Quantum feature space learning: characterisation and possible advantages

Thesis

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE
in
THEORETICAL PHYSICS

Author :	Dyon van Vreumingen
Student id :	s1348434
Supervisor :	Dr. V. Dunjko
In collaboration with :	C. Gyurik MSc.
Second corrector :	Dr. Thomas o'Brien

Leiden, The Netherlands, October 27, 2020

Quantum feature space learning: characterisation and possible advantages

Dyon van Vreumingen

Huygens-Kamerlingh Onnes laboratory, Leiden university
P.O. Box 9500, 2300 RA Leiden, The Netherlands

October 27, 2020

ABSTRACT

Quantum machine learning is currently regarded as one of the most promising candidates for solving problems that appear out of reach using classical computers. Recently, a novel subfield of quantum learning was opened up by Havlíček et al. [1], who proposed a quantum learning algorithm which is closely related to support vector machines, yet which can be implemented on currently available quantum hardware. In this thesis, contribute to quantum machine learning by presenting new results on the capabilities of this algorithm, placing it in the perspectives of classical learning theory and quantum complexity. As the follow-up research which has since been published mainly focusses on details of experimental implementation, results in this direction are still lacking. Specifically, we compare the hyperplane (explicit) and kernel (implicit) formulations of the classifier algorithm, study its generalisation performance in the framework of statistical learning theory, and pin down the precise requirements for a quantum advantage using this algorithm. To this end, we apply the so-called representer theorem, known from the study of kernel methods in machine learning, to show training set optimality of the implicit formulation under regularised error measures. Furthermore, we show a tight upper bound on the fat shattering dimension of this type of quantum classifier, and discuss the implications for generalisation performance. Lastly, we carry out a complexity theoretic study showing that classical intractability of evaluating quantum kernels implies also the intractability of these quantum classifiers. We argue that despite this fact, we cannot claim that there exist problems which are hard to learn classically, but not quantumly, in the PAC learning sense, and subsequently describe the complexity theoretic requirements of quantum CLF learning to achieve quantum learning supremacy.

Keywords Quantum machine learning, support vector machine, complexity theory, computational learning theory

Contents

Chapter 1

Introduction

When a few decades ago it was discovered that a computer reliant on quantum mechanics had the potential to solve problems out of reach for classical computers [2, 3], it opened up a new, large research field now known as quantum computing. Further interest in this field was sparked by the discovery of quantum algorithms such as Grover's [4] and Shor's [5], which promised quadratic and even exponential speedups in fundamental problems like unstructured search and prime factorisation. Ever since, much effort has been put into the identification of the capabilities of quantum computers, which led to the theory of quantum complexity [6], as well as the understanding and harnessing of the sources of quantum speedup [7]. A specialisation in this area is quantum machine learning [8], which seeks to exploit properties of quantum computing to either accelerate classical machine learning, or improve its learning performance. Since the proposal of an exponential quantum speedup in matrix inversion [9], several quantum machine learning algorithms have been put forward [10, 11], which capitalise on this result to show possibilities for speedups in machine learning. The problem with such algorithms, however, is that they are not suitable to run on the quantum hardware that is available today. Currently available quantum hardware falls into the *noisy intermediate-scale quantum* (NISQ) regime: that is, chips with few (on the order 10^3) functional qubits, limited interaction between qubits, and noise caused by quantum decoherence and gate errors [12, 13]. Because of this, most of the quantum machine learning research in recent years has focussed on NISQ compatible algorithms, employing low-qubit, short-depth circuits [14] without the need to compute quantum states to delicate precision. One such work is that by Havlíček et al. [1], which introduces a NISQ supervised classification algorithm that is shown to be closely related to support vector machines [15]. Because of this relationship, the algorithm may be

regarded as a quantum relative to classical SVMs, which allows one to apply the extensive theory of SVMs and hyperplane learning [16] to understand the potential benefits of this learning model. This is indeed the objective of our research: we consider classical learning theory in the context of hyperplane learning, and build upon the work of Havlíček et al. using the theory to scrutinise the properties and capabilities of their quantum learning model under different learning circumstances. Lastly, we consider possible paths for the quantum learning model to distinguish itself from classical models, in terms of quantum advantage.

Our work is structured as follows. Chapter 2 provides a description of quantum mechanics and shows how it gives rise to quantum computing. In chapter 3, we briefly discuss the concept of supervised machine learning, and subsequently introduce the quantum machine learning model from the work of Havlíček et al. [1]. In chapter 4 we formalise SVM-like supervised learning, by using elements of functional analysis as the mathematical foundation of the learning model to define the classifiers to be considered in this work. In addition, we discuss theory of generalisation performance applied to these classifiers. Next, we consider the quantum version of this learning model in chapter 5, relating to the previous chapters in its definition, and discuss the properties and implications of the hyperplane and kernel representations of the model. This chapter also extends concepts of generalisation performance to these quantum classifiers. Then, in chapter 6, we discuss quantum complexity theory, and use this to explore paths towards quantum advantage of the quantum learning model, clarifying statements made by Havlíček et al. with regards to such advantage. This chapter also highlights the distinction between evaluating classifiers and solving learning problems, and discusses the requirements for quantum learning supremacy, which turn out to be stronger than the mere classical infeasibility of evaluating quantum classifiers. Finally, we present our conclusions in chapter 7.

Chapter 2

Quantum computing

In this chapter, we give a concise description of quantum computing which will enable us to discuss supervised quantum machine learning in the following chapters. Through a brief discussion of quantum mechanics providing the definition of quantum states, manipulation of these states and measurement (section 2.1), we establish a quantum computational model, highlighting aspects that we require for the discussion of quantum feature space learning (section 2.2)..

2.1 Quantum states and measurement

In classical mechanics, systems are usually described in terms of dynamically changing variables such as position, velocity, angular momentum and the like. Quantum mechanical systems, however, are different: their properties are described by what we call a *wave function*, or simply a *state*. A quantum state is a function $\Psi(x,t)$ that is dependent on space and time, and satisfies the Schrödinger equation:

$$i\hbar\frac{\partial}{\partial t}\Psi(x,t) = \left[-\frac{1}{2m}\frac{\partial^2}{\partial x^2} + V(x,t)\right]\Psi(x,t), \tag{2.1}$$

with units chosen so that $\hbar = 1$. Assuming the potential V is independent of time, this equation can be split into a time-dependent part,

$$i\hbar\frac{\partial\phi(t)}{\partial t} = E\phi(t), \tag{2.2}$$

and a space-dependent part

$$-\frac{1}{2m}\frac{\partial^2\psi(x)}{\partial x^2} + V(x)\psi(x) = E\psi(x), \tag{2.3}$$

where the constant E is an energy [17]. The complete wave function is then given by

$$\Psi(x,t) = \phi(t)\psi(x). \tag{2.4}$$

From eq. 2.3, it is apparent that at any point in time, the solution space – that is, the space of allowed wave functions – is a Hilbert space. After all, the operator $\mathrm{H} := (-1/2m)\partial^2/\partial x^2 + V(x)$, which can be identified as the hamiltonian of the system, is a linear map on the set of wave functions; hence, eq. 2.3 is an eigenvalue equation, whose solutions may be expressed through the eigenbasis of H. To indicate that ψ is an element of a (complex) Hilbert space and therefore a vector itself, we write it as a *ket* $|\psi\rangle$. Its conjugate transpose is the *bra* $\langle\psi|$. For reasons related to measurements (which we discuss shortly), we must restrict ourselves to states of unit norm:

$$\langle\psi|\psi\rangle = 1. \tag{2.5}$$

The time-dependent part, on the other hand, has solution

$$\phi(t) = e^{-iEt}\phi(0) \tag{2.6}$$

for some constant $\phi(0)$, which yields the complete wave function solution

$$|\Psi(x,t)\rangle = e^{-iEt}|\psi(x)\rangle \tag{2.7}$$

where we absorbed $\phi(0)$ into $|\psi(x)\rangle$. Since this equality holds for any eigenstate $|\psi\rangle$ of H and corresponding eigenenergy E, we may succinctly write it as

$$|\Psi(x,t)\rangle = e^{-i\mathrm{H}t}|\psi(x)\rangle. \tag{2.8}$$

One can show that any hamiltonian H is necessarily hermitian, and therefore $e^{-i\mathrm{H}t}$ is unitary. We have thus arrived at the *time evolution principle*, which asserts that the quantum states of the same system at two different points in time are related through a unitary transformation:

$$|\Psi(x,t')\rangle = \mathrm{U}(t'-t)|\Psi(x,t)\rangle. \tag{2.9}$$

Note that unitary operators preserve the norm of the wave function.

Later on, we will see that it will be convenient to express a quantum state as a *density matrix*:

$$\rho = |\psi\rangle\langle\psi|. \tag{2.10}$$

We see that ρ is a positive-semidefinite hermitian operator, and has unit trace (since $\operatorname{tr}\rho = \langle\psi|\psi\rangle = 1$. It has eigenvalues 1 (for eigenstate ψ) and 0 (all states orthogonal to ψ). In other words, ρ is a rank-one projection, and therefore idempotent: $\rho^2 = \rho$. Density matrices of this form are called *pure states*, and have a one-to-one correspondence to a state vector $|\psi\rangle$ up to global phase. On the other hand, an ensemble of states $\{(\rho_i, p_i)\}$ whose probabilities p_i add up to 1 is called a *mixed state*, and its density matrix equals

$$\rho = \sum_i p_i \rho_i. \tag{2.11}$$

Any mixed state is still positive-semidefinite, hermitian and has trace 1, however is no longer generally idempotent, since

$$\operatorname{tr}\rho^2 = \sum_i p_i^2 < 1 \tag{2.12}$$

if ρ is not a pure state.

Next, let us review how multiple quantum states are joined together. Let $\mathcal{H}$ and $\mathcal{K}$ be Hilbert spaces; then the joint space is formed through the Kronecker product or *tensor product* $\mathcal{H} \otimes \mathcal{K}$. If $\{|i\rangle\}$ and $\{|j\rangle\}$ are bases of $\mathcal{H}$ and $\mathcal{K}$ respectively, then states in $\mathcal{H} \otimes \mathcal{K}$ are expressed in the joint basis $\{|i\rangle \otimes |j\rangle\}$ of $\mathcal{H} \otimes \mathcal{K}$:

$$|\psi\rangle = \sum_{ij} \psi_{ij} |i\rangle \otimes |j\rangle. \tag{2.13}$$

For convenience, we usually write $|\psi\rangle|\phi\rangle$ in place of $|\psi\rangle \otimes |\phi\rangle$. The tensor product has the useful property that any joint operator is evaluated separately on each subspace:

$$(\mathrm{U} \otimes \mathrm{V})(|\psi\rangle \otimes |\phi\rangle) = \mathrm{U}|\psi\rangle \otimes \mathrm{V}|\phi\rangle. \tag{2.14}$$

In quantum mechanics, this construction of composite systems gives rise to the curious phenomenon of *entanglement*. Consider two quantum systems in Hilbert spaces $\mathcal{H}$ and $\mathcal{K}$ with eigenbases $\{|\psi_1\rangle, |\psi_2\rangle\}$ and $\{|\phi_1\rangle, |\phi_2\rangle\}$ respectively. If now, for instance, the joint state of the two systems is $|\psi_1\rangle|\phi_1\rangle$, we can clearly separate the state claiming that the first system is in state $|\psi_1\rangle$ and the second is in state $|\phi_1\rangle$. Such a state is called separable or disentangled. However, the joint state

$$|\Psi\rangle = \frac{1}{\sqrt{2}}(|\psi_1\rangle|\phi_1\rangle + |\psi_2\rangle|\phi_2\rangle) \tag{2.15}$$

is also a valid quantum state. For this state, there exists no single tensor product of individual states (i.e. states in either $\mathcal{H}$ or $\mathcal{K}$) that is equal to Ψ; hence Ψ is called inseparable or entangled. As such, we cannot say that either of the two systems is in a certain state; we can only reason about the joint system being in a joint state. In essence, the two systems have become one through entanglement.

Lastly, we must know how to extract knowledge from quantum states, or more simply put, how to *measure* them. There are a few different interpretations on how measurements occur in quantum mechanics, and we will follow the Copenhagen interpretation, which is the most commonly followed interpretation of quantum mechanics.

Formally, a quantum state is measured through hermitian operators. That is, when a quantum state $|\psi\rangle$ interacts with a measurement device (we shall skip any philosophical discussion on what this precisely means), this device will always measure the state in the eigenbasis of some operator O, and return an eigenvalue of this operator. More precisely, measurement of a generic state, expressed as a superposition over an eigenbasis,

$$|\psi\rangle = \sum_k a_k |\lambda_k\rangle \tag{2.16}$$

returns eigenvalue λ_k with probability $|a_k|^2$. Note that since O is hermitian, its eigenvalues – and therefore all possible measurement outcomes – are real, which is precisely what one would expect to obtain from a measurement. Furthermore, note that

$$\langle\psi|\psi\rangle = \sum_k |a_k|^2 = 1, \tag{2.17}$$

which motivates restricting the norm of quantum states to unity, as mentioned earlier – after all, probabilities must always sum to one.

