalues:

$$\Omega_{\text{seq}} = \left\{\left(\sum_{l=1}^{r}\lambda_{k_l}\right) - \left(\sum_{l=1}^{r}\lambda_{j_l}\right) \;\middle|\; k_l, j_l \in \{1, 2\}\right\}. \tag{4.88}$$

This again results in:

$$\Omega_{\text{seq}} = \{-r, -r+1, \ldots, r-1, r\}. \tag{4.89}$$

Thus, the **frequency spectrum grows linearly** with the number of repetitions r, whether applied in parallel or sequentially.

Summary

Both parallel and sequential repetitions of Pauli-rotation encodings enable a quantum model to express a larger class of functions via an extended Fourier spectrum. Specifically, the model output can be described as a **truncated Fourier series of degree** r, and this scaling mechanism is independent of other details like the choice of trainable unitaries or measurement observable.

Example 4.9. Exponential encoding and frequency spectrum

We can further improve the set of available frequencies by an exponential encoding strategy where input data $x \in [-\pi, \pi)$ is embedded into a quantum circuit using a sequence of N single-qubit Z-rotations with integer weights β_j:

$$S(x) = \bigotimes_{j=1}^{N} e^{-\mathrm{i}\frac{x\,\beta_j}{2}\sigma^z}. \tag{4.90}$$

Each β_j defines the frequency contribution of the j-th encoding gate. The total unitary $S(x)$ acts diagonally in the computational basis, such that the encoded state accumulates a phase depending on the sum of the weighted bit values of the computational basis states.

The frequency spectrum $\Omega^{(N)}$ of the encoded model consists of all integer differences between accumulated phases of computational basis states:

$$\Omega^{(N)} = \left\{ \sum_{j=1}^{N} (k'_j - k_j)\beta_j \ \middle|\ k_j, k'_j \in \{0,1\} \right\}. \tag{4.91}$$

The set $\Omega^{(N)}$ can be built recursively. Given $\Omega^{(N-1)}$, we have:

$$\Omega^{(N)} = \Omega^{(N-1)} \cup (\Omega^{(N-1)} + \beta_N) \cup (\Omega^{(N-1)} - \beta_N). \tag{4.92}$$

To ensure a **dense**, **non-degenerate** spectrum, we choose the β_j recursively such that each new frequency added is larger than twice the maximum existing frequency:

$$\beta_j = 2\sum_{i=1}^{j-1} \beta_i + 1. \tag{4.93}$$

Starting with $\beta_1 = 1$, this recurrence yields:

$$\beta_j = 3^{j-1}, \quad \text{for } j = 1, \ldots, N. \tag{4.94}$$

Thus, the resulting circuit is:

$$S(x) = \bigotimes_{j=1}^{N} e^{-\mathrm{i}\,x\,\frac{3^{j-1}}{2}\sigma^z}, \tag{4.95}$$

and generates a frequency spectrum:

$$\Omega = \left\{ \sum_{j=1}^{N} b_j 3^{j-1} \ \middle|\ b_j \in \{-1, 0, 1\} \right\}. \tag{4.96}$$

This forms a symmetric, dense, integer-valued spectrum of size $|\Omega| = 3^N$.

We can generalise the exponential encoding by using $\beta_j = l^{j-1}$ for some $l > 1$. Larger l creates a sparser spectrum with potentially higher maximum frequencies. This trade-off allows one to:

- Increase expressivity by expanding the range of representable frequencies,

- Or control overfitting by limiting the number of distinct frequency components.

Moreover, to target a specific maximum frequency β^*, one can select weights β_j such that:

$$\sum_{j=1}^{N} \beta_j = \beta^*, \tag{4.97}$$

which may result in degeneracies (repeated frequencies) and can be leveraged to concentrate representational power around specific frequencies.

Exponential encoding provides a flexible and powerful mechanism to modulate the Fourier spectrum of quantum models. By carefully selecting the β_j weights, one can tailor the expressivity and generalisation behaviour of the quantum model to match the desired function class.

References

- Schuld, M., Sweke, R., & Meyer, J. J. (2021). Effect of data encoding on the expressive power of variational quantum-machine-learning models. Physical Review A, 103(3), 032430.

- Shin, S., Teo, Y. S., & Jeong, H. (2023). Exponential data encoding for quantum supervised learning. Physical Review A, 107(1), 012422.

4.6   Quantised classical models

The main goal of quantum machine learning discussed so far is to obtain some computational speedup with respect to the best possible classical algorithms. This can be done by transferring the critical parts of the classical algorithms to the quantum devices. Though predominant, quantum speedup is not the only goal of quantum machine learning research. This section will examine genuine quantum models and what insight they can offer for machine learning. Many questions regarding the usefulness of genuine quantum distributions/models are still open. The training properties of quantum models and their generalisation and robustness are also not fully understood. Concrete, quantised classical models can perhaps provide some answers that might be applicable more generally.

This section will discuss two quantised approaches, the quantum Boltzmann machines and the qboost algorithm.

Quantum Boltzmann machines   We start our discussion with a short review of the classical recurrent networks, specifically Hopfield models and then Boltzmann machines as their generalisation. Hopfield models and Boltzmann machines are recurrent neural networks with G binary variables $s = s_1 s_2 \ldots s_G$, where

$s_j \in \{-1, 1\}$. We can separate the variables s into the hidden $h_j = s_j$ (for $j = 1, 2 \dots, N_h$) and visible variables $v_j = s_{j+N_h}$ for $j = 1, 2 \dots N_v$, where $N = N_h + N_v$. An important object in both the Hopfield networks and the Boltzmann machines is the energy function

$$E(s) = -\sum_{i,j=1}^{N} w_{i,j} s_i s_j - \sum_{i=1}^{N} b_i s_i, \tag{4.98}$$

with real parameters $w_{i,j}$ and b_i. In Hopfield models, the energy function is symmetric $w_{i,j} = w_{j,i}$ and without self connections $w_{i,i} = 0$ and biases $b_i = 0$. The state of each variable in the Hopfield model is then updated according to the perceptron rule

$$s_i^{t+1} = \text{sign}\left(\sum_{j=1}^{N} w_{i,j} s_j\right). \tag{4.99}$$

The perceptron update does not increase the Ising energy Eq. 4.98. Therefore, we are guaranteed to arrive at a local minimum. The perceptron dynamics can be interpreted as an associative recall of patterns saved in the local minima. The problem is that we can store only around 15 different states with 100 bits, which is far from all possible 2^{100} different binary strings.

The recall algorithm is similar to the zero-temperature Monte Carlo algorithm. The Monte Carlo algorithm obtains a new configuration of variables s' by flipping one or more variables of the old configuration s. At zero temperature, the change is accepted if $E(s') < E(s)$.

In this framework, a generalisation of the Hopfield model is straightforward. We associate to each configuration a probability $p(s) = \mathrm{e}^{-E(s)/T}/Z$, where $Z = \sum_s \mathrm{e}^{-E(s)/T}$. The introduced distribution is known as the Boltzmann distribution. If the zero temperature Monte Carlo is related to the Hopfield model, the finite temperature Monte Carlo results in a classical Boltzmann machine. The Boltzmann machines are a variational generative model. The parameters of the Ising energy should be chosen such that the Boltzmann distribution is as close as possible to a real dataset distribution. If this is achieved, we can generate new samples by sampling from the model using the finite temperature Monte Carlo algorithm.

Here, we are interested in the quantum Boltzmann machine. To arrive at the quantum generalisation of the Boltzmann machine, we first have to determine the quantum Hamiltonian associated with the classical Ising energy Eq. 4.98. We arrive at the quantum formulation of the energy Eq. 4.98 by applying the following identities

$$\langle 0 | \sigma^z | 0 \rangle = 1, \quad \langle 1 | \sigma^z | 1 \rangle = -1, \quad \langle i, j | A \otimes B | i, j \rangle = \langle i | A | i \rangle \langle j | B | j \rangle. \tag{4.100}$$

Hence, we can write

$$E(s) = \langle (1-s)/2 | \left(-\sum_{i,j} w_{i,j} \sigma_i^z \sigma_i^z - \sum_i b_i \sigma_i^z \right) | (1-s)/2 \rangle = \langle (1-s)/2 | H_{\text{Ising}} | (1-s)/2 \rangle. \tag{4.101}$$

The Hamiltonian is the Ising Hamiltonian, a quantum formulation of the classical Ising energy. The Hamiltonian is diagonal and contains all possible values of the classical Ising energy. We can now introduce quantum effects by adding non-commuting terms to the Hamiltonian Eq. 4.101. A typical quantum extension is the transverse field Ising model (TFIM)

$$H_{\text{TFIM}} = H_{\text{Ising}} + \sum_i c_i \sigma_i^x. \tag{4.102}$$

Now we can introduce the quantum Boltzmann machines as the Gibbs distribution with respect to the TFIM, namely

$$\rho_{\text{TFIM}} = \mathrm{e}^{H_{\text{TFIM}}/T}/Z, \quad Z = \text{Tr}\, \mathrm{e}^{H_{\text{TFIM}}/T}. \tag{4.103}$$

For a projectors Λ_s the quantum Boltzmann machine introduces a classical probability distribution by $p(s) = \text{Tr}\,\Lambda_s \rho_{\text{TFIM}}$, where $\Lambda_s = |s\rangle\langle s|$. It is an open problem to describe the properties of this kind of "quantum" distribution with regard to generative machine learning tasks. An open problem is finding an efficient training algorithm for quantum Boltzmann machines. We currently have an algorithm that minimises an upper bound of the desired Log-likelihood objective. However, the upper bound can be very far from the actual values; hence, the algorithm often performs poorly.

The described approach to the "quantisation" of the Boltzmann machines encapsulates the main general steps:

1. Write the classical model in the framework of quantum probability

2. Add "quantum" terms, and define the appropriate quantum probability distribution, which reduces to the classical in the case of diagonal operators.

3. Study the properties of the quantised model. How is it different from the classical? Can we efficiently exploit this difference? Applications on NISQ devices...

Besides the quantum Boltzmann machines, there are many other probabilistic quantum models: quantum Hopfield models, quantum hidden Markov models, quantum graphical models (quantum belief propagation), and quantum causal models.

Qboost The final quantum model we will discuss is the Qboost algorithm, one of the first quantum machine learning algorithms. It is a "quantisation" of the classical ensembling algorithms, such as Adaboost or XGboost. The ensembling problem is formulated as follows. Given a labeled dataset $\{x^j, y^j\}$ and K models $h_m(x)$ we want to find a linear combination

$$F_K(w, x) = \sum_{k=1}^{K} w_k h_k(x) \tag{4.104}$$

that performs better than the individual models. What is the best ensembling method strategy is still an open problem.

A simple classical approach to the problem is the AdaBoost algorithm. In this algorithm, the ensemble is increased sequentially, and the weights are derived from the exponential loss $L_c = \sum_i e^{y_i F_m(x_i)}$. The problem with this algorithm is that it does not use any regularisation. Hence, a model is included irrespective of its utility or even if it is repeated many times.

Our strategy to obtain a quantum ensembling algorithm will be to define a quadratic loss, which we will then write as an Ising model. We can directly apply a quantum annealing algorithm or the QAOA algorithm from there. The loss that we will minimise is the mean-square error of the ensemble prediction

$$\min_{W} \left[\frac{1}{N} \sum_{i=1}^{N} (\sum_{k=1}^{K} w_k h_k(x_i) - y_i)^2 + \lambda ||w||_0 \right], \tag{4.105}$$

where $||w||_0 = \sum_{k=1}^{K} w_k^0$. We can rewrite the above expression as

$$\min_w \left[\frac{1}{N} \sum_{i=1}^{N} \left[(\sum_{k=1}^{K} w_k h_k(x_i))^2 - 2 \sum_{k=1}^{K} (w_k h_k(x_i) y_i) + y_i^2 \right] + \lambda ||w||_0 \right] \tag{4.106}$$

$$\downarrow$$

$$\min_w \left[\frac{1}{N} \sum_{k,l=1}^{K} w_l w_k (\sum_{i=1}^{N} h_k(x_i) h_l(x_i)) - 2 \sum_{k=1}^{K} w_k (\sum_{i=1}^{N} h_k(x_i) y_i) + \lambda ||w||_0 \right]$$

The last expression is similar to the classical Ising model. The first term in the brackets is the interaction energy; the last two translate to the bias term. The main difference is that the variables w_i are continuous.

Therefore, we only have to use a finite bitwidth/precision. For example we represent each w_k with only three bits $w_k = 0.w_k^1 w_k^2 w_k^3$, where $w_k^j \in \{0, 1\}$. With this modification, the minimisation objective is exactly the classical Ising model. We can now use a quantum annealer or a QAOA to optimise our objective, including a regularisation term. Therefore, we can tune the balance between the ensemble's accuracy and complexity/size. An early Kaggle evaluation of the boosting model on realistic datasets showed that the classical ensembling methods outperform the quantum version.

References

- Schuld, M., & Petruccione, F. (2021). Machine learning with quantum computers (Vol. 676, pp. 163-169). Berlin: Springer. (Chapter: Fault-Tolerant Quantum Machine Learning)

- Quantum algorithms: quantum machine learning via linear algebra

4.7   Quantum kernels

In this section, we will show that many of the quantum models we have discussed can be represented in the framework of kernel methods. In particular, we will show that

1. quantum models are linear models in the feature vectors $\rho(x)$

2. Quantum models that minimise typical machine learning cost functions have measurements that can be written as "kernel expansions in the data" $A = \sum_m \alpha_m \rho(x_m)$.

3. The problem of finding the optimal measurement for typical machine learning cost functions trained with M data samples can be formulated as an M-dimensional convex optimisation problem.

Before we start our discussion of the quantum kernels, we will briefly review the classical kernel methods.

Kernel methods   Kernels can be considered similarity functions between data points of the input space $\mathcal{X}$. More formally, given a non-empty input space $\mathcal{X}$, with $x_1 x_2, \dots x_N \in \mathcal{X}$, and complex numbers $c_1, c_2, \dots, c_N \in \mathbb{C}$ a function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{C}$ is a kernel if

$$\sum_{i,j=1}^{N} c_i^* c_j \kappa(x_i, x_j) \geq 0. \tag{4.107}$$

Defining a Gram matrix $K_{i,j} = \kappa(x_i, x_j)$ is also useful. Then the above condition translates to positive-semidefiniteness of the Gram matrix $K \geq 0$. The positivity condition Eq. 4.107 implies that $\kappa(x, x) \geq 0$ and $\kappa(x, y) = \kappa(y, x)^*$, where the star denotes the complex conjugation.

The kernel can be used to define a canonical feature map ϕ from the input set to the complex-valued functions on the input set

$$\Phi : \mathcal{X} \to \mathbb{C}^{\mathcal{X}}, \quad x \xrightarrow{\Phi} \kappa(\bullet, x). \tag{4.108}$$

The image of the map is a function which maps elements of the input space to complex numbers. The space

of these functions is a vector space. Two different elements of that vector space can be expressed as

$$f(\bullet) = \sum_{i=1}^{I} \nu_i \kappa(\bullet, x_i), \quad \nu_i \in \mathbb{R}, \tag{4.109}$$

$$g(\bullet) = \sum_{j=1}^{J} \mu_j \kappa(\bullet, x_j), \quad \mu_j \in \mathbb{R}.$$

Given arbitrary elements g, f of the vector space defined by the canonical feature map, we can define an inner product on that space as

$$\langle f, g \rangle = \sum_{i=1}^{I} \sum_{j=1}^{J} \nu_i^* \mu_j \kappa(x_i, x_j). \tag{4.110}$$

That this is a valid product follows from the positivity condition of the kernel Eq. 4.107. An inner product between two canonical feature maps of different elements of the input space is then

$$\langle \Phi(x), \Phi(y) \rangle = \langle \kappa(\bullet, x), \kappa(\bullet, y) \rangle = \kappa(x, y). \tag{4.111}$$

The above expression in Eq. 4.111 solidifies the statement that the kernels are similarity or distance measures between points of the input space $\mathcal{X}$. Moreover, the vector space obtained by the canonical feature map (also called the feature space and denoted by $\mathcal{F}$) can be extended to a Hilbert space by including all limit points of Cauchy series, where the norm is defined by the inner product defined in Eq. 4.111.

We can also make a reverse statement. Given a feature map from the input space to some Hilbert space $\phi : \mathcal{X} \to \mathcal{H}$, we can define a kernel on the input space as the inner product between feature maps of the input elements

$$\kappa(x, y) = \langle \phi(x), \phi(y) \rangle. \tag{4.112}$$

This is a valid kernel, since the inner product is linear and positive-semidefinite. Using the expression Eq. 4.111 on the kernel Eq. 4.112 we see that the Hilbert space $\mathcal{H}$ embedding of the input dataset and the canonical feature space embedding of the dataset have the same geometry, i.e. $\langle \phi(x), \phi(y) \rangle = \langle \Phi(x), \Phi(y) \rangle$ for $x, y \in \mathcal{X}$.

We have shown the main relation between the Hilbert space of the canonical feature map and kernels. Given a kernel, we can find a map from the input space to a Hilbert space, where the inner product between feature maps reproduces the kernel. Conversely, given a map to a Hilbert space, the inner product defines a kernel on the input space. Since the inner product on the Hilbert space obtained by the canonical map Eq. 4.108 reproduces the kernel, we call the obtained feature space the Reproducing Kernel Hilbert Space (RKHS).

The representer theorem The representer theorem is one of kernel theory's most important classical theorems. We want to solve a supervised problem on an input domain $\mathcal{X}$ and a dataset $\mathcal{D} = (x_m, y_m) \in \mathcal{X} \times \mathbb{R}$, where $m = 1, 2, \ldots, M$. We consider a class of model functions $f : \mathcal{X} \to \mathbb{R}$ which we can write as an expansion over the canonical feature maps

$$f(x) = \sum_{l=1}^{\infty} \mu_i \kappa(x, x_i), \quad x_i \in \mathcal{X}. \tag{4.113}$$

Further, we assume that the cost function $\mathcal{C} : \mathcal{X} \to \mathbb{R}$ quantifies the quality of the predictions and includes a monotonically increasing regularisation term. The representer theorem guarantees that the model minimising

the empirical risk (the cost function over the training dataset $\mathcal{R}$) is of the form

$$f_{\text{opt}}(x) = \sum_{m=1}^{M} \nu_m \kappa(x, x_m). \tag{4.114}$$

The main difference between Eq. 4.113 and Eq. 4.114 is that in the first equation we have an infinite sum over $x_j \in \mathcal{X}$ that are not necessarily in the dataset. In contrast, the second sum is only over the canonical maps of the input set elements of the dataset. The representer theorem has many important consequences also in the quantum setting as we shall see in the following sections.

Quantum kernel   Before introducing the quantum kernel, we introduce a Hilbert space of operators. Since we are concerned only with the finite-dimensional case, we can consider density matrices and quantum operators as finite-dimensional matrices. The space of matrices is a vector space. If we further introduce the Hilbert-Schmidt inner product

$$\langle\langle A, B \rangle\rangle = \operatorname{Tr} A^\dagger B, \tag{4.115}$$

the space of matrices and hence also the space of operators becomes a Hilbert space $\mathcal{H}$. Therefore, any encoding

$$\phi : \mathcal{X} \to \mathcal{H}, \quad x \to \rho(x), \quad x \in \mathcal{X}, \quad \rho > 0, \quad \operatorname{Tr} \rho = 1 \tag{4.116}$$

maps the input elements to a feature Hilbert space $\mathcal{H}$. The data-encoding feature map ϕ and the Hilbert-Schmidt inner product in $\mathcal{H}$ then define a quantum kernel

$$\kappa(x, y) = \operatorname{Tr} \rho(x)\rho(y). \tag{4.117}$$

The quantum kernel has an associated canonical feature map Eq. 4.108 that maps the input examples into the RKHS $\mathcal{F}$. In the following, we will study the relation between the operator Hilbert space $\mathcal{H}$ and the RKHS $\mathcal{F}$.

Example 4.10. Let us consider common feature encodings (some of which we discussed in the previous section) and check what types of kernels they produce. In all cases we will have $x \to \rho(x) = |\phi(x)\rangle \langle \phi(x)|$

- *Basis encoding:* In this case we consider $x = i\mathbb{N}$. We have

$$\rho(i) = |i\rangle \langle i|, \tag{4.118}$$

  where $|i\rangle$ denotes the i-th basis state. Since the basis states are orthogonal $|i\rangle \langle j| = \delta_{i,j}$ we have

$$\kappa(i, j) = |\langle i|k\rangle|^2 = \delta_{i,j}^2 = \delta_{i,j}. \tag{4.119}$$

  The basis encoding reproduces the Kronecker delta kernel.

- *Amplitude encoding:* Where we consider normalised complex vectors $x \in \mathbb{C}^{2^n}$ with $|x|_2 = 1$. The amplitude encoding leads to the feature map

$$\rho(x) = |x\rangle \langle x| = \sum_{i,j=1}^{2^n} x_i x_j^* |i\rangle \langle j| \tag{4.120}$$

  The corresponding kernel is

$$\kappa(x, y) = |\langle x|y\rangle|^2 = \sum_{i,j=1}^{2^n} |x_i^* y_j|^2 \delta_{i,j} = |x^\dagger y|_2^2 \tag{4.121}$$

  The amplitude encoding reproduces the quadratic kernel.

- *Repeated amplitude encoding:* We can also repeat the amplitude encoding as follows

$$\rho(x) = |x\rangle\langle x| \otimes |x\rangle\langle x| \otimes \ldots \otimes |x\rangle\langle x|. \tag{4.122}$$

For r repetitions, this leads to the polynomial kernel

$$\kappa(x,y) = |x^\dagger y|^{2r} \tag{4.123}$$

- *Phase encoding:* In the repeated polynomial encoding, we map elements of a vector $x \in [0, 2\pi]^n$ to angles of the Bloch sphere as

$$\rho(x_i) = |\phi(x_i)\rangle\langle\phi(x_i)|, \quad |\phi(x_i)\rangle = \cos(x_i)|0\rangle + \sin(x_i)|1\rangle. \tag{4.124}$$

The phase encoding of the whole vector is then

$$\rho(x) = \rho(x_1) \otimes \rho(x_2) \otimes \ldots \otimes \rho(x_n). \tag{4.125}$$

The associated kernel is

$$\kappa(x,y) = \prod_{i=1}^{n} |\langle\phi(x_i)|\phi(y_i)\rangle|^2 = \prod_{i=1}^{n} |\sin(x_i)\sin(x_i) + \cos(x_i)\cos(x_i)|^2 \tag{4.126}$$

$$= \prod_{i=1}^{n} |\cos(x_i - y_i)|^2.$$

We reproduced the cosine distance kernel.