However, this is not the entire story. Curiously, the wave function itself is also influenced by the measurement, in contrast to classical mechanics. According to the Copenhagen interpretation, the wavefunction upon measurement collapses to the eigenstate corresponding to the eigenvalue that was found. In other words, besides producing an eigenvalue outcome, a measurement applies a projector to the state measured,

$$|\psi\rangle \mapsto \frac{|\lambda_k\rangle\langle\lambda_k|\psi\rangle}{|\langle\lambda_k|\psi\rangle|} \tag{2.18}$$

with probability $|a_k|^2$. Note the normalisation factor $|\langle\lambda_k|\psi\rangle|$ appearing in the denominator to ensure that the result is still a state of unit norm. But

since $\langle\lambda_k|\psi\rangle = a_k$, we have that the probability of obtaining outcome λ_k after measurement is given by the overlap between $|\lambda_k\rangle$ and $|\psi\rangle$:

$$\mathbb{P}(\lambda_k \,|\, |\psi\rangle) = |\langle\lambda_k|\psi\rangle|^2. \tag{2.19}$$

What's even more curious is that this behaviour also occurs with entangled states. Consider the entangled state in eq. 2.15: if we measure the first system through the normalised projector $\sqrt{2}\,|\psi_1\rangle\langle\psi_1|$, the resulting state after measurement is $|\Psi\rangle = |\psi_1\rangle|\phi_1\rangle$. Apparently, measurement of the first system also influences the second system!

With the probability distribution given by the overlaps $|\langle\lambda_k|\psi\rangle|^2$, one can compute the expectation value of an observable under a state $|\psi\rangle$ (that is, the mean of eigenvalue outcomes expected after repeated measurements of the same state):

$$\begin{aligned}\mathbb{E}(\mathrm{O} \,|\, |\psi\rangle) &= \sum_k \lambda_k \,\mathbb{P}(\lambda_k \,|\, |\psi\rangle) \\ &= \sum_k \lambda_k \,|\langle\lambda_k|\psi\rangle|^2 = \sum_k \lambda_k \,\langle\psi|\lambda_k\rangle\langle\lambda_k|\psi\rangle \\ &= \langle\psi|\mathrm{O}|\psi\rangle,\end{aligned} \tag{2.20}$$

by the eigendecomposition $\mathrm{O} = \sum_k \lambda_k |\lambda_k\rangle\langle\lambda_k|$.

If instead we are working with density matrices, the probability of λ_k is written

$$\begin{aligned}\mathbb{P}(\lambda_k \,|\, \rho) &= \sum_i p_i \,\mathbb{P}(\lambda_k \,|\, \rho_i) = \sum_i p_i \,\mathbb{P}(\lambda_k \,|\, |\psi_i\rangle\langle\psi_i|) \\ &= \sum_i p_i \langle\lambda_k|\psi_i\rangle\langle\psi_i|\lambda_k\rangle = \sum_i p_i \,\mathrm{tr}[|\lambda_k\rangle\langle\lambda_k|\rho_i] \\ &= \mathrm{tr}[|\lambda_k\rangle\langle\lambda_k|\rho],\end{aligned} \tag{2.21}$$

and thus the expectation value reads

$$\begin{aligned}\mathbb{E}(\mathrm{O} \,|\, \rho) &= \sum_k \lambda_k \,\mathbb{P}(\lambda_k \,|\, \rho) = \sum_k \lambda_k \,\mathrm{tr}[|\lambda_k\rangle\langle\lambda_k|\rho] \\ &= \mathrm{tr}[\mathrm{O}\rho].\end{aligned} \tag{2.22}$$

With the necessary quantum machinery in place, we shall now consider how this gives rise to a quantum computational model.

2.2 Quantum computation

In classical computation, the fundamental unit of information is a *bit*, which can take on the value 0 or 1. Bit strings are formed by joining multiple bits together, and such bitstrings can be manipulated to implement logical operations; a sequence of such manipulations is what we call an algorithm.

In quantum computing, the fundamental unit of information is a quantum bit, or *qubit* for short: a two-level system whose eigenbasis is written $\{|0\rangle, |1\rangle\}$. This choice of basis is called the *computational basis*, and is usually interpreted as the eigenbasis of the Pauli-Z matrix

$$\mathrm{Z} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{2.23}$$

where $\mathrm{Z}|0\rangle = |0\rangle$ and $\mathrm{Z}|1\rangle = -|1\rangle$. This representation of a qubit immediately unveils a distinction between classical bits and quantum bits: a qubit can be in a superposition of zero and one, since any state

$$|\psi\rangle = a_0|0\rangle + a_1|1\rangle \tag{2.24}$$

with $|a_0|^2 + |a_1|^2 = 1$ is a valid quantum state.

Multiple qubit systems can be joined together through the tensor product as discussed above, to form qubit strings. For such qubit strings, we shall write the entire string as a single ket: for example, $|000\rangle = |0\rangle|0\rangle|0\rangle$. Now for a system consisting of n qubits, the generic joint state shall be written

$$|\psi\rangle = \sum_{z\in\{0,1\}^n} a_z|z\rangle \tag{2.25}$$

where $|z\rangle$ stands for the qubit string corresponding to the bit string z. Note that the dimension of n-qubit Hilbert space grows exponentially in n.

To manipulate qubit string states, we make use of the unitary evolution principle of quantum mechanics: namely that two states are separated in time by a unitary operator. From a computational perspective, this means that any manipulations on qubit strings must be carried out using unitary operators. In analogy to to logic gates on classical bit strings, we call these manipulations *unitary gates*. Similarly, a sequence of unitary gates acting on a set of qubits is a *quantum circuit*.

Frequently occuring single-qubit gates, which take as input a single qubit and outputs a single qubit, are the Pauli gates

$$\mathrm{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \mathrm{Y} = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad \mathrm{Z} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \tag{2.26}$$

and the Hamamard gate

$$\mathrm{H} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{2.27}$$

(not to be confused with the hamiltonian H we introduced earlier) which has the interesting property that

$$\mathrm{X} = \mathrm{HZH}. \tag{2.28}$$

Furthermore, the framework of quantum computing allows for continuous extensions of these gates. In particular, note that the Pauli operators are hermitian; thus the complex exponentials of these operators are valid unitary gates[1]:

$$\mathrm{R_X}(\theta) := e^{-i\theta/2\,\mathrm{X}} = \cos\frac{\theta}{2}\,\mathrm{I} - i\sin\frac{\theta}{2}\,\mathrm{X} = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}, \tag{2.29}$$

$$\mathrm{R_Y}(\theta) := e^{-i\theta/2\,\mathrm{Y}} = \cos\frac{\theta}{2}\,\mathrm{I} - i\sin\frac{\theta}{2}\,\mathrm{Y} = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}, \tag{2.30}$$

$$\mathrm{R_Z}(\theta) := e^{-i\theta/2\,\mathrm{Z}} = \cos\frac{\theta}{2}\,\mathrm{I} - i\sin\frac{\theta}{2}\,\mathrm{Z} = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}, \tag{2.31}$$

for $\theta \in [0, 4\pi]$. However, since $\mathrm{R}_\Sigma(2\pi) = -\mathrm{I}$ with $\Sigma \in \{\mathrm{X}, \mathrm{Y}, \mathrm{Z}\}$, and a global sign is unobservable (since measurement probabilities are invariant under a global phase change), we can restrict ourselves to $\theta \in [0, 2\pi]$. These three operators, which are called the *Pauli rotation matrices*, are so important because they generate the group $SU(2)$ of unitary rotations in $\mathbb{C}^2$. That is, every single-qubit unitary, which is an element of $SU(2)$ up to global phase, can be expressed as a product of Pauli rotation gates:

$$\mathrm{R}_{\boldsymbol{\theta}} = \exp[-i\,(\theta_1\mathrm{X} + \theta_2\mathrm{Y} + \theta_3\mathrm{Z})]. \tag{2.32}$$

Now, since these gates act on single qubits, they cannot create entanglement; after all, if $|\psi\rangle|\phi\rangle$ is a disentangled state, and is acted upon by single-qubit unitaries $\mathrm{U} \otimes \mathrm{V}$, then the resulting state $\mathrm{U}|\psi\rangle \otimes \mathrm{V}|\phi\rangle$ is still disentangled. Therefore we also require multiple-qubit gates. The most commonly appearing of these are the controlled-X, also called CNOT, and controlled-Z gates, which are two-qubit unitaries:

$$\mathrm{C_X} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}; \qquad \mathrm{C_Z} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}. \tag{2.33}$$

[1] In fact, one may regard a rotation gate $\mathrm{R}_\Sigma(\theta)$, for $\Sigma \in \{\mathrm{X}, \mathrm{Y}, \mathrm{Z}\}$, as a time-evolved unitary under the hamiltonian Σ and time $\theta/2$.

These gates can be interpreted as follows: given two qubits, if the first is in state $|1\rangle$, then a X or Z gate respectively is applied to the second qubit. And since we work in a Hilbert space, the gates act linearly on superposition states. For example:

$$\begin{aligned} \mathrm{C_X}\left(\frac{1}{\sqrt{2}}\big(|0\rangle + |1\rangle\big)\,|1\rangle\right) &= \frac{1}{\sqrt{2}}\big(\mathrm{C_X}|01\rangle + \mathrm{C_X}|11\rangle\big) \\ &= \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle). \end{aligned} \tag{2.34}$$

Notice how the initial state $1/\sqrt{2}(|0\rangle+|1\rangle)\otimes|1\rangle$ is separable, but the resulting state after applying $\mathrm{C_X}$ is not. From this example we see that multi-qubit unitaries can create entanglement, whereas single-qubit unitaries cannot, which necessitates the use of the former. In fact, the set of single-qubit unitaries together with the $\mathrm{C_X}$ gate is known to be universal [18] in that any quantum circuit can be expressed using finitely many such gates. However, the complete decomposition may require a number of gates that is exponential in the number of qubits [19].

Clearly, after all gates in a quantum circuit have been applied, one will want to obtain an output from the ciruit by measuring the resulting state. Typically, the state is measured in the computational basis; thus the measurement projectors are $|z\rangle\langle z|$ for $z \in \{0,1\}^n$. Since one may be more interested in (the bit string corresponding to) the output eigenstate rather than its eigenvalue – values which are free to pick by choice of the observable –, we will regard a measurement outcome as either an eigenstate or an eigenvalue depending on context.

Chapter 3

Quantum machine learning

In this chapter, we give a brief discussion on supervised machine learning, by describing what is meant by classification, loss functions, generalisation and data representation. With these notions, we can interpret the quantum supervised learning algorithm proposed in the work of Havlíček et al., which we subsequently introduce. In doing so, we give a summary of their paper, including a precise description of the learning algorithm, and setting the stage for the extension of their work we present in the following chapters.

3.1 Supervised learning

The goal of machine learning is the construction of algorithms which deduce patterns in given data and build a model that approximately describes the data in terms of these patterns, in such a way that this model can generalise well to new, unseen data. That is, the model should roughly capture the structure that new data will have, based on patterns in old data, so that it can make predictions on new data within a reasonable margin of accuracy. Machine learning is typically divided into two branches: unsupervised and supervised learning (a third branch often mentioned is reinforcement learning, which we will not consider in this work). Unsupervised learning deals with the task of finding structure in data about which little or nothing is otherwise known. An example is clustering, where the learner must find a partition of a set of points in a (possibly high-dimensional) vector space which groups the points by proximity, according to some distance measure. With supervised learning, on the other hand, all (or most) data points are supplied with additional information. The task is then, given a data point as input, to output a value of such additional information which corresponds to the input point.

More concretely, the model one seeks to find is a function that maps input data points to the desired output, as inferred from the given input-output pairs. The most commonly practiced method of supervised learning is *classification*, where input data is partitioned into classes, and the algorithm must learn to assign the correct class label to as many data instances as possible. This could be a separation into two classes (binary classification) or more. In order to learn the function, the input data from which the algorithm must be accompanied with the correct class label for each instance. Hence, the input data is usually called the training set of training data; the labels then form the additional information for classification methods.