Quantum models are linear models in embeddings (feature vectors) $\rho(x)$ We begin by rewriting variational quantum models in the operator space formalism. A variational quantum model calculates an expectation value of the form

$$f_\theta(x) = \langle\psi_0| U^\dagger(x) A_\theta U(X) |\psi_0\rangle, \tag{4.127}$$

where $|\psi_0\rangle$ is the initial state, $U(x)$ is the data encoding procedure, A_θ is the variational circuit that is merged with the measurement, and θ are the variational parameters. We now write the function f by using the operator formalism as

$$f_\theta(x) = \operatorname{Tr} A_\theta U(x) |\psi_0\rangle\langle\psi_0| U^\dagger(x) = \operatorname{Tr} A_\theta \rho(x), \quad \rho(x) = U(x) |\psi_0\rangle\langle\psi_0| U^\dagger(x). \tag{4.128}$$

From Eq. 4.128 we immediately see that any quantum circuit of the form Eq. 4.127 is a linear model in the operator Hilbert space. Moreover, since A_θ is Hermitian it can be decomposed as $A_\theta = \sum_\mu a_\mu |\mu\rangle\langle\mu|$, where $|\mu\rangle\langle\mu|$ are projectors on the eigenstates of A and a_μ are the corresponding eigenvalues. Since projectors $|\mu\rangle\langle\mu|$ are also density matrices we have

$$A_\theta = \sum_{x_l \in \mathcal{X}} \mu_l \rho(x_l). \tag{4.129}$$

Therefore, we can associate with each quantum model A_θ an element of the RKHS via the map

$$A_\theta \to \sum_{x_l \in \mathcal{X}} \mu_l \Phi(x_l). \tag{4.130}$$

The quantum RKHS and the space of quantum models are equivalent The previous statement shows that the quantum RKHS $\mathcal{F}$ includes linear models. We now show that functions in the quantum RKHS are quantum models. A general function in the quantum RKHS, $f \in \mathcal{F}$, can be written as

$$f(x) = \sum_k \gamma_k \kappa(x_k, x) \tag{4.131}$$

where κ is now a quantum kernel and $\gamma_k \in \mathbb{R}$. We now use the linearity of the quantum kernel (trace of the embeddings) and express the function $f(x)$ as

$$f(x) = \sum_k \gamma_k \text{Tr}\, \rho(x_k)\rho(x) \tag{4.132}$$

$$= \text{Tr} \sum_k \gamma_k \rho(x_k)\rho(x)$$

$$= \text{Tr}\, A\rho(x)$$

where $A = \sum_k \gamma_k \rho(x_k)$ is a Hermitian operator. The final expression in Eq. 4.133 is exactly the definition of a variational quantum model in the operator space formalism Eq. 4.128.

We have now shown that the space of variational quantum models of the form Eq. 4.128 is equivalent to the quantum RKHS. This equivalence holds for the chosen embedding procedure $\rho(x)$.

We can now invoke the representer theorem and write the model that minimises the regularised empirical risk as a linear combination of the dataset embeddings

$$f_{\text{opt}} = \sum_{m=1}^{M} \alpha_m \text{Tr}\, \rho(x_m)\rho(x) \tag{4.133}$$

$$= \text{Tr} \sum_{m=1}^{M} \alpha_m \rho(x_m)\rho(x) = \text{Tr}\, A_{\text{opt}}\rho(x).$$

The optimal measurement/variational circuit can be expressed as a linear combination of the input space embeddings, i.e. $A_{\text{opt}} = \sum_{m=1}^{M} \alpha_m \rho(x_m)$.

Optimising quantum kernels The kernel method viewpoint on the variational circuits guarantees that the global optimum of the variational problem is linear in dataset feature maps $\rho(x_m)$. Now we will show that the problem of finding the corresponding coefficients α_m is a simple, convex optimisation problem.

We consider a dataset $\mathcal{D} = \{(x_m, y_m), m = 1, 2, \ldots, M\}$, and minimise regularised loss function $L : \mathcal{X} \times \mathcal{Y} \times \mathbb{R} \to [0, \infty)$, i.e. we want to solve the following optimisation problem

$$f_{\text{opt}} = \inf_{f \in \mathcal{F}} \lambda |f|_F^2 + \mathcal{R}_L(f). \tag{4.134}$$

where λ is the regularisation parameter, $|\bullet|_F^2$ denotes the Frobenious norm in the RKHS, and $\mathcal{R}_L(f)$ calculates for a given function f the average loss over the dataset. The representer theorem guarantees that the optimal model has the form

$$f_{\text{opt}} = \sum_{\alpha=1}^{M} \alpha_m \text{Tr}\, \rho(x_,)\rho(x). \tag{4.135}$$

Our optimisation problem now reduces to the problem of determining the linear expansion coefficients α_m. Let us first consider the functional $\mathcal{R}_L(f)$ which can be written as

$$\mathcal{R}_L(f) = \frac{1}{M} \sum_{m=1}^{M} L(x_m, y_m, \sum_{l=1}^{M} \alpha_l \text{Tr}\, \rho(x_m)\rho(x_l)). \tag{4.136}$$

If we now assume that L is a convex function of the network outputs (which is true for typical loss functions), we see that $\mathcal{R}_L(\alpha)$ is also a convex function of α_m. Moreover, $\mathcal{R}_L$ has only M degrees of freedom.

We can express the second part of the optimisation function as

$$|f|_F^2 = \sum_{m,l} \alpha_m \alpha_l \mathrm{Tr}\, \rho(x_m)\rho(x_l) \tag{4.137}$$
$$= \sum_{m,l} \alpha_m \alpha_l \kappa(x_m, x_k) = \alpha^{\mathrm{T}} K \alpha.$$

Since the Gram matrix K is positive-semidefinite, the function of the Frobenius norm is a convex function of α. Therefore, the optimisation problem Eq. 4.134 in the RNKS reduces to an M-dimensional convex optimisation problem

$$\inf_{\alpha \in \mathbb{R}^M} \frac{1}{M} \sum_{m=1}^{M} L\left(x_m, y_m, \sum_{l=1}^{M} \alpha_l \kappa(x_l, x_m)\right) + \lambda \alpha^{\mathrm{T}} K \alpha. \tag{4.138}$$
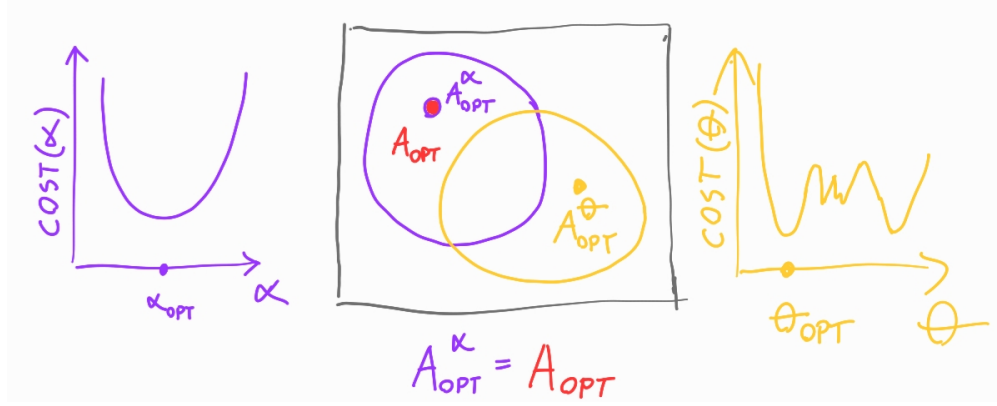

Figure 4.2: Schematic representation of the optimisation problems in the RKHS setting (left) and the variational setting (right). The RKHS optimisation problem is convex, i.e., it has only one global/local minimum. The variational problem is higher dimensional (generally 2^N where N is the number of qubits) and is a non-convex optimisation problem.

We have transformed a potentially hard 2^N dimensional optimisation problem into an M-dimensional convex optimisation problem. Moreover, the solution of the convex problem is guaranteed to be globally optimal (see Fig. 4.2). To find a solution to this problem is still quadratic in M and is hence not practical for large datasets. Still, the theory provides new tools to construct practically relevant quantum machine learning models. The discussed results also determine which types of embedding procedures will not lead to significant quantum speedups.

References

- Schuld, M., & Petruccione, F. (2021). Machine learning with quantum computers (Vol. 676, pp. 163-169). Berlin: Springer. (Chapter: Quantum Models as Kernel Methods)

- Schuld, M., & Killoran, N. (2019). Quantum machine learning in feature Hilbert spaces. Physical review letters, 122(4), 040504.

- Schuld, M. (2021). Supervised quantum machine learning models are kernel methods. arXiv preprint arXiv:2101.11020.

4.8   Barren Plateaus in Random Parameterised Quantum Circuits

In previous lectures, we introduced parameterised quantum circuits (PQCs) and their use in hybrid quantum-classical algorithms such as the Variational Quantum Eigensolver (VQE) and Quantum Approximate Optimisation Algorithm (QAOA). These methods are especially relevant in the context of Noisy Intermediate-Scale Quantum (NISQ) devices, which can execute shallow quantum circuits but remain limited by decoherence and gate fidelity. The core idea of these hybrid algorithms is to optimise a parameterised quantum circuit with respect to a cost function, often expressed as the expectation value of a Hamiltonian or an overlap with a target quantum state. This process is reminiscent of training classical neural networks, where parameters are updated to minimise a loss function. When physical insight is available, the circuit ansatz can be tailored to the structure of the problem. For example, the Unitary Coupled Cluster ansatz is often used in quantum chemistry. However, a structured ansatz may not be feasible in many machine learning applications or under severe hardware constraints. Instead, a generic or random parameterised circuit—sometimes called a "hardware-efficient ansatz"—is used. While random PQCs are easy to implement and hardware-friendly, they introduce a major challenge: the *barren plateau* problem. In such circuits, the optimisation landscape often features large regions where the cost function gradient is exponentially small in the number of qubits. This makes gradient-based optimisation methods ineffective.

This phenomenon can be understood through the lens of concentration of measure. In high-dimensional spaces, most points are close to the mean value of any smooth function. In the quantum setting, this implies that expectation values and gradients of random observables over random states concentrate near their average. This effect has been formalised using tools such as Levy's lemma.

Importantly, the flat regions of the cost landscape do not correspond to useful local minima, but to large, featureless regions with values close to the global average (such as that of the maximally mixed state). Numerical and analytical results show that even relatively shallow circuits, with depth scaling as $O(n^{1/d})$ on a d-dimensional layout, can enter this barren regime.

In this section, we will analyse the mathematical structure behind this gradient vanishing phenomenon and explore how it influences the trainability of PQCs. In the following section, we will discuss practical strategies to mitigate barren plateaus, particularly problem-inspired ansätze and adiabatic optimisation.

To explicitly verify the vanishing gradient and its exponential suppression in variational quantum circuits (RPQCs), we examine the expectation value and variance of the gradient under design assumptions.

We will discuss random parameterised quantum circuits (RPQCs)

$$U(\vec{\theta}) = U(\theta_1, ..., \theta_L) = \prod_{l=1}^{L} U_l(\theta_l) W_l \tag{4.139}$$

where $U_l(\theta_l) = \exp\left(-i\theta_l V_l\right)$, V_l is a hermitian operator, and W_l is a generic unitary operator that does not depend on any angle θ_l. Circuits of this form are a natural choice due to a straightforward evaluation of the gradient with respect to most objective functions. We have discussed the evaluation of the gradients in previous sections. Consider an objective function $E(\theta)$ expressed as the expectation value over some hermitian operator H,

$$E(\vec{\theta}) = \langle 0 | U(\vec{\theta})^\dagger H U(\vec{\theta}) | 0 \rangle . \tag{4.140}$$

When the RPQCs are parameterised in this way, the gradient of the objective function takes a simple form:

$$\partial_k E \equiv \frac{\partial E(\vec{\theta})}{\partial \theta_k} = i \langle 0 | U_-^\dagger \left[V_k, U_+^\dagger H U_+ \right] U_- | 0 \rangle \tag{4.141}$$

where we introduce the notations $U_- \equiv \prod_{l=0}^{k-1} U_l(\theta_l) W_l$, $U_+ \equiv \prod_{l=k}^{L} U_l(\theta_l) W_l$, and henceforth drop the subscript k from $V_k \to V$ for ease of exposition. Finally, we will define our RPQCs $U(\vec{\theta})$ to have the

property that for any gradient direction $\partial_k E$ defined above, the circuit implementing $U(\vec{\theta})$ is sufficiently random such that either U_-, U_+, or both match the Haar distribution up to the second moment. The circuits U_- and U_+ are independent.