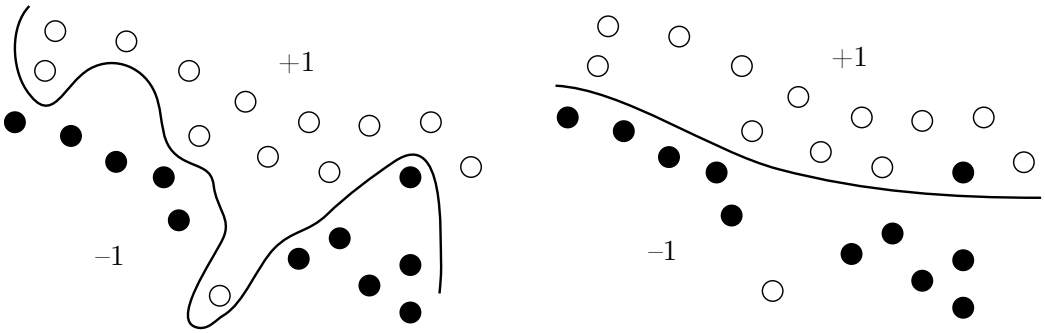
To clarify what precisely training means, let us describe classification in a more mathematical fashion. Let $\mathcal{X}$ be the set from which the input data is drawn, let $\mathcal{Y}$ be a label set, and say there is some ground truth mapping from $\mathcal{X}$ to $\mathcal{Y}$ that determines the correct label $y \in \mathcal{Y}$ for every $x \in \mathcal{X}$. The objective of a learning algorithm is now to find a hypothesis function $c : \mathcal{X} \to \mathcal{Y}$ from a given set of hypothesis functions (the *hypothesis family*) such that, ideally, the output $c(x)$ matches the correct label $y \in \mathcal{Y}$ for any $x \in \mathcal{X}$. To this end, a training set $\mathcal{T} \subseteq \mathcal{X} \times \mathcal{Y}$ containing correct input-output pairs is supplied; this is the only information available to the learning algorithm for inferring a good function. A good hypothesis then minimises some measure of error on this training set, which implies that at least on the provided data, it classifies accurately. A straightforward example of such an error measure is the number of misclassified instances:

$$E_{\mathcal{T}}[c(\cdot)] = \sum_{(x,y)\in\mathcal{T}} \mathbb{1}_{c(x)\neq y}, \tag{3.1}$$

where $\mathbb{1}_\pi$ is the indicator function, which maps a proposition π to 1 if it is true, and to 0 if it is false. In the case of binary classification with labels $\mathcal{Y} = \{+1, -1\}$, eq. 3.1 can be simplified to

$$E_{\mathcal{T}}[c(\cdot)] = \frac{1}{2} \sum_{(x,y)\in\mathcal{T}} |c(x) - y|. \tag{3.2}$$

An often observed phenomenon in machine learning is *overfitting*, where a learned hypothesis captures the training data too well. That is, it classifies (almost) all points in the training set correctly, but thereby mimics the training data to such an extent that it fails to capture the general structure in the data, which impedes generalisation to future data. See figure 3.1. Since more complex models are more likely to overfit, it is common practice in supervised learning to augment the error measure with a *regularisation term*, which depends only on one or more properties of $c(\cdot)$ itself that are independent of



(a) (b)

Figure 3.1. *(a) An overfitted model. The classification boundary is too complex and places too much emphasis on outliers in the training set. It is more likely to misclassify future data points. (b) A simpler, less overfitting model. Despite misclassification of some training set outliers, the overall structure of the data is better captured, thus the model will likely generalise better to new data.*

the data. If the regularisation term is chosen such that it becomes small for simple models, minimisation of the augmented error measure ensures that the chosen model both classifies the training data well and simultaneously does not heavily overfit. The general term for an error measure which may be augmented with a regularisation term is a *loss function*. We will discuss generalisation performance in more detail in section 4.3.

Usually, hypothesis families are *parametrised*, i.e. the functions in the family are expressed in terms of a set of continuous parameters $\boldsymbol{\theta}$. Training such a model then translates to optimising these parameters to achieve a minimal loss value. In case this loss value is differentiable in $\boldsymbol{\theta}$[1], an optimal solution, i.e. a solution that minimises the loss value, typically satisfies the condition

$$\frac{\partial}{\partial \boldsymbol{\theta}} L_{\mathcal{T}}[c_{\boldsymbol{\theta}}(\cdot)] = \mathbf{0}. \tag{3.3}$$

where $L_{\mathcal{T}}$ is the loss value on the training set $\mathcal{T}$. Indeed, many training algorithms make use, in either basic or more sophisticated ways, of such gradient computations [20]. However, often there exist multiple configurations of $\boldsymbol{\theta}$ which satisfy eq. 3.3, which do not minimise $L_{\mathcal{T}}$; such configurations are called *local minima*, whereas the optimal solution is called the *global minimum*.

[1]Note that the error in eq. 3.2 is not differentiable since $\mathcal{Y}$ is not continuous and hence $c(x)$ is not a continuous function. However, a common method, which we will encounter shortly, is to define $c_{\boldsymbol{\theta}}(x)$ as a thresholded value of an underlying continuous function $f_{\boldsymbol{\theta}}(x)$; an error measure incorporating $f_{\boldsymbol{\theta}}(x)$ is typically differentiable in this case.

Before we continue to describe the type of classification model we study in this research, it is worthwhile to note that many learning algorithms, in order to produce an accurate predictive model, build an internal representation of the data at hand. This representation can be regarded as a collection of extracted features that characterise the data; for instance, when distinguishing images of handwritten digits 0 and 1, the presence of a hole in the middle may function as a feature that is characteristic of the digit 0. Typically, the representation is a map from the original data space to some representation space with a higher (or lower) dimension, finding a useful representation may be included in the learning procedure. For example, neural networks map their input into a number of neuron layers before making a decision on which label to choose [21]. The principle of representation building applies to a collection of other learning procedures, including support vector machines (SVMs) [15], principal component analysis [22] (which may be used in either a supervised or unsupervised fashion) and dictionary learning [23]. We will see that such representations come naturally to the method of quantum machine learning we study in this work.

3.2 Variational quantum circuit learning

At the current moment, a ubiquitous method to implement machine learning on quantum computers is *variational quantum circuit learning*. This method revolves around the use of parametrised quantum gates, such as the parametrised Pauli gates described in section 2.2. Quantum circuits consisting of such parametrised gates are called parametrised quantum circuits, or variational circuits in reference to the variational quantum eigensolver [24], which was one of the first manifestations of variational learning. The main idea of variational learning is that one may define a loss value dependent on some outcome probability or expectation value of a quantum circuit setup; and since the circuit is parametrised, so is the loss value, which can thus be minimised accordingly. For example, the variational eigensolver seeks to find the ground state of a hamiltonian H by optimising over a set of parametrised quantum states $|\psi(\boldsymbol{\theta})\rangle$. Since these parametrised states may be prepared by applying a parametrised circuit $\mathrm{U}(\boldsymbol{\theta})$ to the all-zero state $|0\rangle$, the loss value, which is the expected energy $\langle \mathrm{H} \rangle$, may be expressed as

$$L(\boldsymbol{\theta}) = \langle 0|\mathrm{U}^\dagger(\boldsymbol{\theta})\mathrm{H}\mathrm{U}(\boldsymbol{\theta})|0\rangle. \tag{3.4}$$

Then, by minimising $L(\boldsymbol{\theta})$ over the parameters $\boldsymbol{\theta}$, one obtains a state $|\psi^*\rangle = \mathrm{U}(\boldsymbol{\theta}^*)|0\rangle$ which is close to the ground state of H.

Variational learning extends well beyond finding ground states, and is an instinctive basis for quantum machine learning, for two reasons. First of all, it is similar to many classical machine learning algorithms, e.g. neural networks, which can also be regarded as a form of parametrised circuits; therefore one can use a large body of prior knowledge about classical machine learning to describe variational learning. Secondly, variational learning is applicable to small circuits, which makes it very suitable for implementation on NISQ processors.

One approach to variational learning, which is the main work that our research builds on, is that of Havlíček et al. [1]. In their paper, they describe a parametrised quantum learning approach that is similar to SVMs, for the task of binary classification. Here, the representation of a data point $\mathbf{x} \in \mathbb{R}^m$ is a quantum state $|\Phi(\mathbf{x})\rangle$; the operation which maps a data point to such a state is called the *feature map*. This feature map is a parametrised circuit $\mathrm{U}_\Phi(\mathbf{x})$, applied to the state $|0\rangle$, whose parameters are dependent on the entries of $\mathbf{x}$. More precisely, the authors define their feature map circuit as follows:

$$\mathrm{U}_\Phi(\mathbf{x}) = \mathrm{V}_\Phi(\mathbf{x})\mathrm{H}^{\otimes n}\mathrm{V}_\Phi(\mathbf{x})\mathrm{H}^{\otimes n} \tag{3.5}$$

where n is the number of qubits, H is the Hadamard gate, and

$$\mathrm{V}_\Phi(\mathbf{x}) = \exp\left(i \sum_{S\subseteq[n]} \phi_S(\mathbf{x}) \prod_{i\in S} \mathrm{Z}_i\right). \tag{3.6}$$

with Z_i being the Pauli Z gate applied to the i-th qubit, and $\phi_S(\mathbf{x}) \in \mathbb{R}$ a coefficient (possibly continuously) depending on $\mathbf{x}$. This setup was chosen specifically with NISQ processing capabilities in mind: the preparation circuit U_Φ is a shallow circuit (since many of the Z gates can be applied in parallel), and in their experiment, the authors choose the sets S in eq. 3.6 such that the qubit interactions are sparse ($|S| \leq 2$) and short-range.

Classification of each data point is carried out by measuring the expectation value of a variational observable $\mathrm{F}(\boldsymbol{\theta}) = \mathrm{W}^\dagger(\boldsymbol{\theta})\mathrm{D}\mathrm{W}(\boldsymbol{\theta})$ in the state $|\Phi(\mathbf{x})\rangle$, where $\mathrm{W}(\boldsymbol{\theta})$ is a parametrised circuit, and D is a fixed diagonal observable relative to the Z basis. If the estimated expectation value, after a number of repeated measurements, is larger than some threshold d, the predicted label given by the classifier is $+1$; otherwise, it is -1. That is, the output of the quantum process is a random variable $\tilde{c}(\mathbf{x})$ which estimates the

classification value[2]

$$c(\mathbf{x}) = \operatorname{sgn}\big(\langle 0|\mathrm{U}_\Phi^\dagger(\mathbf{x})\mathrm{W}^\dagger(\boldsymbol{\theta})\mathrm{D}\mathrm{W}(\boldsymbol{\theta})\mathrm{U}_\Phi(\mathbf{x})|0\rangle - d\big). \tag{3.7}$$

In order to train the circuit, the authors define the loss value

$$L_{\mathcal{T}}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{T}|}\sum_{(\mathbf{x},y)\in\mathcal{T}} \mathbb{P}(\tilde{c}(\mathbf{x}) \neq y), \tag{3.8}$$

which is minimised using a gradient descent algorithm (note that the probability values are continuous in $\boldsymbol{\theta}$).

At first sight, the connection between this variational learning algorithm and SVMs seems far away. Nonetheless, the authors show a remarkable resemblance between the two methods. Indeed, if we write $\rho_\Phi(\mathbf{x}) = |\Phi(\mathbf{x})\rangle\langle\Phi(\mathbf{x})|$, we notice that

$$\langle\Phi(\mathbf{x})|\mathrm{F}(\boldsymbol{\theta})|\Phi(\mathbf{x})\rangle = \operatorname{tr}[\mathrm{F}(\boldsymbol{\theta})\rho_\Phi(\mathbf{x})]. \tag{3.9}$$

Let us analyse this expression a bit more. First, both $\mathrm{F}(\boldsymbol{\theta})$ and $\rho_\Phi(\mathbf{x})$ are hermitian for any $\boldsymbol{\theta}$ and $\mathbf{x}$; furthermore, the set $H(2^n)$ of $2^n \times 2^n$ complex hermitian matrices is a real vector space. Indeed, there exist bases, such as the normalised Pauli basis $\mathcal{P} = 2^{-n/2}\{\mathrm{I},\mathrm{X},\mathrm{Y},\mathrm{Z}\}^{\otimes n}$ such that every element of $H(2^n)$ can be written as a linear combination of these basis elements. Note that $H(2^n)$ is a 4^n-dimensional space. The inner product in this space is given by the Frobenius inner product on matrices:

$$\langle \mathrm{A}, \mathrm{B}\rangle = \operatorname{tr}[\mathrm{AB}]. \tag{3.10}$$

If we expand A and B in the basis $\mathcal{P}$, where P_i denotes the i-th element of $\mathcal{P}$, we can see that this inner product is equivalent to the standard inner product in $\mathbb{R}^{4^n}$:

$$\begin{aligned}\operatorname{tr}[\mathrm{AB}] &= \operatorname{tr}\left[\sum_i a_i \mathrm{P}_i \sum_j b_j \mathrm{P}_j\right] \\ &= \sum_{ij} a_i b_j \operatorname{tr}[\mathrm{P}_i\mathrm{P}_j] = \sum_{ij} a_i b_j \delta_{ij} \\ &= \sum_i a_i b_i.\end{aligned} \tag{3.11}$$

[2]This value would be attained if infinitely many repreated measurements were allowed. We discuss the relationship between the number of measurements and the estimation accuracy in chapter 6.

The identity $\mathrm{tr}[\mathrm{P}_i\mathrm{P}_j] = \delta_{ij}$ follows from $\mathrm{tr}[\Sigma] = 0$ for $\Sigma \in \{\mathrm{X}, \mathrm{Y}, \mathrm{Z}\}$, and

$$\mathrm{tr}[\mathrm{P}_i^2] = 2^{-n}\,\mathrm{tr}[\mathrm{I}^{\otimes n}] = 1. \tag{3.12}$$

In summary, the expectation value $\langle\Phi(\mathbf{x})|\mathrm{F}(\boldsymbol{\theta})|\Phi(\mathbf{x})\rangle$ is precisely the inner product between the normal vector of a 4^n-dimensional hyperplane $\mathbf{w}(\boldsymbol{\theta})$ with entries $w_i(\boldsymbol{\theta}) = \mathrm{tr}[\mathrm{F}(\boldsymbol{\theta})\mathrm{P}_i]$, and the 4^n-dimensional representation $\rho_\Phi(\mathbf{x})$; this inner product combined with a thresholding function is a standard description of SVM classifiers.

Since we are looking to produce a quantum classifier which labels as many points correctly as possible, finding a hyperplane that gives rise to such a classifier is an optimisation process. In fact, this optimisation process has an alternative formulation, as described by Havlíček et al. Instead of considering hyperplanes, one can also formulate the problem in terms of inner products between feature vectors; this inner product is called the *kernel*, denoted $k(\cdot,\cdot)$. The authors show that the natural kernel form for this variational learning setup reads

$$k(\mathbf{x},\mathbf{x}') = \mathrm{tr}[\rho_\Phi(\mathbf{x})\rho_\Phi(\mathbf{x}')] = |\langle\Phi(\mathbf{x})|\Phi(\mathbf{x}')\rangle|^2 \tag{3.13}$$

in accordance with the inner product in eq. 3.10, and that the corresponding classifier is given by

$$c(\mathbf{x}) = \mathrm{sgn}\left(\sum_{(\mathbf{x}',y')\in\mathcal{T}} \alpha_{\mathbf{x}'} y'\, k(\mathbf{x},\mathbf{x}') + b\right) \tag{3.14}$$

where $\alpha_{\mathbf{x}} \geq 0$. We will derive this expression for general (including classical) SVMs in section 4.2, and the link to the quantum formulation in section 5.1. Note that the optimisation of the kernel classifier is no longer a variational process: since the only free parameters in eq. 3.14 are the coefficients $\alpha_{\mathbf{x}'}$, the optimisation of these parameters can be moved to a classical computer, while the kernels are evaluated on a quantum computer.

We thus have two representations of this quantum classifier; following the paper by Schuld and Killoran [25], who proposed the same type of quantum classifier in their work published independently from that of Havlíček et al., we shall call the hyperplane representation the *explicit* form, and the refer to the kernel representation as the *implicit* form. This distinction now begs the question: what is the relationship between quantum explicit and implicit classifiers? Does one have an advantage over the other? We discuss these questions in section 5.2.

Havlíček et al. argue that, in order to achieve any quantum advantage at all through this learning method, the kernel $k(\cdot,\cdot)$ ought to be hard to

estimate classically; otherwise we might as well run the entire process on a classical computer. They thus remark that a hard-to-estimate kernel could provide a source of quantum advantage. In this context, they give motivation for the application of their quantum learning method through a conjecture that the kernel defined by eq. 3.13 and the feature map in eq. 3.5 is hard to estimate. This conjecture is based on the resemblance of the feature map to a quantum circuit used by Rötteler [26] to show the existence of an oracle relative to which P is separated from BQP. Since this quantum circuit can efficiently solve the so-called hidden shift problem for bent Boolean functions with help of an oracle, which is not possible using a classical computer, this suggests that some form of advantage in terms of efficiency could be achieved by using this feature map. However, this is not a complete proof; but since this is a highly involved complexity theoretic question, any further discussion is out of the scope of this work.

Besides the papers of Havlíček et al. and of Schuld and Killoran, recently a number of papers [27, 28] have been published which discuss quantum kernels in the same sense for binary quantum classification. These, however, mainly focus on details regarding the implementation of specific kernels, which is an objective different from that of our work.

In the following chapters, we formalise and elaborate the algebraic theory between explicit and implicit classifiers, both in a general (chapter 4) and the quantum (chapter 5) setting, including discussions on generalisation performance. Furthermore, we elaborate on possibilities for quantum advantage in chapter 6, by giving a precise definition of hardness, and discussing consequences for learning.

Chapter 4

Feature space supervised learning

The aim of this chapter is to formalise the mathematical description of linear classifiers which are presented in quantum form by Havlíček et al. To this end, we use functional analysis to define such classifiers (section 4.1), which naturally leads to the distinction between hyperplane and kernel classifiers, and reveals a useful property known as the representer theorem (section 4.2). Subsequently, we discuss an important aspect of machine learning, namely generalisation to unseen data, from the perspective of statistical learning theory with linear classifiers (section 4.3). These notions will be revisited in chapter 5, where we will use them in order to characterise learning properties of quantum hyperplane and kernel classifiers.

4.1 Continuous linear functional classifiers

We shall work from the ground up, putting into place first the mathematical framework describing the representation of the classification method. The representation space is a Hilbert space (i.e. a vector space endowed with an inner product), and the core of our classifiers is a continuous linear functional. As we will later see, this type of representation has nice mathematical properties, and connects well to what can be achieved on NISQ devices in terms of supervised learning (chapter 5).

Definition 4.1. Let $\mathcal{F}$ be a Hilbert space with inner product norm $\|f\|_{\mathcal{F}} = \sqrt{\langle f, f\rangle_{\mathcal{F}}}$. A *continuous linear functional* is a linear map $L : \mathcal{F} \to \mathbb{R}$ which is bounded in the sense that

$$\exists M \in \mathbb{R}_{>0} : \forall f \in \mathcal{F} : |L(f)| \leq M \|f\|_{\mathcal{F}}. \tag{4.1}$$

Such functionals have the following useful property.

Theorem 4.2 (Riesz representation theorem [29, theorem 2.14])**.** *All continuous linear functionals L from a Hilbert space $\mathcal{F}$ to the real numbers can be expressed as $L(f) = \langle f, \phi \rangle_{\mathcal{F}}$ for some $\phi \in \mathcal{F}$.*

Clearly, the space of continuous linear functionals from $\mathcal{F}$ to $\mathbb{R}$ is isomorphic to $\mathcal{F}$. This is the dual space of $\mathcal{F}$.

Consider now Hilbert spaces of functions $f : \mathcal{X} \to \mathbb{R}$, where $\mathcal{X}$ is any set. In such Hilbert spaces, there exists for all $x \in \mathcal{X}$ an *evaluation functional* δ_x which maps a function f to its function value $f(x)$. This notion leads to the definition of a reproducing kernel Hilbert space.

Definition 4.3. A Hilbert space $\mathcal{F}$ of functions $f : \mathcal{X} \to \mathbb{R}$ is a *reproducing kernel Hilbert space* (RKHS for short) if its evaluation functional $\delta_x : f \mapsto f(x)$ is a continuous linear functional for all $x \in \mathcal{X}$.

From theorem 4.2 it follows that every function f in an RKHS $\mathcal{F}$ has, for all $x \in \mathcal{X}$, an evaluation of the form

$$f(x) = \langle f, \phi_x \rangle_{\mathcal{F}} \tag{4.2}$$

for some $\phi_x \in \mathcal{F}$ that is dependent on x. We call ϕ_x a *feature function* or *feature vector* for x. Such a feature function shall be our representation of a point x. Note that the space of all feasible representations is the dual space of $\mathcal{F}$.

Now, a RKHS is a special kind of Hilbert space, which allows for a neat functional relationship between two points in the representational space, called a kernel.

Definition 4.4. Let $\mathcal{F}$ be a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$. A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is said to be a *reproducing kernel*, or kernel for short, of $\mathcal{F}$, if

(1) $\forall x \in \mathcal{X} : k(\cdot, x) \in \mathcal{F}$, and

(2) $\forall x \in \mathcal{X} : \forall f \in \mathcal{F} : \langle f, k(\cdot, x) \rangle_{\mathcal{F}} = f(x)$.

Following this definition, a kernel k must satisfy $\langle k(\cdot, x), k(\cdot, y) \rangle_{\mathcal{F}} = k(x, y)$.

Theorem 4.5 (Sejdinovic and Gretton [30, proposition 29])**.** *A Hilbert space of functions $\mathcal{F}$ is a RKHS if and only if it has a kernel k. This kernel is unique.*

Corollary 4.6. *Every RKHS has a unique kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ of the form*

$$k(x, y) = \langle \phi_x, \phi_y \rangle_{\mathcal{F}} \ \ \forall\, x, y \in \mathcal{X} \tag{4.3}$$

where ϕ_x and ϕ_y are feature functions for x and y respectively.

Proof. From theorem 4.2 and condition 2 in definition 4.4 we have

$$\forall f \in \mathcal{F} : \forall x \in \mathcal{X} : \langle f, \phi_x \rangle_{\mathcal{F}} = f(x) = \langle f, k(\cdot, x) \rangle_{\mathcal{F}}, \tag{4.4}$$

implying that $k(\cdot, x) = \phi_x$, and therefore

$$k(x, y) = \langle k(\cdot, x), k(\cdot, y) \rangle_{\mathcal{F}} = \langle \phi_x, \phi_y \rangle_{\mathcal{F}}. \tag{4.5}$$

Q.E.D.

If the input space $\mathcal{X}$ is a compact metric space (like e.g. $\mathbb{R}^N$), there is another route to a RKHS and feature functions, known as *Mercer's condition*. We shall briefly state without proof the relevant theorem and its consequences.

Theorem 4.7 (Mercer's condition [31])**.** *Let $\mathcal{X}$ be a compact metric space and $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ a continuous positive-semidefinite kernel in the sense that*

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j k(x_i, x_j) \geq 0 \tag{4.6}$$

for all finite sequences of points $x_1, \ldots, x_n \in \mathcal{X}$ and all $c_1, \ldots, c_n \in \mathbb{R}$. Then there exist orthonormal functions e_i such that

$$k(x, y) = \sum_{j} \lambda_i e_i(x) e_i(y) \tag{4.7}$$

and, taking $\phi_x = \sum_i \sqrt{\lambda_i} e_i(x)$, we have

$$k(x, y) = \langle \phi_x, \phi_y \rangle, \tag{4.8}$$

hence the space of these functions is a RKHS.

Having established the mathematical formulation of the representational space we will be working with, we are now ready to give the definition of a *continuous linear functional classifier*, which is defined on a binary label set $\mathcal{Y} = \{+1, -1\}$.

Definition 4.8. A continuous linear functional (CLF) classifier is a function $c_f : \mathcal{X} \to \{+1, -1\}$ of the form

$$c_f(x) = \operatorname{sgn}(f(x) - d) \tag{4.9}$$

with $d \in \mathbb{R}$ and f an element of a RKHS $\mathcal{F}$, i.e. $f(x) = \delta_x f$ with $\delta_x : \mathcal{F} \to \mathbb{R}$ a continuous linear evaluation functional. We call f the *classification function* of c_f.

4.2 Explicit and implicit classifiers

We now proceed to introduce more practical expressions of CLF classifiers in the form of *explicit* and *implicit* classifiers. We will construct our classifiers by defining the underlying RKHS according to a prior chosen finite-dimensional feature map. In this chapter, we establish all notions in a classical setting, aided by examples from a maximum-margin classifier (also known as support vector machine or SVM); in chapter 5, we make the transition to quantum CLF classifiers.

Definition 4.9. An explicit classifier c_ϕ or *feature classifier* on a training set $\mathcal{T} \subseteq \mathcal{X} \times \mathcal{Y}$ is a CLF classifier whose classification function space $\mathcal{F}$ is a real, finite-dimensional Hilbert space with the standard vector dot product and whose classification function evaluation is determined by a feature map $\phi : \mathcal{X} \to \mathcal{F} : x \mapsto \boldsymbol{\phi}(x)$ as

$$f(x) = \mathbf{w} \cdot \boldsymbol{\phi}(x). \tag{4.10}$$

As such, the classifier is of the form $c(x) = \operatorname{sgn}(\mathbf{w} \cdot \boldsymbol{\phi}(x) - d)$.

That is, the set of classifiers is the set of separating hyperplanes in feature space that assigns $+1$ to points on one side of the plane, and -1 to points on the other side.

The following lemma asserts that this type of classification function forms as RKHS and thus inherits all the useful properties of such a space.

Lemma 4.10. *The classification function space $\mathcal{F}$ of an explicit classifier is a RKHS as long as there exists some $M \in \mathbb{R}$ such that $\|\phi(x)\|_\mathcal{F} \leq M \ \forall x \in \mathcal{X}$.*

Proof. The vector space $\mathbb{R}^n$ endowed with the standard inner product is a Hilbert space; therefore $\mathcal{F}$ is as well. To see it is also a RKHS, observe that by the Cauchy-Schwarz inequality

$$\begin{aligned} |\delta_x f| &= |\mathbf{w} \cdot \boldsymbol{\phi}(x)| \\ &\leq \|\mathbf{w}\| \cdot \|\boldsymbol{\phi}(x)\| \\ &= \|f\|_\mathcal{F} \cdot \|\phi(x)\|_\mathcal{F} \\ &\leq \|f\|_\mathcal{F} \cdot \sup_{x \in \mathcal{X}(\mathcal{T})} \|\phi(x)\|_\mathcal{F} \\ &\leq M \|f\|_\mathcal{F}, \end{aligned} \tag{4.11}$$

implying that δ_x is a continuous linear evaluation functional. Q.E.D.