We will use the properties of the Haar measure on the unitary group $d\mu_{\text{Haar}}(U) \equiv d\mu(U)$, which is the unique left- and right-invariant measure such that

$$\int_{U(N)} d\mu(U) f(U) = \int d\mu(U) f(VU) = \int d\mu(U) f(UV) \tag{4.142}$$

for any $f(U)$ and $V \in U(N)$, where the integration domain will be implied to be $U(N)$ when not explicitly listed. While this property is valuable for proofs, quantum circuits that exactly achieve this invariance generically require exponential resources. This motivates the concept of unitary t-designs, which satisfy the above properties for restricted classes of $f(U)$, often requiring only modest polynomial resources. Suppose $\{p_i, V_i\}$ is an ensemble of unitary operators, with unitary V_i being sampled with probability p_i. The ensemble $\{p_i, V_i\}$ is a k-design if

$$\sum_i p_i V_i^{\otimes t} \rho (V_i^\dagger)^{\otimes t} = \int d\mu(U) U^{\otimes t} \rho (U^\dagger)^{\otimes t}. \tag{4.143}$$

This definition is equivalent to the property that if $f(U)$ is a polynomial of at most degree t in the matrix elements of U and at most degree t in the matrix elements of U^*, then averaging over the t-design $\{p_i, V_i\}$ will yield the same result as averaging over the unitary group with the respect to the Haar measure.

The average value of the gradient is a concept that requires additional specification because, for a given point, the gradient can only be defined in terms of the circuit that led to that point. We will use a practical definition that leads to the value we are interested in, namely

$$\langle \partial_k E \rangle = \int dU p(U) \partial_k \langle 0| U(\vec{\theta})^\dagger H U(\vec{\theta}) |0\rangle \tag{4.144}$$

where $p(U)$ is the probability distribution function of U.

Expectation Value of the Gradient By the RPQC definition, the full unitary $U = U_+ U_-$ is composed of two independently distributed unitaries, with either U_+ or U_- forming at least a 2-design. The independence implies the joint distribution factorises as:

$$p(U) = \int dU_+ \, p(U_+) \int dU_- \, p(U_-) \, \delta(U_+ U_- - U). \tag{4.145}$$

Using this, the average gradient becomes:

$$\langle \partial_k E \rangle = i \int dU_- \, p(U_-) \operatorname{Tr} \left\{ \rho_- \int dU_+ \, p(U_+) \left[V, U_+^\dagger H U_+ \right] \right\}. \tag{4.146}$$

Assuming Haar-randomness up to the first moment (1-design), we utilise the identity:

$$\int d\mu(U) \, U O U^\dagger = \frac{\operatorname{Tr} O}{N} I, \tag{4.147}$$

where $N = 2^n$ and O is any operator.

Case 1: U_- is a 1-design   Define $\rho_- = U_- \left| 0 \right\rangle \left\langle 0 \right| U_-^\dagger$. Then:

$$\langle \partial_k E \rangle = i \int d\mu(U_-) \operatorname{Tr} \left\{ \rho_- \left[V, \int dU_+ \, p(U_+) \, U_+^\dagger H U_+ \right] \right\}$$
$$= \frac{i}{N} \operatorname{Tr} \left\{ \left[V, \int dU_+ \, p(U_+) \, U_+^\dagger H U_+ \right] \right\} = 0. \tag{4.148}$$

Case 2: U_+ is a 1-design

$$\langle \partial_k E \rangle = i \int dU_- \, p(U_-) \operatorname{Tr} \left\{ \rho_- \int d\mu(U_+) \left[V, U_+^\dagger H U_+ \right] \right\}$$
$$= i \frac{\operatorname{Tr} H}{N} \int dU_- \, p(U_-) \operatorname{Tr} \left\{ \rho_- \left[V, I \right] \right\} = 0. \tag{4.149}$$

Variance of the Gradient   The variance is defined by:

$$\operatorname{Var}[\partial_k E] = \left\langle (\partial_k E)^2 \right\rangle, \tag{4.150}$$

given that $\langle \partial_k E \rangle = 0$. For Haar averages up to the second moment (2-design), we use the integration formula:

$$\int d\mu(U) \, U_{i_1 j_1} U_{i_2 j_2} U^*_{i'_1 j'_1} U^*_{i'_2 j'_2} = \frac{\delta_{i_1 i'_1} \delta_{i_2 i'_2} \delta_{j_1 j'_1} \delta_{j_2 j'_2} + \delta_{i_1 i'_2} \delta_{i_2 i'_1} \delta_{j_1 j'_2} \delta_{j_2 j'_1}}{N^2 - 1} \tag{4.151}$$

$$- \frac{\delta_{i_1 i'_1} \delta_{i_2 i'_2} \delta_{j_1 j'_2} \delta_{j_2 j'_1} + \delta_{i_1 i'_2} \delta_{i_2 i'_1} \delta_{j_1 j'_1} \delta_{j_2 j'_2}}{N(N^2 - 1)}. \tag{4.152}$$

We evaluate three distinct scenarios:

Case 1: U_- is a 2-design

$$\operatorname{Var}[\partial_k E] = \frac{2 \operatorname{Tr}(\rho^2)}{N^2} \operatorname{Tr} \left\langle H_u^2 V^2 - (H_u V)^2 \right\rangle_{U_+} = -\frac{\operatorname{Tr}(\rho^2)}{2^{2n}} \operatorname{Tr} \left\langle [V, H_u]^2 \right\rangle_{U_+}, \tag{4.153}$$

where $H_u = u^\dagger H u$.

Case 2: U_+ is a 2-design

$$\operatorname{Var}[\partial_k E] = \frac{2 \operatorname{Tr}(H^2)}{N^2} \operatorname{Tr} \left\langle \rho_u^2 V^2 - (\rho_u V)^2 \right\rangle_{U_-} = -\frac{\operatorname{Tr}(H^2)}{2^{2n}} \operatorname{Tr} \left\langle [V, \rho_u]^2 \right\rangle_{U_-}, \tag{4.154}$$

with $\rho_u = u \rho u^\dagger$.

Case 3: Both U_- and U_+ are 2-designs

$$\operatorname{Var}[\partial_k E] = 2 \operatorname{Tr}(H^2) \operatorname{Tr}(\rho^2) \left(\frac{\operatorname{Tr}(V^2)}{2^{3n}} - \frac{\operatorname{Tr}(V)^2}{2^{4n}} \right). \tag{4.155}$$

All three cases reveal exponential decay in n, highlighting the severity of barren plateaus in high-dimensional quantum circuits.

4.8.1 Contrast with Gradients in Classical Deep Networks

Finally, we contrast our results with the vanishing (and exploding) gradient problems observed in classical deep neural networks. Two key differences emerge in the quantum case: (i) the scaling behaviour of the vanishing gradients, and (ii) the complexity involved in estimating expectation values.

In classical networks, gradients typically vanish (or explode) exponentially with the number of *layers*. In contrast, in quantum circuits, gradients decay exponentially with the number of *qubits*, which is generally a much more severe form of decay for fixed-depth architectures. This arises due to the structure of the parameter landscape. In classical deep networks, the gradient for a particular parameter depends on the sum over exponentially many paths connecting input to output neurons, where cancellation occurs due to random initialisation. Similarly, in quantum circuits, the gradient depends on interference among exponentially many computational paths, and the amplitude signs fluctuate randomly. Due to normalisation, this leads to gradient magnitudes that saturate at exponentially small values in the number of qubits.

Moreover, gradient *estimation* differs significantly. In classical training, the batch gradient is estimated numerically and is limited by machine precision, scaling logarithmically as $O(\log(1/\epsilon))$ in the desired error ϵ. Small gradients are still trackable across batches, allowing for eventual convergence. In contrast, quantum gradients must be estimated via repeated circuit evaluations, and the estimation error scales as $O(1/\epsilon^\alpha)$, with $\alpha \geq 1$ depending on the estimator. If the number of measurements is insufficient to resolve the exponentially small signal, the result is effectively indistinguishable from noise, leading to a random walk in parameter space.

By the concentration of measure, such a walk will have an exponentially small probability of escaping the barren plateau. Hence, without additional algorithmic strategies (e.g., tailored ansätze, parameter initialisation schemes, or alternative cost functions), gradient descent in quantum machine learning cannot overcome this obstacle in polynomial time. This highlights a fundamental barrier to naively porting classical training paradigms to quantum systems.

References

- McClean, J. R., Boixo, S., Smelyanskiy, V. N., Babbush, R., & Neven, H. (2018). Barren plateaus in quantum neural network training landscapes. Nature communications, 9(1), 4812.

4.9 Quantum Convolutional Neural Networks

In the previous section, we have seen that in general, random PQC exhibit a barren plateau problem. In this section, we will consider a specific PQC architecture, which avoids the problem of barren plateaus and performs well on a variety of supervised machine learning tasks.

We take inspiration from classical machine learning. Convolutional neural networks (CNNs) have demonstrated remarkable success in classical machine learning tasks such as image classification, leveraging architectural elements like convolution and pooling to extract hierarchical and translationally invariant features from data. In a typical CNN, feature maps evolve through layers of local convolutions—weighted sums over small neighbourhoods—and pooling, which reduces dimensionality by downsampling key activations. Nonlinear activation functions are used between layers to increase expressiveness, and the final classification is made via a fully connected layer. The number of learnable parameters is typically independent of input size, enabling efficient scaling.

Inspired by this structure, we introduce a quantum analogue: the quantum convolutional neural network (QCNN), which adapts the core principles of CNNS to quantum systems. The QCNN is composed of

alternating convolution and pooling layers followed by a fully connected unitary layer, and is designed to act on an n-qubit quantum state ρ_{in}.

In this quantum model, each convolution layer applies a translationally invariant pattern of two-qubit unitary gates (W), akin to sliding convolution kernels. The pooling layers implement a reduction of degrees of freedom by performing measurements on specific qubits, with outcomes conditioning unitaries (I) on neighbouring qubits, thus introducing a nonlinearity. This sequence is repeated for L layers until only a few qubits remain. A fully connected unitary F is then applied, and a final observable O is measured on the resulting state ρ_{out}.

Like classical CNNs, the QCNN architecture is parameter-efficient: only $O(\log n)$ parameters are required for an n-qubit input, representing an exponential reduction relative to generic quantum circuits. This efficiency makes QCNNs particularly suitable for scalable quantum learning tasks.

Given a training dataset of labelled quantum states $\{(|\psi_\alpha\rangle, y_\alpha)\}_{\alpha=1}^{M}$ with $y_\alpha \in \{0, 1\}$, the QCNN can be trained to minimise a loss function such as the mean squared error:

$$\text{MSE} = \frac{1}{2M} \sum_{\alpha=1}^{M} (y_\alpha - f_{\{W,I,F\}}(|\psi_\alpha\rangle))^2, \tag{4.156}$$

where $f_{\{W,I,F\}}(|\psi_\alpha\rangle)$ denotes the expectation value of the output measurement.

To understand the theoretical underpinnings of QCNNs, we can relate their structure to two powerful concepts in quantum information and tensor networks (discussed in the following sections): the multiscale entanglement renormalisation ansatz (MERA) and quantum error correction (QEC). MERA efficiently represents many-body quantum states (Vidal, 2007; Aguado and Vidal, 2008; Pfeifer et al., 2009) through alternating layers of unitaries and isometries, building long-range entanglement hierarchically. The QCNN reverses this structure, effectively collapsing entanglement while preserving relevant features.