As we discussed before, one can regularise the error measure to limit overfitting and thereby increase generalisation performance. For CLF classifiers, we shall make use of function norm regularisation. We call the combination of an error measure and regularisation a *regularised risk functional*, whose minimum is to be considered the optimal classification function on a training set $\mathcal{T}$.

Definition 4.11. A regularised risk functional on a training set $\mathcal{T}$ and a RKHS $\mathcal{F}$ is a functional $R : \mathcal{F} \to \mathbb{R}$ of the form

$$R[f] = E\big(\{(x, y, c_f(x)) : (x, y) \in \mathcal{T}\}\big) + s(\|f\|_{\mathcal{F}}) \tag{4.12}$$

where $c_f(\cdot)$ is a (CLF) classifier with classification function $f(\cdot)$, E is any arbitrary error measure with respect to $\mathcal{T}$, and s is a strictly monotonically increasing function of the norm of f.

We shall later give a motivation for this particular choice of regularisation, and discuss how it affects generalisation performance.

Let us now consider a widely used example of a feature classifier, namely the *support vector machine*.

Example 4.12. A support vector machine (SVM) is a CLF classifier in the Hilbert space of $\mathbb{R}^N$ with the standard inner product. In the explicit formulation of a SVM, its classification function f is given by an N-dimensional normal vector $\mathbf{w}$ which describes a hyperplane in $\mathbb{R}^N$. Its entries and the offset $d \in \mathbb{R}$ are trained to separate a training set $\mathcal{T}$ by a *maximal margin*: i.e. a hyperplane whose distance

$$D((\mathbf{w}, d), \boldsymbol{\phi}) = \frac{|\mathbf{w} \cdot \boldsymbol{\phi} - d|}{\|\mathbf{w}\|} \tag{4.13}$$

to the closest point of either class is maximal. Under the conditions

$$y(\mathbf{w} \cdot \boldsymbol{\phi}(x) - d) \geq 1 \quad \forall (x, y) \in \mathcal{T}, \tag{4.14}$$

the training set is linearly separated by the hyperplane, by a margin of $2/\|\mathbf{w}\|$. As such, a maximal-margin separating hyperplane can be found by maximising $2/\|\mathbf{w}\|$, or equivalently minimising $1/2\,\|\mathbf{w}\|^2$, under the conditions in eq. 4.14. This can be formulated as a minimisation of the primal cost lagrangian

$$L_P(\mathbf{w}, d, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{(x,y)\in\mathcal{T}} \alpha_x[y(\mathbf{w} \cdot \boldsymbol{\phi}(x) - d) - 1] \tag{4.15}$$

where $\alpha_x \geq 0$ are the Lagrange multipliers introduced to assure the conditions of eq. 4.14 are met. Note that for all $d \in \mathbb{R}$, $\boldsymbol{\alpha} \in \mathbb{R}^T_{\geq 0}$, L_P is indeed a regularised risk functional on $\mathbf{w}$, as the first term is a strictly monotonically increasing function of $\|f\|$ and the second term is a valid error function of f on $\mathcal{T}$.

The second type of classifier, the implicit kind, is closely related to the explicit classifier, but is expressed directly in terms of the kernel of the underlying RKHS.

Definition 4.13. An implicit classifier c_{k_ϕ} or *kernel classifier* on a training set $\mathcal{T}$ is a finite-dimensional Hilbert space CLF classifier whose function evaluation is of the form

$$f(x) = \sum_{(x',y') \in \mathcal{T}} \alpha_{x'} \, k_\phi(x', x) \tag{4.16}$$

where $\alpha_{x'} \in \mathbb{R}$, and the kernel k_ϕ is given by the finite-dimensional inner product:

$$k_\phi(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{F}} = \boldsymbol{\phi}(x) \cdot \boldsymbol{\phi}(x') \tag{4.17}$$

with ϕ a feature map as in definition 4.9.

Henceforth, by slight abuse of notation, we shall write $x \in \mathcal{T}$ to mean $(x, y) \in \mathcal{T}$ if y does not appear in the corresponding expression.

Now, following example 4.12, a SVM can be expressed as an implicit classifier.

Example 4.14. The SVM classifier transforms elegantly into an implicit classifier when we impose on the primal lagrangian the minimisation conditions $\partial/\partial \mathbf{w} \, L_P = \mathbf{0}$, $\partial/\partial d \, L_P = 0$:

$$\begin{aligned} \frac{\partial}{\partial \mathbf{w}} L_P = \mathbf{w} - \sum_{(x,y) \in \mathcal{T}} \alpha_x y \, \boldsymbol{\phi}(x) = \mathbf{0} \\ \Rightarrow \mathbf{w} = \sum_{(x,y) \in \mathcal{T}} \alpha_x y \, \boldsymbol{\phi}(x), \end{aligned} \tag{4.18}$$

$$\frac{\partial}{\partial d} L_P = - \sum_{(x,y) \in \mathcal{T}} \alpha_x y = 0. \tag{4.19}$$

Plugging these conditions back into eq. 4.15, we obtain the *dual lagrangian*:

$$L_D = -\frac{1}{2} \sum_{(x,y) \in \mathcal{T}} \sum_{(x',y') \in \mathcal{T}} \alpha_x \alpha_{x'} y y' \, \boldsymbol{\phi}(x) \cdot \boldsymbol{\phi}(x') + \sum_{(x,y) \in \mathcal{T}} \alpha_x. \tag{4.20}$$

Here, the kernel $k(x, x') := \boldsymbol{\phi}(x) \cdot \boldsymbol{\phi}(x')$ straightforwardly appears. The classification function $c(x)$ becomes

$$\begin{aligned} c(x) &= \operatorname{sgn}\left(\left[\sum_{(x',y')\in\mathcal{T}} \alpha_{x'} y' \boldsymbol{\phi}(x')\right] \cdot \boldsymbol{\phi}(x) - d\right) \\ &= \operatorname{sgn}\left(\sum_{(x',y')\in\mathcal{T}} \alpha_{x'} y'\, k(x', x) - d\right). \end{aligned} \tag{4.21}$$

So far, there seems to be little motivation to choose either the explicit or the implicit formulation of CLF classifiers, since both evaluate functions in the same RKHS. However, there is a gain in computational complexity that can be made by using the implicit formulation. To see this, consider the following feature map containing the scalar products between every element of an input vector $\mathbf{x} \in \mathbb{R}^N$:

$$\boldsymbol{\phi}(\mathbf{x}) = \begin{bmatrix} x_1 x_1 & \cdots & x_1 x_N & \cdots & x_N x_1 & \cdots & x_N x_N \end{bmatrix}^\top, \tag{4.22}$$

which is a vector in $\mathbb{R}^{N^2}$. Using the explicit model, one will be evaluating inner products between functions $\mathbf{w} \in \mathbb{R}^{N^2}$ and feature vectors $\boldsymbol{\phi}(\mathbf{x})$, which has a complexity on the order $O(N^2)$. The inner product of two feature vectors, on the other hand, can be cast into a more convenient form:

$$\boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{x}') = \sum_{ij} x_i x_j x'_i x'_j = \left(\sum_i x_i x'_i\right)^2 = (\mathbf{x} \cdot \mathbf{x}')^2. \tag{4.23}$$

It is apparent that, by first computing the inner product $\mathbf{x} \cdot \mathbf{x}'$ and subsequently squaring the result, one requires only $O(N)$ operations to compute an implicit classifier on the same RKHS and training set. Similar complexity gains can be shown for other types of kernels, such as radial basis function kernels [32], whose feature space is of infinite dimension. In fact, one can show using Mercer's condition (theorem 4.7) that the set of radial basis functions corresponds to a RKHS.

Besides complexity, another important result was established by Schölkopf et al. [33], regarding minimisation capabilities of CLF classifiers.

Theorem 4.15 (Representer theorem [33, theorem 1])**.** *Consider a RKHS $\mathcal{F}$ with its corresponding kernel k. Given a training set $\mathcal{T}$ and a regularised risk functional R, any function $f_* \in \mathcal{F}$ that minimises R admits a representation*

$$f_*(x) = \sum_{x'\in\mathcal{T}} \beta_{x'}\, k(x', x) \tag{4.24}$$

with $\beta_{x'} \in \mathbb{R}\ \forall x' \in \mathcal{X}$.

In other words, a CLF classifier that achieves a minimum regularised risk on any training set $\mathcal{T}$ can be expressed as an implicit classifier. We have seen a special case of this in example 4.14, where the minimisation conditions imply the existence of an implicit form, which was found when moving from the primal lagrangian to the dual. Note that, even though any implicit classifier can always be expressed as an explicit classifier – take

$$\mathbf{w} = \sum_{x'} \beta_{x'} \boldsymbol{\phi}(x') \tag{4.25}$$

so that $\mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}) = \sum_{(x,x')} \beta_{x'}\, k(x', x)$ –, the converse does not hold true, which makes the existence of an implicit form nontrivial. Indeed, the form in eq. 4.25 restricts $\mathbf{w}$ to the linear subspace spanned by the feature vector set $\{\boldsymbol{\phi}(x') : x' \in \mathcal{T}\}$, and as such, any explicit classifier with $\mathbf{w}$ outside this linear subspace cannot be represented as an implicit classifier with this training set. Still, any explicit classifier that is guaranteed to minimise a regularised risk functional can be expressed as an implicit classifier. Note however that the theorem does not necessarily hold for types of regularisation other than norm regularisation. In any case, the representer theorem will be key in the discussion of quantum CLF classifiers (section 5.2), where they will give rise to a distinction between explicit and implicit classifiers.

4.3 Generalisation performance

In the above, we have focussed mainly on the mathematical structure of our representation, and minimisation of risk on a training set. But generalisation performance is at least as important: indeed, we began this chapter noting that a classification model should generalise to unseen data points to be of any use. We used a regularisation term (definition 4.11), which manifested itself as a margin in the SVM context, for the purpose of limiting overfitting; this section will provide an argument for why this provides the desired generalisation behaviour.

A common way to view generalisation performance is through the principle of *structural risk minimisation*, which was described by Vapnik and Chervonenkis [34]. According to the principle, it is assumed that the data to be classified, together with its correct labels, comes from a joint probability distribution $p(\mathbf{x}, y)$. That is, the training set $\mathcal{T}$ is a sample from this distribution, and any feature data is expected be drawn from p as well. In this setting, a model with good generalisation performance is a classifier $c(\cdot)$

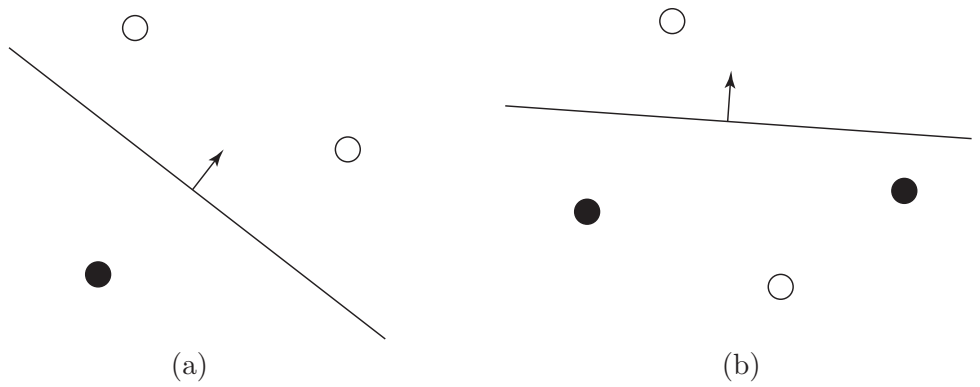


Figure 4.1. *A line in $\mathbb{R}^2$ shattering points. (a) There exists an arrangement of three points in $\mathbb{R}^2$ (namely any placement such that the points are not colinear) such that for every label assignment, there is a line that correctly labels all three points. Hence the family of all lines can shatter three points. (b) With four points, there exists a label assignment which cannot be decided by a line, no matter the arrangement of the points. Therefore the family of all lines cannot shatter four points.*

which minimises the total risk R, that is the expected error E over p:

$$R := \mathbb{E}(E) = \int_{\mathcal{X}\times\mathcal{Y}} E(c(\mathbf{x}), y)\, \mathrm{d}p(\mathbf{x}, y). \tag{4.26}$$

Clearly, for learning problems, p is not known – otherwise we wouldn't need to use any learning algorithms at all. In a learning setting, the only information available is the training set. What we can compute however, is a quantity called the *empirical risk* R_{emp} which is the average error that $c(\cdot)$ achieves on the training set:

$$R_{\mathcal{T}}^{\text{emp}} = \langle E \rangle_{\mathcal{T}} = \frac{1}{|\mathcal{T}|} \sum_{(x,y)\in\mathcal{T}} E(c(\mathbf{x}), y). \tag{4.27}$$

Usually, a classifier is said to generalise well on a distribution if the *generalisation risk* $R^{\text{gen}} = R - R^{\text{emp}}$ is upper bounded. Vapnik showed that such an upper bounded can be given for families of classifiers, which depends on the geometrical structure of the family. Let us consider more precisely what this means.