Furthermore, the measurement and feedback mechanism in pooling layers parallels QEC: measurement outcomes can be interpreted as syndromes, with corresponding unitaries acting to correct localised errors (Preskill, 1998). In this sense, QCNNs implement a hybrid between MERA and nested QEC protocols, allowing them to detect and correct deviations from target states or phases.

This synergy enables QCNNs to classify quantum phases and detect quantum phase transitions with exponentially fewer measurements than traditional observables. For instance, a QCNN trained to recognise a representative ground state $|\psi_0\rangle$ of a given phase can map perturbed versions of $|\psi_0\rangle$ back toward it via repeated correction, effectively simulating a renormalisation-group flow. As a result, QCNNs offer a promising architecture for both practical quantum machine learning and foundational studies of quantum many-body systems.

Importantly, it has been shown that the variance of the derivatives of the cost function vanishes only polynomially with the system size. However, more recent studies show that due to the same reason, the QCNN can be trained and evaluated classically. This is in agreement with a more general result, which argues that current PQCs without barren plateaus are classically simulable and can not provide any benefit with respect to classical machine learning methods.

In the next section, we will discuss a model-agnostic approach to combat barren plateaus, which will combine the insights from adiabatic quantum computing and parametrised quantum circuits.

References

- Cong, I., Choi, S., & Lukin, M. D. (2019). Quantum convolutional neural networks. Nature Physics, 15(12), 1273-1278.

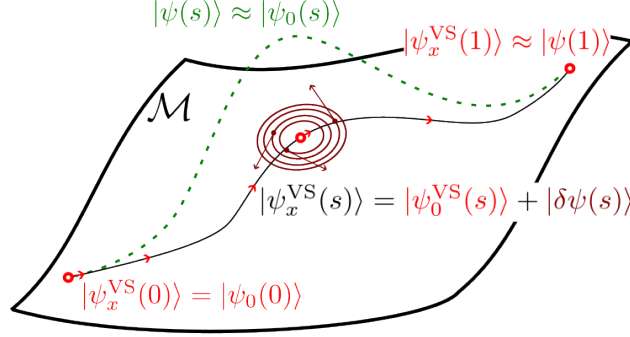## 4.10 Variational Ground-State Adiabatic Theorem and Barren Plateaus in PQCs

Figure 4.3: Schematic representation of the variational ground-state quantum adiabatic theorem. The dashed green line represents the exact quantum adiabatic evolution $|\psi(s)\rangle \approx |\psi_0(s)\rangle$, visiting high-entanglement regions. The solid black line corresponds to the instantaneous variational ground state $\left|\psi_0^{\mathrm{VS}}(s)\right\rangle$, with the small red arrows indicating that it changes slowly with time. The time-dependent variational state $\left|\psi_x^{\mathrm{VS}}(s)\right\rangle$ is separated into $\left|\psi_0^{\mathrm{VS}}(s)\right\rangle$ and their difference $|\delta\psi(s)\rangle$. The latter can be decomposed in the eigenbasis of the linearised evolution map, whose eigenvalues ω_l determine the frequencies of the elliptic trajectories around the instantaneous variational ground state. Since $1/T \ll \omega_l$, the fast dynamics of $|\delta\psi(s)\rangle$ (denoted by long, dark-red arrows) averages to zero, and the time-dependent variational state $\left|\psi_x^{\mathrm{VS}}(s)\right\rangle$ follows the instantaneous variational ground state $\left|\psi_0^{\mathrm{VS}}(s)\right\rangle$. When the target ground state of the final Hamiltonian lies within the variational manifold, the time-dependent variational state will converge to it as the annealing time T becomes large.

Parameterised quantum circuits (PQCs) have emerged as a leading framework for variational quantum algorithms (VQAs), such as the variational quantum eigensolver (VQE) and the quantum approximate optimisation algorithm (QAOA). Despite their versatility, PQCs suffer from the *barren plateau* problem—an exponentially vanishing gradient in the cost landscape—which severely limits the scalability of variational methods. This phenomenon has been analysed in depth by McClean et al. (2018) and Cerezo et al. (2021), where it was shown that overly expressive or random ansätze tend to generate states close to the Haar measure, leading to vanishing gradients.

A promising approach to mitigating this issue is rooted in the *variational ground-state quantum adiabatic theorem*. This strategy emulates adiabatic quantum computation (AQC) by evolving a parameterised quantum state slowly along a path determined by a time-dependent Hamiltonian $H(s)$, where $s = t/T$ is the normalised time. The conventional adiabatic theorem, discussed by Albash and Lidar (2018), guarantees that a system prepared in the ground state of H_0 will remain in the ground state of the instantaneous Hamiltonian $H(s)$ if the evolution is sufficiently slow.

In gate-based quantum circuits, we can approximate this continuous evolution within a variational manifold $\mathcal{M}$ defined by a PQC ansatz. The resulting variational trajectory $\left|\psi_x^{\mathrm{VS}}(s)\right\rangle$, governed by the time-dependent variational principle (TDVP), evolves in a physically meaningful subspace that avoids the random sampling responsible for barren plateaus.

As illustrated in Fig. 4.3, the variational state follows the instantaneous variational ground state $\left|\psi_0^{\mathrm{VS}}(s)\right\rangle$ with high fidelity if the initial and final ground states both lie within $\mathcal{M}$. The deviation $|\delta\psi(s)\rangle = \left|\psi_x^{\mathrm{VS}}(s)\right\rangle - \left|\psi_0^{\mathrm{VS}}(s)\right\rangle$ is of order $\mathcal{O}(1/T)$, provided that the annealing time T is large compared to the inverse of the

lowest excitation frequencies in the variational subspace (see Hackl et al., 2020).

This variational approach can be adopted with classical and quantum ansätze. Here we will consider quandum PQCs. The main question in this context is what happens with the gradients in the vicinity of the variational ground states. In this regard, we consider a hardware-efficient ansatz exhibiting barren plateaus and use it to solve an Ising spin glass problem, which is computationally demanding. We first calculate the variational ground state by using a variant of gradient descent, and then calculate the energy standard deviation around the obtained variational minimum. The Fig. 4.4 shows the result at different points in the annealing procedure. In all cases, we find that close to the variational ground state, the standard deviation remains independent of the system size, while at random points $\Delta\theta \approx 2\pi$ it vanishes exponentially.
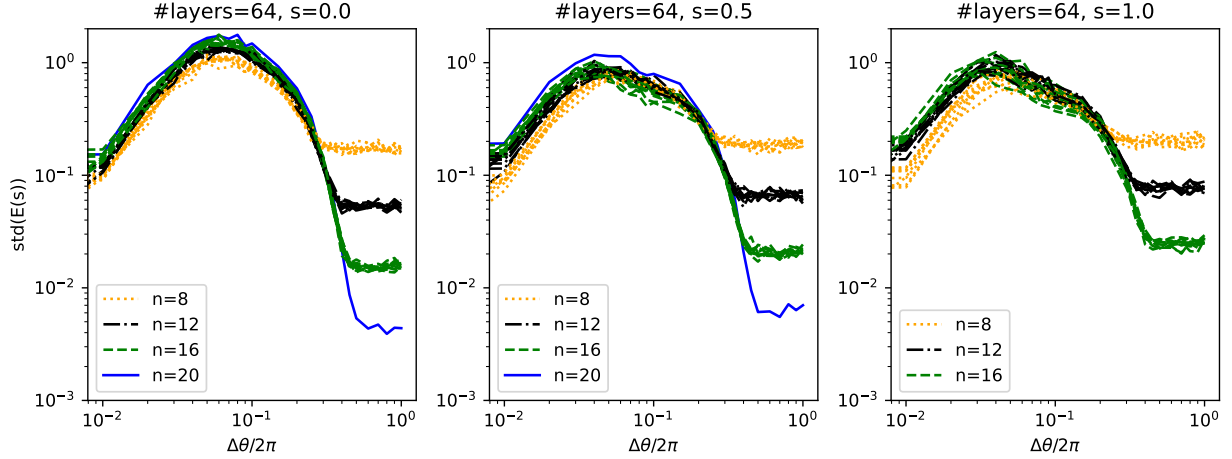
Figure 4.4: Scaling of the standard deviation of the loss (in this case, the energy of the model) close to the variational ground-state obtained by a time-dependent gradient descent method. In a finite region shrinking as a square root of the number of parameters, the standard deviation of the loss function and hence the gradient, is independent of the number of qubits. The PQC used to produce these results exhibits the barren plateaus problem. However, by following the adiabatic path, we are able to follow a particular local minimum. $\Delta\theta$ is the size of the hypercube around the variational ground state in which the standard deviation is calculated. If $\Delta\theta$ is very small, we are in a linear regime close to the variational ground state, where the standard deviation grows quadratically, and if $\Delta\theta$ is 2π, we essentially choose a random point in the parameter space.

This simple example demonstrates that we can circumvent the barren plateau problem by following the adiabatic path.

References

- Unpublished work
- Žunkovič, B., Torta, P., Pecci, G., Lami, G., & Collura, M. (2025). Variational ground-state quantum adiabatic theorem. Physical Review Letters, 134(13), 130601.

4.11   Power of data in quantum machine learning

In the previous sections, we considered specific training problems with quantum machine learning models that arise from the concentration of measure in expressive quantum ansatzes. In this section, we will discuss the effect of data in the context of quantum simulation and how this translates to quantum machine learning. We will see that data itself can elevate classical models to rival or even surpass quantum approaches without data, even when the underlying quantum circuits are classically intractable. Consequently, the key to understanding quantum advantage in machine learning lies not only in computational complexity but also in the geometric and statistical properties of the data. This perspective motivates a principled framework to assess prediction advantages, using tools such as kernel methods and geometric tests, to quantify when quantum models offer meaningful improvements over classical counterparts.

We consider a supervised learning task with a set of N training examples $\{(x_i, y_i)\}$, where x_i are input vectors sampled i.i.d. from a distribution $\mathcal{D}$ and $y_i \in \mathbb{R}$ are real-valued outputs generated by a quantum model. Each input x_i is mapped into a quantum state using a continuous data-encoding unitary $U_{\text{enc}}(x_i)$ acting on the n-qubit initial state $|0\rangle^{\otimes n}$, resulting in the state $|x_i\rangle = U_{\text{enc}}(x_i) |0\rangle^{\otimes n}$, with density matrix $\rho(x_i)$.

A quantum neural network (QNN), defined by a parametrised unitary $U_{\text{QNN}}(\theta)$, is applied to the encoded state, followed by a measurement of an observable O. The expected value gives the output label:

$$f(x_i) = \langle x_i | U_{\text{QNN}}^{\dagger} O U_{\text{QNN}} |x_i\rangle .$$

While computing $f(x)$ can be classically intractable in general, as we now demonstrate, the presence of training data changes the problem substantially.

Hardness Without Data.   Suppose we use amplitude encoding for classical vectors $x_i \in \mathbb{R}^p$ with $\|x_i\|_2 = 1$, yielding quantum states:

$$|x_i\rangle = \sum_{k=1}^{p} x_i^k |k\rangle .$$

For a general QNN defined by time evolution under a many-body Hamiltonian, computing $f(x)$ is known to be classically hard and has been discussed in the context of Hamiltonian simulation. More precisely:

Proposition 1. If a classical algorithm without training data can efficiently compute $f(x)$ for arbitrary U_{QNN} and O, then BPP = BQP.