A family of classifiers is said to *shatter* a set of points in the decision space if for every possible label assignment there exists a classifier instance in this family which labels each point correctly. Then, the *Vapnik-Chervonenkis*

dimension, or VC dimension for short, of a classifier family, is the maximum number of points for which there exists a geometrical arrangement such that the points in this arrangement are shattered by the family. Take for example the family of all separating hyperplanes in $\mathbb{R}^2$, which are lines. As can be seen in figure 4.1, this family shatters at most three points; therefore its VC dimension is 3. In fact, it can be shown with relative ease [35] that the VC dimension of separating hyperplanes in $\mathbb{R}^N$ is at most $N + 1$. This gives an immediate bound on the VC dimension of implicit classifiers: since every new point is projected into the space spanned by the feature vectors $\phi(x')$, the VC dimension of an implicit classifier with N_f feature vectors weighted by nonzero $\alpha_{x'}$ is at most $N_f + 1 \leq |\mathcal{T}| + 1$.

In essence, the VC dimension defines an expressivity measure of a family of classifiers: the number of points that can be classified correctly using a classifier family is always upper bounded by the maximum number of points shattered by this family, i.e. its VC dimension. As such, we can regard families of high VC dimension to be complex and highly expressive, whereas those with low VC dimension are simple and less expressive.

While high expressivity is beneficial for the classification of complex data sets, there is a tradeoff between expressivity and generalisation performance, as we mentioned in section 3.1. As shown by Vapnik [36], the VC dimension can be used to express this tradeoff in a precise, formal way: he shows that, given the VC dimension h of the classifier family in question, the generalisation risk on data from a distribution p can be upper bounded as follows:

$$R^{\text{gen}} \leq \sqrt{\frac{h[\log(2|\mathcal{T}|/h) + 1] + \log(4/\delta)}{|\mathcal{T}|}}, \tag{4.28}$$

with probability $1 - \delta$, provided $\mathcal{T}$ is drawn from the same distribution p. We directly see that a high VC dimension – i.e. high model expressivity – comes with a high upper bound on the generalisation risk, and thus a small likelihood of good generalisation performance.

Fortunately however, it turns out that a tighter bound can be achieved on the VC dimension through a regularisation term natural to hyperplane classifiers. This regularisation term is in fact the margin, which we encountered earlier as the minimum distance between a separating hyperplane and a point of either class. Using this margin, the following bound on the VC dimension of hyperplane classifiers was shown.

Theorem 4.16 (Vapnik [16, theorem 8.3])**.** *A subset of hyperplane classifiers with classification function $f : \mathbb{R}^N \to \mathbb{R}$ taken from a real N-dimensional RKHS $\mathcal{F}$, classifying points $\phi \in \mathbb{R}^N$ subject to $\|\phi\|_{\mathcal{F}} \leq r$ (with $\|\cdot\|$ the*

standard inner product norm on $\mathbb{R}^N$), that satisfies the constraints

$$\inf_{\phi} |\langle f, \phi \rangle - d| = 1 \tag{4.29}$$

and $\|f\|_{\mathcal{F}} \leq a$ has the VC dimension h bounded above by

$$h \leq \min(r^2 a^2, N) + 1. \tag{4.30}$$

Since a hyperplane satisfying the condition in eq. 4.29 separates the set of points with margin $\gamma = 1/a$ (see example 4.12), we may also write

$$h \leq \min(r^2/\gamma^2, N) + 1. \tag{4.31}$$

According to this result, classifiers that achieve a large margin are expected to perform better in terms of generalisation, because they have a lower VC dimension. This is the main motivation to maximise the margin of SVM classifiers, which appeared as a regularisation term in example 4.12.

However, as nice as this improved result seems, unfortunately one cannot directly insert the VC dimension bound of eq. 4.31 into the generalisation risk bound of eq. 4.28. This is for the subtle reason that we cannot guarantee any future data points (such as those in the test set) to lie outside the margin γ and inside the sphere of radius r, which is an assumption made in the proof of theorem 4.16 [16, 37, 38]. In particular, if at least one point of the future data were to fall inside the margin, we would have to adjust the model to have a smaller margin to fit this test point, thus increasing its VC dimension.

Clearly, we require a generalisation bound that circumvents this problem. One such bound was shown, shortly after the publication of Vapnik's original bound, by Shawe-Taylor et al. [39]. In their proof, the authors introduce an extension to the VC dimension, called the *fat shattering dimension*.

Definition 4.17 (Shawe-Taylor et al. [39, definition 4.1])**.** Let $\mathcal{F}$ be a set of real valued functions. We say that a set of points X is γ-shattered by $\mathcal{F}$ if there are real numbers s_x indexed by $x \in X$ such that for all binary strings y indexed by $x \in X$, there is a function $f_y \in \mathcal{F}$ satisfying

$$f_y(x) \begin{cases} \leq s_x - \gamma & \text{if } y_x = -1, \\ \geq s_x + \gamma & \text{if } y_x = +1. \end{cases} \tag{4.32}$$

The fat shattering dimension $\text{fat}_{\mathcal{F}}$ of the set $\mathcal{F}$ is a function from the positive real numbers to the integers which maps a value γ to the size of the largest γ-shattered set, if this is finite or infinity otherwise.

The fat shattering dimension is sometimes regarded as an 'effective VC dimension' since it plays a very similar role to the original VC dimension. In fact, if $\gamma = 0$, we recover the original VC dimension (note that, for separating hyperplanes, the VC dimension requires the s_x to be identical for all x). Interestingly, the fat shattering dimension of a set of hyperplanes separating points by a margin γ turns out to have a very similar expression to that in eq. 4.31, which further strengthens the connection between the two notions. This is captured in the following theorem.

Theorem 4.18 (Shawe-Taylor et al. [39, corollary 5.4])**.** *Let $\mathcal{F}$ be the set of linear functions of the form*

$$f(\phi) = \langle f, \phi \rangle - d \tag{4.33}$$

with $\|f\| = 1$, restricted to points in a ball of N dimensions of radius u about the origin and with thresholds $|d| \leq u$. Then the fat shattering dimension of $\mathcal{F}$ can be bounded by

$$\mathrm{fat}_{\mathcal{F}}(\gamma) \leq \min\{9u^2/\gamma^2, N+1\} + 1. \tag{4.34}$$

Subsequently, it is shown that this fat-shattering dimension can be used directly to bound the generalisation risk of a set of hyperplane classifiers.

Theorem 4.19 (Shawe-Taylor et al. [39, definition 4.9])**.** *Consider a set of real-valued functions $\mathcal{F}$ having fat shattering dimension bounded above by $\mathrm{fat}_{\mathcal{F}}(\gamma)$. If a classifier with classification function $f \in \mathcal{F}$ separates all points in a training set $\mathcal{T}$ correctly and by margin γ, then with confidence $1 - \delta$ the expected generalisation risk is bounded above by*

$$R^{\mathrm{gen}}(\mathcal{T}, k) \leq \frac{2}{T}\left(k \log\left(\frac{8eT}{k}\right)\log(32T) + \log\left(\frac{8T}{\delta}\right)\right) \tag{4.35}$$

where $k = \mathrm{fat}_{\mathcal{F}}(\gamma/8)$ and $T = |\mathcal{T}|$.

Notice how this theorem provides an expression for the generalisation risk independently from future data, as the value of k is computed for a selected $f \in \mathcal{F}$ which classifies $\mathcal{T}$ correctly, and the margin γ it achieves in doing so, before being inserted in expression 4.35. As such, this theorem is in a correct form to allow new points: unlike the generalisation risk bound in theorem 4.28, theorem 4.19 only conditions on a fixed set of points being separated by some margin γ, instead of all future points. This makes the fat shattering dimension suitable for direct insertion into eq. 4.35. From inequality 4.34, then, we see that the benefit of achieving high margin still remains: the

higher the margin, the lower the fat shattering dimension and the lower the upper bound on the generalisation risk. However, a difference between eqs. 4.35 and 4.28 is that the latter is a square root dependence, where the former is linear.

In chapter 6, we consider how quantum hyperplane classifiers compare to classical ones in terms of fat shattering dimension and generalisation performance.

Chapter 5

Quantum feature space learning

In this chapter, we combine the concepts of classical supervised feature space learning from chapter 4 with the theory of quantum computing as discussed in chapter 2 in order to precisely define the quantum counterpart of CLF classifiers which, like the classical versions, can be expressed both in the explicit and implicit formulations. In doing so, we follow the work of Havlíček et al., who describe quantum explicit and implicit classifiers as discussed in chapter 3. The objective of this chapter is to build upon this work, providing a thorough construction of these classifiers, and a comparison between quantum explicit and implicit classifiers. Section 5.1 introduces the classifiers describing the feature space, the classification function formulations and discusses the conditions in which the classifiers can be evaluated on a quantum computer, aided by the example of a quantum SVM. Subsequently, section 5.2 provides a comparison between the two types of classifiers in terms of the structure of their respective classification function spaces under computability conditions, with a discussion on training set classification performance and connections to other known NISQ learning algorithms. Lastly, in section 5.3 we apply the notions of generalisation from section 4.3 to quantum CLF classifiers. We derive a tight upper and lower bound for the fat shattering dimension of general quantum CLF classifiers, and discuss consequences for the choice of a classifier that generalises well to new data.

5.1 Quantum CLF classifiers

We shall now define quantum CLF classifiers, considering the explicit formulation first.

Definition 5.1. A quantum explicit classifier is an explicit classifier c_Φ

whose function space is the space $H(2^n)$ of quantum observables on n qubits, equipped with a feature map Φ which maps $x \in \mathcal{X}$ onto the subset of n-qubit pure density matrices through a polynomial size quantum circuit.

The feature map Φ maps x to a pure state $\rho_\Phi(x) = |\Phi(x)\rangle\langle\Phi(x)|$. In the feature space, $\rho_\Phi(x)$ has unit norm, since $\|\rho_\Phi(x)\|_H = \sqrt{\mathrm{tr}[\rho_\Phi]^2} = 1$ for all x.

From this definition, we can see that the space of quantum explicit classification functions is a RKHS by lemma 4.10, since $\|\rho_\Phi(x)\|_H = 1 \ \forall x$. Furthermore, we observe that a quantum explicit classifier can indeed be evaluated using a quantum computer. Firstly, the feature map may be realised as a unitary transformation $\mathrm{U}_\Phi(x)$ on the initial state $|0\rangle$, so that $\rho_\Phi(x) = \mathrm{U}_\Phi(x)|0\rangle\langle 0|\mathrm{U}_\Phi^\dagger(x)$. Secondly, the inner product in observable space is represented by the computation of the expectation value of a quantum observable $\mathrm{H} \in H(2^n)$ for a system in the state $\rho_\Phi(x)$:

$$\begin{aligned} f(x) &= \mathrm{tr}[\mathrm{H}\rho_\Phi(x)] \\ &= \langle\Phi(x)|\mathrm{H}|\Phi(x)\rangle. \end{aligned} \tag{5.1}$$

We see that the observable H plays the same role as the normal vector $\mathbf{w}$ of the separating hyperplane appearing in the classical explicit classifier construction. However, the fact that it is a matrix gives rise to a particular parametrisation. After all, by the spectral theorem we may write any hermitian observable H as a spectral decomposition $\mathrm{W}^\dagger\mathrm{D}\mathrm{W}$ with W a unitary operator and $\mathrm{D} = \sum_z \lambda(z)|z\rangle\langle z|$ a real diagonal matrix. Therefore we have

$$f(x) = \langle 0|\mathrm{U}_\Phi^\dagger(x)\mathrm{W}^\dagger\mathrm{D}\mathrm{W}\mathrm{U}_\Phi(x)|0\rangle. \tag{5.2}$$

The operator W can be parametrised as a sequence of continuous unitary gates. Expression 5.2 then gives us a straightforward way to implement the quantum classifier: after preparation of the state $|\Phi(x)\rangle = \mathrm{U}_\Phi(x)|0\rangle$, we can measure the expectation $\langle\mathrm{H}\rangle$ by applying $\mathrm{W}_{\boldsymbol{\theta}}$ via a quantum circuit, performing a measurement in the computational basis, and computing $\lambda(z)$ on the outcome z. Repetition of this preparation and measurement process then yields an approximate expectation value for $f(x)$. Since $\mathrm{W}_{\boldsymbol{\theta}}$ is parametrised, one can implement a learning procedure by optimising the circuit for a given loss function in the sense of eq. 3.3.

However, the requirement of efficient evaluation imposes restrictions on this class of quantum classifier. After all, if one allows full freedom in the choice of the observable H, computing and therefore optimising the quantum classifier may require exponential time in the number of qubits. For one, $\lambda(z)$ may not be computable in polynomial time; therefore to ensure this quantum

learning method is feasible, one must restrict themselves to observables whose eigenvalue function λ is efficiently computable. But what is more important is the implementation of W may require exponentially many gates in the number of qubits; that is, full freedom in the choice of observables includes superpolynomial circuits whose expectation value is not BQP-computable. This places a stringent condition on the choice of observables.

Lastly, we require that the feature map unitary $\mathrm{U}_\Phi(x)$ be a poly-time circuit for all x. While this is not a restriction on the function space itself, it does mean that the set of all such unitaries is smaller than $U(2^n)$ for growing n, since general n-qubit unitaries may require exponentially many gates as mentioned in chapter 2. As such, the range of points in feature space is limited.