Ease With Data.   Despite this hardness in the absence of data, training a classical model to approximate $f(x)$ may be straightforward when training data is available. Expanding the expectation, we find:

$$\begin{aligned} f(x_i) &= \left(\sum_{k=1}^{p} x_i^{k*} \langle k| \right) U_{\text{QNN}}^{\dagger} O U_{\text{QNN}} \left(\sum_{l=1}^{p} x_i^l |l\rangle \right) \\ &= \sum_{k=1}^{p} \sum_{l=1}^{p} B_{kl} x_i^{k*} x_i^l, \end{aligned} \tag{4.157}$$

where $B_{kl} = \langle k| U_{\text{QNN}}^{\dagger} O U_{\text{QNN}} |l\rangle$ are fixed coefficients. Thus, $f(x)$ is a quadratic function in the input components.

Using classical regression techniques, one can fit such a function with high accuracy.

Testing Quantum Advantage Continuing our exploration of quantum machine learning, we now develop a general framework to assess the potential for quantum prediction advantage in supervised learning tasks. This approach builds on the geometry of learning spaces and results in practical tests for quantum superiority.

We begin by considering a quantum model defined by the function

$$f(x) = \text{Tr}(O^U \rho(x)), \quad \text{with } O^U = U_{\text{QNN}}^\dagger O U_{\text{QNN}},$$

where $\rho(x)$ is the input state and O is an observable. Suppose we are given N training samples $\{(x_i, y_i = f(x_i))\}$.

After training on this data, there exists an ML algorithm that outputs $h(x) = w^\dagger \phi(x)$, where $\phi(x)$ is a feature map and $k(x_i, x_j) = K_{ij} = \phi(x_i)^\dagger \phi(x_j)$ is the kernel, which has a prediction error bounded by:

$$\mathbb{E}_{x \sim \mathcal{D}} |h(x) - f(x)| \leq c \sqrt{\frac{s_K(N)}{N}},$$

for some constant $c > 0$, where $s_K(N)$ quantifies the model complexity. The model complexity is defined as:

$$s_K(N) = \sum_{i,j=1}^{N} (K^{-1})_{ij} \, \text{Tr}(O^U \rho(x_i)) \, \text{Tr}(O^U \rho(x_j)).$$

After training we have $s_K(N) = \|w\|^2$. Smaller $s_K(N)$ implies better generalisation, reflecting how well the kernel geometry aligns with the structure of the data labels. The derivation of this result is out of the scope of the lectures (for now). We will discuss only the implications.

Computational Aspects: Given measurements of $\text{Tr}(O^U \rho(x_i))$, $s_K(N)$ can be computed by inverting the $N \times N$ kernel matrix K, requiring $\mathcal{O}(N^3)$ time classically.

Regularization: To prevent overfitting, a regularization term $\|w\|^2$ is often added during training. This ensures $s_K(N)$ remains bounded even if the model does not perfectly fit the training data.

Implications for Quantum Advantage If a classical model achieves a small $s_K(N)$ on a quantum dataset, then it can predict $f(x)$ accurately, even if $f(x)$ is hard to compute classically. Hence, the potential for quantum advantage depends on minimising $s_K(N)$ over classical models.

To assess this, we define an asymmetric geometric quantity:

$$g_{12} = g(K^1 \| K^2) = \sqrt{\|\sqrt{K^2}(K^1)^{-1}\sqrt{K^2}\|_\infty},$$

where K^1 and K^2 are the kernel matrices of two models, and the spectral norm $\|\cdot\|_\infty$ captures maximal directional discrepancy. This leads to the inequality:

$$s_{K^1} \leq g_{12}^2 s_{K^2},$$

indicating how well model K^1 can perform relative to model K^2.

Classical vs Quantum Models: Define $g_{\text{CQ}} = g(K^{\text{C}} \| K^{\text{Q}})$. If $g_{\text{CQ}} \approx 1$, classical ML can match quantum ML performance. If $g_{\text{CQ}} \gg 1$, quantum methods may yield significantly better predictions.

Constructive Insight: When g_{CQ} is large, one can explicitly construct a dataset on which quantum models outperform classical ones. In practice, g_{CQ} is minimised over a suite of classical methods to find the best classical "adversary."

Role of Effective Dimension in Quantum Kernel Methods

Consider quantum kernel methods with $k^Q(x_i, x_j) = \text{Tr}(\rho(x_i)\rho(x_j))$. Let $\text{vec}(X)$ denote the vectorization of matrix X, then:

$$s_Q = \text{vec}(O^U)^T P_Q \text{vec}(O^U),$$

where P_Q is the projector onto the subspace spanned by $\{\text{vec}(\rho(x_i))\}_{i=1}^N$.

Define the effective dimension:

$$d = \text{rank}(K^Q) = \dim(P_Q) \leq N.$$

Since P_Q is a projector:

$$s_Q \leq \min(d, \text{Tr}(O^2)),$$

and the prediction error becomes:

$$\mathbb{E}_{x \sim \mathcal{D}}|h(x) - f(x)| \leq c\sqrt{\frac{\min(d, \text{Tr}(O^2))}{N}}.$$

Computational Note: Both d and g_{CQ} can be computed classically via singular value decomposition on the $N \times N$ kernel matrices.

Concluding Remarks on Predictive Advantage

When both g_{CQ} and $\min(d, \text{Tr}(O^2))$ are small, we conclude that classical models can learn the target function effectively, regardless of the quantum circuit depth.

Finally, quantum advantage arises only when $s_C \gg s_Q$—that is, when quantum models align well with the function $f(x)$ but classical models do not. The geometric and model-complexity tests described provide a principled, data-driven methodology for assessing this possibility.

References

- Huang, H. Y., Broughton, M., Mohseni, M., Babbush, R., Boixo, S., Neven, H., & McClean, J. R. (2021). Power of data in quantum machine learning. Nature communications, 12(1), 2631.

5 Quantum-inspired machine learning

In this section, we will introduce two frameworks/methods originating from quantum mechanics and apply them to machine learning problems. First, we will discuss tensor networks, which are an efficient way to represent low-entanglement many-body quantum states variationally. Then we will look at dequantisation methods and algorithms that assume sample and query access to input vectors.

5.1 Tensor networks

A tensor network is a set of tensors which are contracted to form a new tensor. Common operations like matrix multiplication and scalar product can be interpreted as tensor networks where we contract two tensors in order to get a new one. In this context also a number is a zero-dimensional tensor.

Diagramatic notation   A standard notation that helps us to visualise operations on tensors and keep track of indices is the diagramatic notation. The tensors are diagramatically represented as shapes (circles, squares, triangles) with legs. The number of legs determines the dimension of the tensor (see table 5). For example, a number can be represented as a circle without legs and a vector as a circle with one leg.

Object	Index	Diagrammatic
Number	a	
Vector	a_i	
Matrix	$M_{i,j}$	
3rd order Tensor	$T_{i,j,k}$	

Table 5: Comparison of diagrammatic and index notation of low-order tensors.
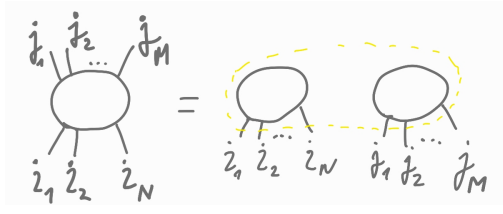
Typically, we do not write indices and names of the tensors in the diagrammatic notation, since this is clear from the context/drawing and does not provide any new information. Also, the orientation of the legs does not matter unless we use a convention, e.g., the legs pointing up in a vector means a complex conjugate of the legs pointing down. In most literature, no such rule is assumed.

The primary operations we can do on tensors are tensor product, contractions, trace, and grouping/splitting of indices.

Tensor product   is the most common operation when we are dealing with many-body quantum systems. The reason is that quantum mechanics has a tensor product structure. If one system is described with a quantum probability vector $a \in \mathcal{H}_1$ and the other with a vector $b \in \mathcal{H}_1$, then the composite system is described with a tensor product $c = a \otimes b \in \mathcal{H}_1 \otimes \mathcal{H}_2$. Tensor product of tensors A and B is defined as

$$[A \otimes B]_{i_1,\ldots,i_N,j_1,\ldots j_M} = A_{i_1,\ldots i_N} B_{j_1,\ldots j_M}. \tag{5.1}$$

In the diagrammatic notation, the tensor product of tensors is represented as tensors drawn one after another (without any legs connected)

$$\tag{5.2}$$

The yellow dashed line is drawn for convenience and is usually omitted. We also omit the indices of legs since they do not provide any new information.

Contractions   A generalisation of the tensor product is a contraction. We not only construct a new tensor by multiplying elements from different tensors, but also sum over contracted indices. A simple example is matrix multiplication

$$A_{ij} = \sum_k B_{ik} C_{kj}. \tag{5.3}$$

In the diagrammatic notation, the contraction is represented by joining the corresponding legs of different tensors. The matrix multiplication example is then

$$i\!-\!\!\bigcirc\!\!-\!j = i\!-\!\!\bigcirc\!\!\bigcirc\!\!-\!j$$

. (5.4)

Trace A contraction of indices on the same tensor is called a trace. For example, a trace of a matrix is a sum of diagonal elements $\mathrm{Tr}\, A = \sum_i A_{i,i}$. In the diagrammatic notation, this is

$$\bigcirc = \bigcirc\!\!\bigcirc$$

. (5.5)

Generally, we can also take a trace in tensors with more than two indices. In this way, the result is not a number but a tensor with a lower dimension. For example, if we trace a 3rd-order tensor, we obtain a vector

$$a_i = \sum_j T_{i,j,j} \tag{5.6}$$

or in the diagrammatic notation

$$-\!\!\bigcirc = -\!\!\bigcirc\!\!\bigcirc$$

. (5.7)

Reshape: grouping and splitting of indices (*vector = matrix = tensor*) Critical operations that we can apply to tensors are grouping and splitting of indices. In other words, the reshaping of tensors. For example, a matrix with elements $A_{i,j}$ can be represented as a vector v_α as $v_{(iM + j)} = A_{i,j}$, where $i = 0, \ldots N - 1$ and $j = 0, \ldots M - 1$. This vectorisation is not unique. We could equally well vectorise the matrix A as $v_{(jN + i)} = A_{i,j}$. These two options are commonly referred to as column major and row major vectorisations. In the diagrammatic notation, we group indices by simply drawing them close together or with a bold leg

$$-\!\!\bigcirc\!\!-\!\!\rightarrow \bigcirc\!\!= \quad \text{OR} \quad \bigcirc\!\!-\!\!$$

. (5.8)

The size of the new index is equal to the product of the sizes of the grouped indices. In the above matrix-to-vector example, this means that $\alpha = 0, \ldots MN - 1$.

Grouping of indices can also be applied only on a part of the tensor indices, and together with another grouping of indices. For example, a five-dimensional tensor can be transformed to a matrix by grouping two and three indices

$$\bigtimes\!\!\!\rightarrow \;=\!\!\bigcirc\!\!\equiv$$

. (5.9)

Whereas we can always group two indices, one index can not be split into an arbitrary number of indices. This is clear since the reshaped tensor must have the same number of elements. For example, a vector with NM elements can be split into an $N \times M$ matrix.

Singular value decomposition on tensors   Tensor reshaping allows us to reshape any tensor to a matrix (assuming the tensor is not a number or a vector with a prime number of elements). This allows us to apply operations that are typically reserved for matrices to general tensors. The most important operation in this context is the singular value decomposition (SVD). Given a complex matrix A, the SVD results in two unitaries U and V and a real positive diagonal matrix S such that $A = USV^\dagger$. If we keep all the singular values, the decomposition is exact. However, if we keep only the most significant D elements, the decomposition results in a best approximation (measured in Hilbert-Schmidt norm) of the original matrix by a matrix of rank D. In fact, the SVD decomposition decomposes a matrix into a tensor network. If we apply this decomposition repeatedly, we can decompose a high-dimensional tensor as a tensor network of low-dimensional tensors. The most common tensor network that is obtained is the matrix product state (see Example. 5.1)

Example 5.1. Let us consider a tensor T with N legs of size 2. The tensor T has 2^N elements. Our aim is to represent the N dimensional tensor as a contraction of two two-dimensional and $N-2$ three-dimensional tensors A^j, $j = 1, \dots N$
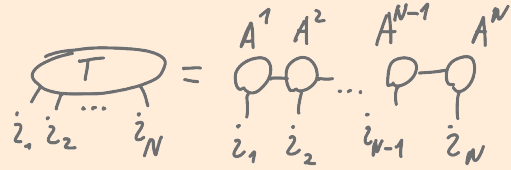

To get the above decomposition, we will successively reshape the largest tensor in the decomposition and apply the SVD. First, we reshape the tensor T into a matrix by grouping the indices $i_2, \dots, i_N$. Then we use the SVD on the matrix. And finally, we contract the matrices S and V from the SVD decomposition. In the diagrammatic notation,n these steps are
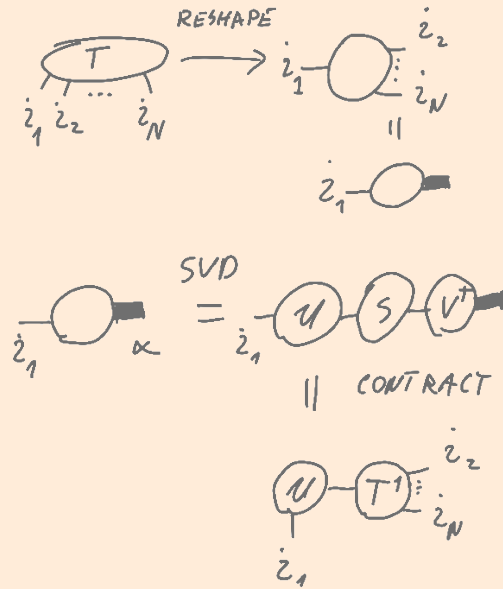
We already obtained the first tensor, namely $A^1 = U$, from the above decomposition. To get the next tensor A^2, we perform similar transformations on T^1, i.e. a matrix obtained by multiplying S and $V^\dagger$ from the

first SVD decomposition. First, we reshape the matrix T^1 by grouping the left index and the index i_2 in the left index of the reshaped matrix and indices $i_3, \ldots, i_N$ in the right index of the reshaped matrix. Then we perform the SVD, contraction of S and V, and reshaping to finally get the tensors A^2 and T^2. We write these steps diagramatically as
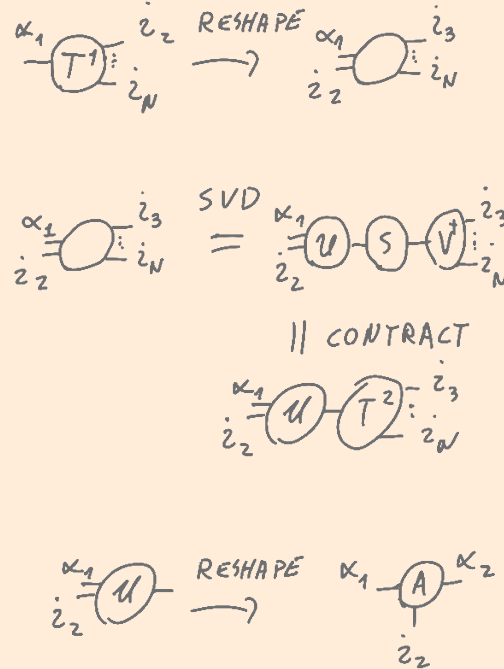
To get the tensor A^3, we perform the same steps as above but on the tensor T^2; similarly, for the remaining tensors up to the last. The last tensor A^N is equal to T^{N-1} since it has only one index on the right side and has therefore already the correct shape.

By keeping all the singular values in the SVD decomposition, we arrive at an exact matrix product state representation of the original tensor. However, if we keep only the first D singular values in each SVD decomposition, we obtain an approximation. These approximations are very common in many-body quantum systems, since they describe physically important quantum probability distributions. In particular, ground states of gapped one-dimensional Hamiltonians.

Tensor network contraction Contracting a general tensor network is an NP-hard problem. The most straightforward argument to see this is to transform the NP-hard graph colouring problem into a tensor network contraction problem.

The graph colouring problem: Given a graph, find the number of different colourings with d colours such that all connected vertices have different colours.

We transform this problem into a tensor network contraction problem by introducing two types of tensors

$$e_{i_1,\dots i_K} = \begin{cases} 1 & i_1 = i_2 = \dots i_K \\ 0 & \text{else} \end{cases} \tag{5.10}$$

$$\eta_{i,j} = \begin{cases} 1 & i \neq j \\ 0 & i = j \end{cases},$$

where $i, j = 1, \dots d$. To each vertex, we attach the tensor e, and to each edge, we attach the tensor η. The tensor e ensures that the contraction of the obtained tensor network will have non-vanishing contributions only if all outgoing indices have the same value (corresponding to the colour). On the other hand, the tensor η ensures that all the connected edges will have a different colour. Hence, the contraction of the obtained network gives exactly the number of allowed colourings with d colours. For an example see Fig. 5.1
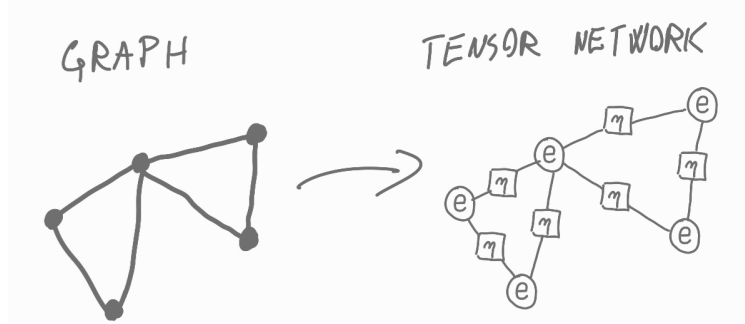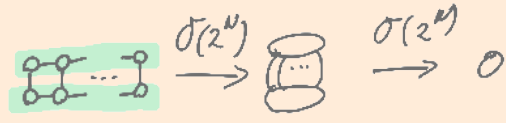

Figure 5.1: Example of the transformation from the graph to a tensor network contraction, which results in the number of different colourings with d colours. The size of each contracted leg in the tensor network diagram is equal to the number of colours d.

Example 5.2. As we have seen, the tensor network contraction problem is NP-hard. However, particular tensor networks have efficient contractions. In this example, we will look at one inefficient and two efficient contractions of a matrix product state norm given by the tensor network diagram

$$\tag{5.11}$$

We get an inefficient contraction of the tensor network Eq. 5.11 if we first contract the inner bonds of the matrix product states and obtain an exponentially large tensor in the number of legs N. The final contraction is then obtained by calculating the 2-norm of that tensor (see Eq. 5.12).

$$\tag{5.12}$$

The first efficient contraction can be done in parallel. First, we calculate the contractions corresponding to the same tensors in the MPS (denoted by green in the first step in Eq. 5.13) to obtain N matrices of size $D^2 \times D^2$. The parallel complexity of this step is $\mathcal{O}(D^4 d)$ since all the contractions can be done in parallel. In the successive steps, we multiply together pairs of neighbouring matrices $\log_2(N)$ times. In each step,

all the calculations can be done in parallel with complexity $\mathcal{O}(D^6)$. The total parallel complexity of this algorithm is hence $\mathcal{O}(\log_2(N)D^6 + D^4 d)$.

$$\mathcal{O}(D^6) \times \log_2 N \qquad (5.13)$$

The final, most efficient, sequential algorithm is given by the steps shown in Eq. 5.14

$$\mathcal{O}(NdD^3) \qquad (5.14)$$

We contract from left to right. In the initial step, performed only once, we contract the leftmost tensors – $\mathcal{O}(dD^2)$. Then we perform the following two contractions $N-2$ times. First, we contract the leftmost tensor with the top remaining matrix product state tensor – $\mathcal{O}(dD^3)$. Then we contract the obtained tensor with the leftmost tensor of the bottom MPS tensor – $\mathcal{O}(D^3 d)$. The final two contractions are similar to the main $N-2$ contractions, with the difference that the remaining MPS tensors do not have the right bond index. The total complexity of this algorithm is hence $\mathcal{O}(NdD^3)$.

Tensor network contractions and quantum computation   The circuit model of quantum computation can be interpreted as a contraction of a tensor network with appropriate unitary constraints. A teleportation example is shown in Fig. 5.2. Hence, efficient classical simulation of quantum circuits is intimately related to tensor network contractions. In order to determine if a given quantum circuit is hard to calculate classically, we have to find its best contraction scheme, which is a hard problem in general.

Tensor networks in machine learning   Tensor networks have been used as a special type of multilinear model similar to variational quantum circuits and as a compact representation of weight matrices in deep neural networks. They have been applied to many supervised, unsupervised, and reinforcement learning problems. Most applications of tensor networks to machine learning problems are proof of principle. So far, the only exceptions are the anomaly detection problem discussed in Example. 5.3 and its extension to positive unlabelled learning.

Example 5.3. The anomaly detection problem is formulated as follows. Given a dataset, the task is to determine if a new, test example is from the same distribution as the training examples. The difference
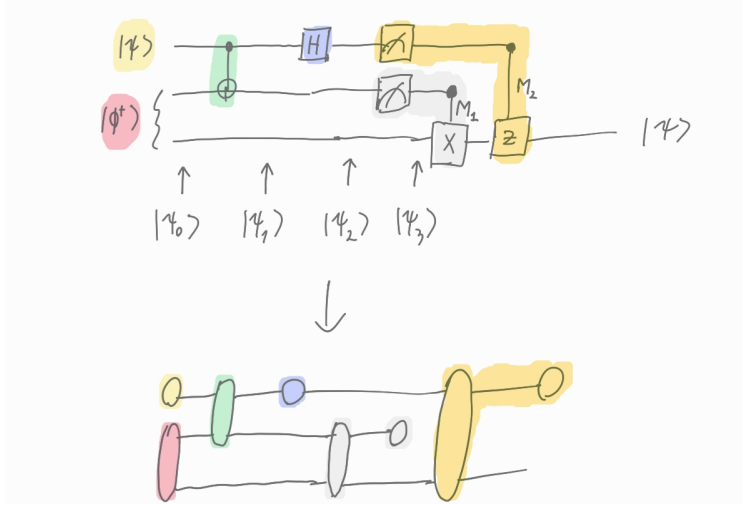
Figure 5.2: Interpreting the teleportation circuit as a tensor network contraction. The initial state is a tensor product of a vector with a 2-dimensional tensor. The following operations act on one or two qubits. The corresponding tensors are two and four-dimensional. The measurement is simulated by a contraction with an appropriate state.

with respect to the standard classification problem is that we do not have out-of-distribution examples for training.

The tensor network solution to this problem is summarised in Fig. 5.3.

1. First, we embed the examples in a high-dimensional Hilbert space $\mathcal{H}_\infty$ by using a data encoding procedure discussed in Section 4.1, $\Phi(x) \in \mathcal{H}_1$.

2. Then we project the encoded example into a new Hilbert space $P\Phi(x) \in H_2$. If the norm of the projected vector is smaller than ϵ, we have an anomaly.

3. The projector is chosen such that its kernel is large. For the choice in Fig. 5.3, the kernel has size $d^{N-\lfloor N/S \rfloor}$, where d is the embedding dimension, N the number of examples, and S determines the size of the Hilbert space $\mathcal{H}_2$.

4. We can use standard gradiend descend based training with the loss $\mathcal{L} = \frac{1}{M} \sum i = 1^M \|\log(D(x)) - 1\|_2 + \alpha \log(\|P\|_F)$. The loss has two contributions. The first decreases if the in-distribution training examples have a norm close to $\sqrt{e}$ after the projection. The second decreases if the kernel of the projector is large. The balance between the two contributions is controlled by a hyperparameter $\alpha > 0$.