For completeness, let us look at the quantum formulation of the explicit SVM as in example 4.12.

Example 5.2. We can extend the notion of a SVM to quantum explicit classifiers, as a classical SVM is an example of an explicit classifier. We can follow the same steps, except the primal lagrangian of an observable H is now given by

$$L_P(\mathrm{H}, d, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathrm{H}\|^2 - \sum_{(x,y)\in\mathcal{T}} \alpha_x[y(\mathrm{tr}[\mathrm{H}\rho_\Phi(x)] + d) - 1]. \tag{5.3}$$

We observe that this quantum model can be trained to find a maximum margin hyperplane; indeed, the quantum primal lagrangian, whose minimum yields a maximum margin hyperplane, can be evaluated using a quantum computer. The second term involves an explicit classification function evaluation which we know can be carried out; the first term, then, is the squared norm of H and is equal to

$$\begin{aligned}\|\mathrm{H}\|^2 &= \langle \mathrm{H}, \mathrm{H}\rangle = \mathrm{tr}[\mathrm{HH}] = \mathrm{tr}[\mathrm{D}^2] \\ &= \sum_{z\in\{0,1\}^n} \lambda^2(z)\end{aligned} \tag{5.4}$$

with $\lambda(z)$ the eigenvalue of H corresponding to the eigenstate $|z\rangle$, being the z-th diagonal entry of D.

We have seen the norm of the classification function before, in the definition of a regularised risk functional (4.11), and it occurs here in the form of $\|\mathrm{H}\|$ playing the role of the inverse margin. This creates a problem: in order to compute the squared norm of H, a sum of exponentially many (squared) eigenvalues must be calculated, which in general requires exponential time.

In order to circumvent this, one must choose $\mathcal{H}$ such that $\sum_z \lambda^2(z)$ admits an analytical expression which can be evaluated in time $O(\text{poly}(n))$; for instance, one could use observables of at most polynomial rank, i.e. with only polynomially many nonzero eigenvalues. This, however, comes at the cost of even further restricting the freedom of observable choice.

To include maximisation of the margin in the optimisation procedure, the diagonal D must be parametrised. A simple way to do this, while adhering to the restrictions we just put forward, is to fix D beforehand – as in the description of explicit classifiers by Havlíček et al. – and allow it to scale by a factor a which can vary throughout the optimisation process. This ensures that for any observable H encountered, there exists a scaling $a_*\mathrm{H}$ such that at least one point lies precisely on the margin. Another way could be to employ a paramterised family of observables whose sum of squared eigenvalues remains poly-time computable after scaling every individual eigenvalue (which is true for polynomial-rank observables in any case).

To address the issue of exponentially costly evaluation, we now introduce the related class of quantum implicit classifiers, whose definition follows naturally from the previous considerations.

Definition 5.3. A quantum implicit classifier is an implicit classifier c_{k_Φ} whose kernel k_Φ is given by

$$k_\Phi(x, x') = \mathrm{tr}[\rho_\Phi(x)\rho_\Phi(x')] = |\langle\Phi(x)|\Phi(x')\rangle|^2, \tag{5.5}$$

and which is therefore expressed as

$$c_{k_\Phi}(x) = \mathrm{sgn}\left(\sum_{x'\in\mathcal{T}} \alpha_{x'}\,\mathrm{tr}[\rho_\Phi(x)\rho_\Phi(x')] + d\right). \tag{5.6}$$

Note that this type of classifier implicitly implements the observable

$$\mathrm{H} = \sum_{x\in\mathcal{T}} \alpha_x \rho_\Phi(x). \tag{5.7}$$

The kernel may be evaluated for a sample x using a quantum computer, by estimating the overlaps $|\langle\Phi(x)|\Phi(x')\rangle|^2$ for all x_i in the training set, which can be done by preparing the states $\mathrm{U}_\Phi^\dagger(x')\mathrm{U}_\Phi(x')|0\rangle$ and measuring the probability of finding the all-zero outcome. In this formulation, we see that no unitary W and diagonal D are present. Thus the use of this classifier eliminates the need to prepare a possibly exponentially complex parametrised circuit and to calculate a possibly exponentially expensive sum of eigenvalues.

In addition, the squared norm of the implicit observable reads

$$\begin{aligned}\|\mathrm{H}\|^2 &= \sum_{x\in\mathcal{T}}\sum_{x'\in\mathcal{T}} \alpha_x\alpha_{x'}\,\mathrm{tr}[\rho_\Phi(x)\rho_\Phi(x')] \\ &= \sum_{x\in\mathcal{T}}\sum_{x'\in\mathcal{T}} \alpha_x\alpha_{x'}\,k_\Phi(x,x') \end{aligned} \tag{5.8}$$

which can be evaluated in polynomial time if $\mathcal{T}$ contains sufficiently few points. In practice, one could even compute all kernel values $k_\Phi(x,x')$ beforehand, storing them in a table, and calculate $\|\mathrm{H}\|^2$ from this table and the values α_x.

However, the polynomial constraints on the feature map now directly restrict the function space of implicit classifiers, since its classification function is induced by the feature map. Furthermore, since any training procedure should consider each point in a training set at least once, $\mathcal{T}$ must be of size polynomial in n for learning to be efficient; this then assures that computation of $c_{k_\Phi}(x)$ requires only polynomially many kernel evaluations.

5.2 Comparison between quantum explicit and implicit classifiers

With the definitions of quantum explicit and implicit classifiers in place, there is more room for comparison between the two. Indeed, as we have seen, restrictions on the choice of the observable in the explicit model renders the two formulations more distinct in comparison to the classical case. This raises new questions: how do the classification function spaces of quantum explicit and implicit classifiers relate? Can we express quantum implicit classifiers as explicit ones? In other words, how does the representational power of these classifier types compare? We consider these questions in this section.

In this context, we point out that using the representer theorem to create a distinction between quantum explicit and implicit classifiers was mentioned by Schuld and Killoran [25]. The aim of this section is to work this out more precisely, and give an insight into the difference between the structural properties of quantum explicit and implicit classifiers.

We commence with a comparison of complete explicit and implicit function spaces, disregarding any polynomial restrictions discussed earlier. To this end, consider the function space $\mathcal{F}^{\Phi,\mathcal{H}} = \{f^\Phi_\mathrm{H} \mid \mathrm{H}\in\mathcal{H}, \mathcal{H}\subseteq H(2^n)\}$ of all quantum explicit classification functions equipped with the feature map $\Phi : x \mapsto \rho_\Phi(x)$ and characterised by an observable $\mathrm{H}\in\mathcal{H}$; and the space $\mathcal{F}^{k_\Phi,\mathcal{T}} = \{f^{k_\Phi,\mathcal{T}}_{\boldsymbol{\alpha}} \mid \boldsymbol{\alpha}\in\mathbb{R}^{|\mathcal{T}|}\}$ of quantum implicit classification functions with

kernel k_Φ as in definition 5.3. Note that $\mathcal{F}^{k_\Phi,\mathcal{T}}$ is dependent on $\mathcal{T}$ as the sum of kernels of any implicit classifier runs over all elements of $\mathcal{T}$. The following theorem then establishes the inclusion relationship between the unrestricted function spaces.

Theorem 5.4. *For any n-qubit feature map Φ and training set $\mathcal{T}$, $\mathcal{F}^{k_\Phi,\mathcal{T}}$ is a linear subspace of $\mathcal{F}^{\Phi,H(2^n)}$.*

Proof. As noted earlier, an implicit classifier implements the observable $\mathrm{H} = \sum_{(x,y)\in\mathcal{T}} \alpha_x \rho_\Phi(x)$. If $|\mathcal{T}| \geq \dim H(2^n) = 4^n$, H may be of rank 4^n, and $\mathcal{F}^{k_\Phi,\mathcal{T}} = \mathcal{F}^{\Phi,H(2^n)}$; in this case, $\mathcal{F}^{k_\Phi,\mathcal{T}}$ is of maximum cardinality since if we add an element $\tilde{x}$ to $\mathcal{T}$, the enlarged set $\{\rho_\Phi(\tilde{x})\} \cup \{\rho_\Phi(x)\}_{(x,y)\in\mathcal{T}}$ still spans a space no larger than $H(2^n)$. On the other hand, restricting $\mathcal{T}$ to fewer elements, or choosing Φ so that the $\rho_\Phi(x)$ are no longer linearly independent, results in the set of training feature vectors spanning only a subspace of $H(2^n)$, implying strict inclusion. As such, since $\mathcal{F}^{\Phi,H(2^n)}$ is a linear space (because $H(2^n)$ is), $\mathcal{F}^{k_\Phi,\mathcal{T}} \subseteq \mathcal{F}^{\Phi,H(2^n)}$ for any n-qubit Φ and training set $\mathcal{T}$. Q.E.D.

Next, consider *minimum-risk* classification functions $f_*^{\Phi,\mathcal{T},R} \in \mathcal{F}^{\Phi,\mathcal{H}}$ and $f_*^{k_\Phi,\mathcal{T},R} \in \mathcal{F}^{k_\Phi,\mathcal{T}}$, with respect to a regularised risk functional R as in definition 4.11. Observe that $f_*^{\Phi,\mathcal{T},R}$ is related to the training set through R, as it must satisfy a minimum risk on $\mathcal{T}$. Again assuming full freedom in the choice of H (i.e. $\mathcal{H} = H(2^n)$), a connection between the two types of minimum-risk classifiers is provided by the representer theorem (theorem 4.15), which leads directly to the following statement.

Lemma 5.5. *For any feature map Φ and training set $\mathcal{T}$, the minimum-risk element $f_*^{\Phi,\mathcal{T},R}$ of the space $\mathcal{F}^{\Phi,H(2^n)}$ of quantum explicit classification functions, with respect to a regularised risk functional R on $\mathcal{T}$ and $\mathcal{F}^{\Phi,H(2^n)}$, is contained in the corresponding space of quantum implicit classification functions $\mathcal{F}^{k_\Phi,\mathcal{T}}$, and is the minimum-risk element $f_*^{k_\Phi,\mathcal{T},R}$ of $\mathcal{F}^{k_\Phi,\mathcal{T}}$.*

Proof. $\mathcal{F}^{\Phi,H(2^n)}$ is a RKHS by lemma 4.10 and thus obeys the representer theorem, implying that $f_*^{\Phi,\mathcal{T},R}$ has a kernel representation and is therefore an element of $\mathcal{F}^{k_\Phi,\mathcal{T}}$. Since $\mathcal{F}^{k_\Phi,\mathcal{T}} \subseteq \mathcal{F}^{\Phi,H(2^n)}$ by theorem 5.4, $f_*^{\Phi,\mathcal{T},R}$ is also the minimum-risk element $f_*^{k_\Phi,\mathcal{T},R}$ of $\mathcal{F}^{k_\Phi,\mathcal{T}}$. Q.E.D.

In other words, even though a set of implicit classification functions $\mathcal{F}^{k_\Phi,\mathcal{T}}$ may be a (strict) subset of a set of explicit functions $\mathcal{F}^{\Phi,\mathcal{H}}$, it always contains the minimum-risk element for any Φ and $\mathcal{T}$. Restrictions on the observable set $\mathcal{H}$, then, may only increase the risk value $R[f_*^{\Phi,\mathcal{T},R}]$ of the minimum-risk element of $\mathcal{F}^{\Phi,\mathcal{H}}$. We can express this result as follows.

Theorem 5.6. *Let Φ be a feature map, $\mathcal{T}$ a training set, $\mathcal{F}^{\Phi,\mathcal{H}} \subseteq \mathcal{F}^{\Phi,H(2^n)}$ a set of quantum explicit classification functions, $\mathcal{F}^{k_\Phi,\mathcal{T}} \subseteq \mathcal{F}^{\Phi,H(2^n)}$ a set of quantum implicit classification functions and R a regularised risk functional on $\mathcal{T}$ and $\mathcal{F}^{\Phi,H(2^n)}$. Let $f_*^{\Phi,\mathcal{T},R} \in \mathcal{F}^{\Phi,\mathcal{H}}$ and $f_*^{k_\Phi,\mathcal{T},R} \in \mathcal{F}^{k_\Phi,\mathcal{T}}$ be the minimum-risk elements with respect to R of $\mathcal{F}^{\Phi,\mathcal{H}}$ and $\mathcal{F}^{k_\Phi,\mathcal{T}}$ respectively. Then*

$$R[f_*^{\Phi,\mathcal{T},R}] \geq R[f_*^{k_\Phi,\mathcal{T},R}] \tag{5.9}$$

for any Φ, $\mathcal{T}$ and R.