5. Finally, we have to ensure that the calculation of the projected norm is efficient. This is clear since after the contractions of the embeddings with the projectors, we obtain a standard matrix product state norm calculation (see Example. 5.2).
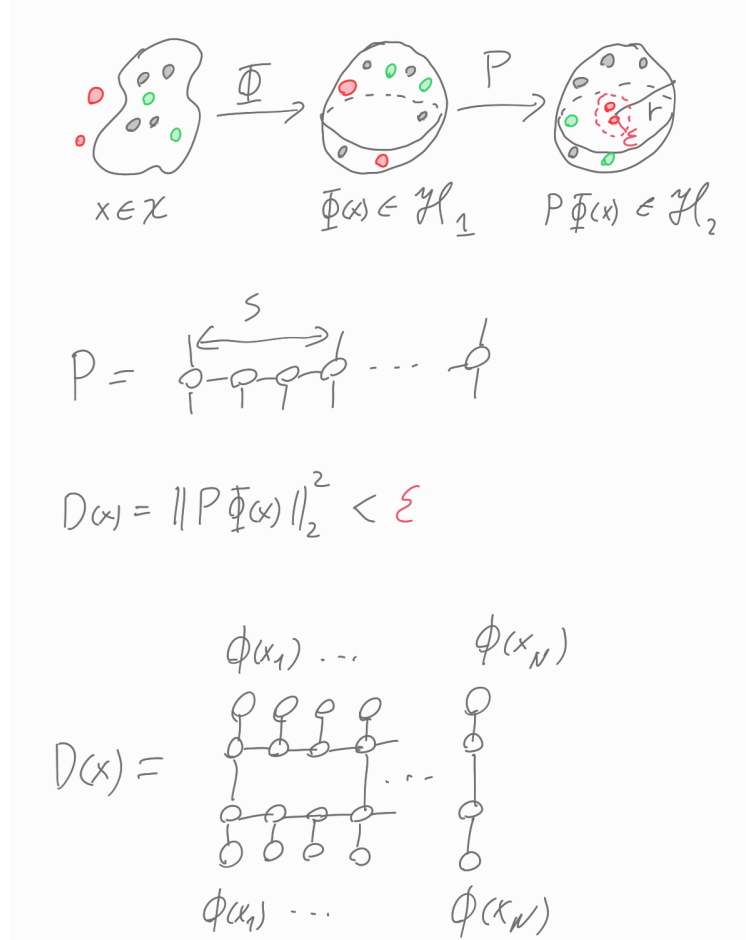
Figure 5.3: The figure shows the idea of anomaly detection with tensor networks. An example is first mapped to a Hilbert $\mathcal{H}_1$ space and then projected to the Hilbert space $\mathcal{H}_2$. If the norm of the projected vector is smaller than ϵ, we have an anomaly (red circles). The projector P has a tensor network structure with a large kernel. The norm of the final projected vector can be evaluated efficiently.

References

- Fernández, Y. N., Ritter, M. K., Jeannin, M., Li, J. W., Kloss, T., Louvet, T., ... & Waintal, X. (2024). Learning tensor networks with tensor cross interpolation: new algorithms and libraries. arXiv preprint arXiv:2407.02454.

- https://github.com/google/TensorNetwork (With few tutorials, references and video lectures)

- https://www.math3ma.com/blog/matrices-as-tensor-network-diagrams

- https://arxiv.org/abs/1306.2164

- https://arxiv.org/abs/1708.00006

- https://arxiv.org/abs/1603.03039

- https://doi.org/10.1137/090752286

- Wang, J., Roberts, C., Vidal, G., & Leichenauer, S. (2020). Anomaly detection with tensor networks. arXiv preprint arXiv:2006.02516.

5.2 Dequantisation

Quantum computation consists of three steps: 1) data encoding 2) quantum algorithm 3) measurement (readout of the result). When calculating the runtime of quantum algorithms, we implicitly assume that the data encoding step and the measurement step can be performed efficiently. However, this is a very strong assumption. In fact, only data with a special structure can be encoded exponentially fast, which is typically required to get the exponential speedup in many quantum algorithms. In particular, the algorithms relying on qBLAS. To complicate matters even further, the data encoding complexity highly depends on the exact architecture of the quantum device. Therefore, it seems reasonable to abstract away the data encoding and readout procedures when comparing quantum and classical computation. In quantum algorithms, we already assume efficient state preparation. Hence, we have to equip classical algorithms with an equally strong assumption and then check if we can improve their runtime.

The classical equivalent to the efficient quantum state preparation assumption is the sample and query access assumption.

Sample and query access We have a sample and query access to $x \in \mathcal{C}^N$, denoted by $SQ(x)$, iff we can query an index $i \in 1, 2, \ldots N$ for its value x_i and quary for $||x||_2$. With $SQ^\nu(x)$ we denote $SQ(x)$ with an access to an approximate norm $\bar{x} = (1 \pm \nu)||x||_2$. We get an intuitive understanding of the sample and query access assumption as follows. We want a fair comparison between the quantum and classical algorithms. Suppose we assume an efficient quantum state preparation procedure for the quantum algorithm. In that case, we have to give the classical algorithm access to measurements of the initial state in the computational basis (see Fig. 5.4). In other words, we start with a quantum state that we can access by measurements, but can only perform classical processing. In this way, the quantum and classical algorithms are given the same resources. Since the norm of the quantum state is always one, and we can determine efficiently (in time $\mathcal{O}(T)$, for some fixed T) the elements x_i, the efficient state preparation procedure implements the $SQ(x)$.

Dequantisation We say we dequantize a quantum protocol $\mathcal{S} : |\phi_1\rangle |\phi_2\rangle \ldots |\phi_N\rangle \rightarrow |\psi\rangle$ if we find a classical algorithm of the form $SQ(\phi_1, \phi_2, \ldots, \phi_N) \rightarrow SQ(\psi)$. In other words, given a sample and query access to the inputs, we have to efficiently prepare a sample and query access for the output of the desired calculation (see Fig. 5.4). The fact that Born-type measurements significantly speed up several machine learning protocols and randomised linear lagebra is a well known fact in numerical methods literature, however, a connection to quantum machine learning has been recognised only recently. The dequantised algorithms provide a clean framework for the abstraction of the preparation procedures and strong bounds on the exponential speedup of the quantum algorithms. Moreover, this framework can be a pillar of new types of classical-quantum algorithms, where the quantum part provides the sample and query access to the input of the classical algorithm.

Example 5.4. Nearest centroid classification The nearest centroid classification assigns a class based on the distance to the centroids of the training examples. To do that, we need to calculate the scalar product of the training examples of each class and the test example. Classically, this is done in $\mathcal{O}(N)$ steps, where N is the size of the input. In the following, we will first consider the quantum algorithm and then the dequantised classical algorithm with complexity $\mathcal{O}(T \frac{1}{\epsilon} \log \frac{1}{\delta})$.

The complexity of the nearest centroid algorithm is determined by the complexity of the calculation of the scalar product between two vectors $x, y \in \mathbb{C}^N$, i.e. $x \cdot y$.

Quantum algorithm In the quantum case, we assume that we can efficiently prepare the states $|x\rangle$ and
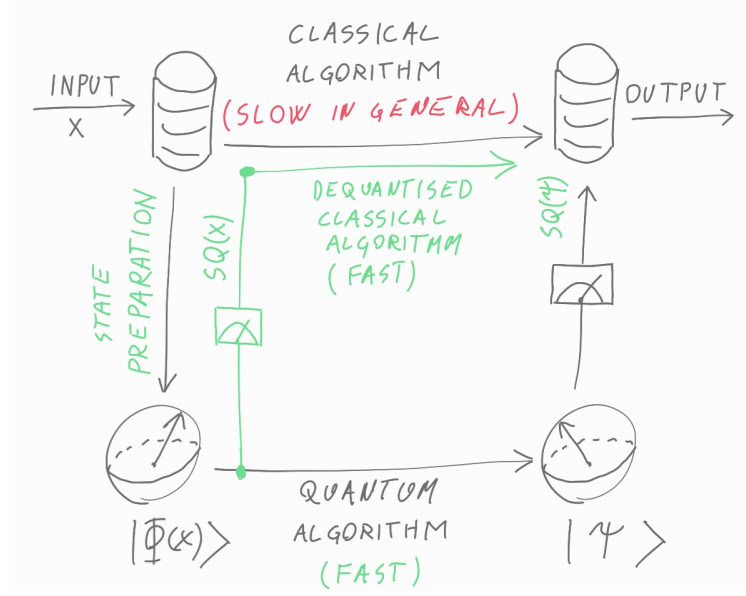
Figure 5.4: Intuitive representation of the dequantisation procedure and sample and query access assumption. We dequantise the quantum algorithm (marked with blue) if we find a polynomially equivalent classical algorithm having access to the initial quantum state (marked with green). This procedure is interesting only if a classical algorithm without the $SQ(x)$ assumption is slow. Slow in this context means exponentially slower in the size of the input compared to the quantum algorithm.

$|y\rangle$ corresponding to amplitude-encoded vectors x and y. The algorithm giving the overlap $\langle x|y\rangle$ is

$$. \tag{5.15}$$

The probability value that the auxiliary qubit is in the state $|0\rangle$ is $P(0) = \frac{1}{2}(1 + ||x\rangle\langle y||^2)$. The overlap can be calculated to precision ϵ and with probability $1 - \delta$ in time $\mathcal{O}(T\frac{1}{\epsilon^2}\log\frac{1}{\delta})$.

Dequantised classical algorithm The classical algorithm calculating the scalar product $x \cdot y$ given the sample and query access to x and query access to y is given below

1. $s = 54\frac{1}{\epsilon^2}\log\frac{2}{\delta}$

2. Collect measurements $i_1, \ldots i_s$ from $|x\rangle$; requires sample access

3. Calculate $z_j = x_{i_j} y_{i_j} \frac{||x||_2^2}{|x_{i_j}|}$; requires query access to x, $||x||_2$, and y

4. Separate z_j's into $6\log\frac{2}{\delta}$ buckets of size $\frac{9}{\epsilon^2}$ and take the mean of each bucket.

5. The output of the algorithm is the component-wise median of the means.

Scetch of the proof The number of samples s gives the time complexity and is trivially $\mathcal{O}(\frac{T}{\epsilon^2} \log \frac{1}{\delta})$, where T is the time necessary to determine x in the querry access (similar to determining one expectation value in the quantum algorithm). We now have to bound the error and success probability.

First we note that z_j is a random variable with mean $x \cdot y$ and bounded variance $\text{var}(z_j) \leq ||x||_2^2 ||y||_2^2$. The variance of the mean of the copies in one bucket is hence bounded by $\frac{\epsilon^2}{9} ||x||_2^2 ||y||_2^2$. Using the Chebishev inequality, we observe that the probability that the mean is $\frac{\epsilon}{\sqrt{2}} ||x||_2 ||x||_2$ away from the desired result is bounded from above by $\frac{2}{9}$. Finally, we use the Chernoff-Hoeffding inequality to show that the probability that the median of the means of the buckets is with probability $1-\delta$ inside the desired result $x \cdot y \pm \epsilon ||x||_2 ||y||_2$.

The quantum case can be sped up quadratically by using the amplitude amplification algorithm with the final runtime complexity $\mathcal{O}(\frac{T}{\epsilon} \log \frac{1}{\delta})$. Nevertheless, the discussed dequantised classical algorithm is polynomially equivalent to the best quantum algorithm.

References

- Tang, E. (2023). Quantum machine learning without any quantum. University of Washington.