We can illustrate this theorem by imposing a specific restriction on $\mathcal{H}$, for example to the subspace of $H(2^n)$ orthogonal to the subspace spanned by the feature density operators of an implicit classifier. For example, consider the feature map given by $\mathrm{U}_\Phi(x) = \mathrm{R}_\mathrm{X}(\phi(x))$. The corresponding feature density operators are

$$\rho_\Phi(x) = \frac{1}{2}(\mathrm{I} - \sin\phi(x)\,\mathrm{Y} + \cos\phi(x)\,\mathrm{Z}) \tag{5.10}$$

and hence all observables H_Φ "spanned" by this feature map are of the form $\mathrm{H}_\Phi = a_\mathrm{I}\mathrm{I} + a_\mathrm{Y}\mathrm{Y} + a_\mathrm{Z}\mathrm{Z}$. Then, if $\mathcal{H} = \{a_\mathrm{X}\mathrm{X} \,|\, a_\mathrm{X} \in \mathbb{R}\}$, $\mathcal{F}^{k_\Phi,\mathcal{T}}$ is clearly no longer included in $\mathcal{F}^{\Phi,\mathcal{H}}$. This can also be seen in a more roundabout way from the fact that

$$\mathrm{tr}[a_\mathrm{X}\mathrm{X}\rho_\Phi(x)] = a_\mathrm{X}\langle 0|\mathrm{R}_\mathrm{X}(-\phi(x))\,\mathrm{X}\,\mathrm{R}_\mathrm{X}(\phi(x))|0\rangle = a_\mathrm{X}\langle 0|\mathrm{X}|0\rangle = 0 \tag{5.11}$$

(note that R_X and X commute), which shows that observables in $\mathcal{H}$ cannot distinguish between any feature density operator, implying that a minimum risk cannot be achieved by an observable in $\mathcal{H}$, and hence $\mathcal{F}^{k_\Phi,\mathcal{T}} \nsubseteq \mathcal{F}^{\Phi,\mathcal{H}}$ by the representer theorem. Note that if we add $a_\mathrm{I}\mathrm{I}$ to an observable $\mathrm{H} \in \mathcal{H}$, the resulting observable still makes no distinction between density matrices since $\mathrm{tr}[\mathrm{I}\rho] = 1$ for any ρ. As such, the same argument can be made for any explicit observable sets that are orthogonal to $\mathcal{H}_\Phi = \{\mathrm{H}_\Phi\}$ only with respect to the non-identity components. Of course, if $\mathcal{H}$ is only partially orthogonal to $\mathcal{H}_\Phi$, it still does not include $\mathcal{H}_\Phi$, but it may contain the minimum-risk element depending on the labels and the error measure.

Besides this artificial restriction, there is a case that may occur in practice: namely that in the decomposition $\mathrm{H} = \mathrm{W}^\dagger\mathrm{D}\mathrm{W}$, the diagonal is fixed up to a scaling factor – a scenario we mentioned earlier in the discussion of evaluation complexity of explicit classifiers. In this case, it turns out that generally $\mathcal{F}^{k_\Phi,\mathcal{T}} \nsubseteq \mathcal{F}^{\Phi,\mathcal{H}}$ for the reason that $\mathcal{F}^{\Phi,\mathcal{H}}$ is not a linear space. To see this, take two such observables H and H′ which we write $\mathrm{H} = \mathrm{W}^\dagger\mathrm{D}\mathrm{W}$ and $\mathrm{H}' = (\mathrm{W}')^\dagger\mathrm{D}(\mathrm{W}')^\dagger$ with W, W' elements of some parametrised circuit set

$\mathcal{W} \subseteq U(2^n)$. If this set of observables forms a linear space, then the equation $a\mathrm{H} + b\mathrm{H}' = c\mathrm{H}'' = c(\mathrm{W}'')^\dagger \mathrm{D}\mathrm{W}''$, with $a, b, c \in \mathbb{R}$ and $\mathrm{W}'' \in \mathcal{W}$ must hold. For this however, the eigenvalues of H and $\mathrm{H} + \mathrm{H}'$ have to match, which is unlikely, since H and H' typically diagonalise in a different basis [40]. This is illustrated by the fact that for a single qubit, even when all unitaries W are allowed (i.e. $\mathcal{W} = U(2)$) and $\mathrm{D} = \mathrm{diag}(k, l)$, the eigenvalues of $a\mathrm{H} + b\mathrm{H}'$ can be shown to be of the form

$$\lambda_\pm = \frac{1}{2}(a+b)(k+l) \pm \sqrt{(k-l)^2\, r(a, b, \boldsymbol{\theta}, \boldsymbol{\theta}')} \tag{5.12}$$

where $\boldsymbol{\theta}, \boldsymbol{\theta}'$ are the parameters of W, W' respectively, and r is an oscillating function of $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$. This shows a match for general $\boldsymbol{\theta}, \boldsymbol{\theta}'$ if and only if $k = l$, or $\mathrm{D} = k\mathrm{I}$, a trivial case.

Besides comparison between explicit and implicit in terms of the representer theorem, there is another important point to be made. In machine learning, one typically seeks to find a classification model that, while displaying good performance on the training set, is as simple as possible, in order to prevent overfitting and enhance generalisation performance. The natural way to simplify an implicit classifier, then, is to set some of the coefficients α_x to zero; this method is used by SVMs, which set α_x to zero for any feature vector that does not lie precisely on the margin (the vector with nonzero α_x are called *support vectors*). An explicit classifier can mimic this by using an observable of low rank: after all, the rank of an implicit observable (eq. 5.7) is at most the number of elements with nonzero α_x. However, explicit classifiers offer an additional route to simplification, namely via structural restrictions in the observable, either via the diagonal measurement operator D, or the parametrised circuits W. This is something implicit classifiers lack by definition, and which therefore gives explicit an edge when it comes to generalisation performance.

In summary, we have the following main findings.

- The set of all quantum implicit classification functions (and hence classifiers) on the Hilbert space $H(2^n)$ with respect to a feature map Φ is contained in the set of all quantum explicit classification functions on $H(2^n)$ with Φ.

- Nonetheless, a classifier that minimises a regularised risk functional (definition 4.11) can always be expressed as an implicit classifier.

- For restricted subsets of quantum explicit classification functions which do not contain the set of quantum implicit classification functions induced by a training set $\mathcal{T}$, such as sets that form no vector space or

those that are (partially) orthogonal to the implicit set, the minimum-risk explicit classifier is not guaranteed to attain the same risk value as the minimum-risk implicit classifier.

Of course, these findings are conditioned on the use of a regularised risk functional, whose regularisation term is limited to a monotonically increasing function of the function norm. For other choices of a regularisation term, for example one that decreases in the value of the norm at some point, or is not expressible in terms of the norm at all, these results do not hold.

Lastly, let us make a connection between the two types of quantum CLF classifiers and other quantum machine learning methods. We have seen that the set of quantum explicit classification functions is in fact the set of supervised variational quantum classification functions: indeed, such algorithms typically encode data points x into a state $|\Phi(x)\rangle$ and optimise an expectation value represented by a variational circuit and a measurement observable (cf. eq. 5.1). This approach is for example very common in the description of quantum neural networks [41, 42]. Now, such classification methods could witness an improvement in classification performance by expressing them as implicit classifiers. That is to say, from the representer theorem (theorem 4.15) and corollary 5.5 we know that the optimal variational classifier with respect to some regularised error is be contained in the space of corresponding kernel representations; as such, casting a classifier into an implicit form circumvents the restrictions in the variational circuit that may prevent it from reaching a global optimum. Nonetheless, there are still reasons to opt for certain restricted explicit classification function sets. One could be to follow a known ansatz; for example, the multi-scale entanglement renormalisation anzatz (MERA) has been studied as an efficient representation of many-body states [43] and its structural properties have been applied with success in supervised quantum learning [44]. Another incentive could be to achieve approximate optima with few resources by applying shallow-circuit observables.

However, for a large class of restricted variational learning schemes, namely those that seek to optimise the expectation value of some fixed problem hamiltonian (including well-known algorithms such as the variational quantum eigensolver [24] and the quantum approximate optimisation algorithm [45]), this is not applicable: we have seen that this class of functions is not a linear space, and hence we cannot use the representer theorem to construct a corresponding implicit model. Indeed, the optimum classifier would fall outside the set of fixed-hamiltonian circuits, and would therefore solve a problem that is different from the original.

5.3 Generalisation performance of quantum CLF learning

As we argued in the previous chapter, besides knowledge about classification performance on training sets, we also require an idea of the generalisation performance to paint a complete picture of the learning capabilities of quantum CLF classifiers. In section 4.3, we introduced generalisation performance in the context of statistical learning theory, which we will now apply to these classifiers. In particular, we will consider the fat shattering dimension of quantum CLF classifiers (definition 4.17), which directly gives us insight into their expected generalisation error though theorem 4.19.

The fat shattering dimension of quantum learning was studied by Aaronson [46]. In his paper, he introduces quantum state learning in relation to quantum state tomography. The idea is that, for some unknown state ρ, one seeks to accurately estimate the output probabilities $\mathrm{tr}[\rho \mathrm{E}]$ for as many measurements E as possible without precisely determining ρ itself, given a set of measurement probabilities $\mathrm{tr}[\rho \mathrm{E}_i]$ for measurements $\mathrm{E}_1, \ldots, \mathrm{E}_m$. In this sense, this approach to quantum state learning is almost identical to quantum CLF learning. As such, the notion of the fat shattering dimension can be applied to quantum state learning, and Aaronson gives the following upper bound.

Theorem 5.7 (Aaronson [46, theorem 2.6])**.** *Let m and n be positive integers with $m > n$, and let $\mathcal{P}$ be the set of n-qubit density matrices. Suppose a set of observables $E = \{\mathrm{E}_1, \ldots, \mathrm{E}_m\}$ with eigenvalues in $[0, 1]$ is γ-shattered by $\mathcal{P}$; that is, there exist ρ_y for all $y \in \{0,1\}^m$, as well as real numbers $s_1, \ldots, s_m$, such that for all $y \in \{0,1\}^m$ and $i \in \{1, \ldots, m\}$,*

$$\mathrm{tr}[\rho_y \mathrm{E}_i] \begin{cases} \leq s_i - \gamma & \text{if } y_i = 0, \\ \geq s_i + \gamma & \text{if } y_i = 1. \end{cases} \tag{5.13}$$

Then $n/\gamma^2 = \Omega(m)$.

In this theorem, we can easily recognise the fat shattering dimension: if we identify $y = y_1 \ldots y_m$ as a sequence of labels, then $\mathcal{T} = \{(\mathrm{H}_i, y_i)\}_{i=1}^m$ is a training set to be γ-shattered; an upper bound on the number m of observables that can be γ-shattered by $\mathcal{P}$ is then the fat shattering dimension $\mathrm{fat}_{\mathcal{P}}(\gamma)$. Thus, $\mathrm{fat}_{\mathcal{P}}(\gamma) = O(n/\gamma^2)$ by inversion of the lower bound from the theorem.

This result however is not in the form that suits our quantum classification setting: it describes classification of observables using quantum states, while we are concerned with classifying quantum states using observables.

One might think that this issue could be easily fixed by switching the role of ρ_y and E_i in eq. 5.13. However, this is not possible, since the relation between the two is asymmetric. That is, the quantifiers cannot be simply switched, as the index of ρ_y runs over all bitstrings of length m, while that of E_i runs from 1 to m. As such, shattering states with observables is a different problem than shattering observables with states, and the fat shattering dimension in theorem 5.7 cannot be applied directly to quantum CLF learning[1]. Besides this, there is another, more intuitive reason against this bound in quantum CLF learning: considering the bound for classical CLF classifiers in theorem 4.18, which may generally scale in the dimensionality, it seems very unlikely that a bound with only logarithmic dependence would appear in the quantum case, at least for unrestricted families of observables. After all, the only restriction of quantum CLF learning is that the *data points* be pure density matrices, while in quantum state learning the *classifier family* is restricted. More precisely, it was shown by Ambainis et al. that quantum states can only store a linear amount of information in the number of qubits [47] – and this in fact forms the basis of the fat shattering dimension for quantum state learning in theorem 5.7; however for unrestricted observables, which are 4^n-dimensional vectors in the space of hermitian operators, we know that an exponential amount of information in n can be stored. Furthermore, while quantum state learning allows more freedom in the data points (measurement observables versus quantum states), the space of quantum states is still of exponential dimensionality. Together, we should expect a higher degree of expressivity for quantum CLF learning as compared to quantum state learning.

We shall now show that this intuition is indeed correct. To this end, we first revisit theorem 4.18, which establishes a bound on the fat shattering dimension of general CLF classifiers. This theorem is not quite in the form we would like for our quantum setting, since it conditions on the norm of the hyperplane being unity, whereas in the quantum setting, it is the points (the feature space density matrices) that have unit norm, while the hyperplanes (the observables) may scale freely. Fortunately, the conversion of the theorem to this setting is easy.

Lemma 5.8. *Let the set $\mathcal{F}$ of linear functions and the dimensionality N be as in theorem 4.18, but with $\|f\| \leq v$, and points restricted to unit sphere about*

[1] While in fact one can adapt the bound to CLF learning, this results in a bound that is loose (in the worst case, exponentially in n), and we can show a better bound following a different path.

the origin. Then the fat shattering dimension of $\mathcal{F}$ can be upper bounded by

$$\mathrm{fat}_{\mathcal{F}}(\gamma) \leq \min\{9v^2/\gamma^2, N+1\} + 1. \tag{5.14}$$

Proof. Let us first consider the restricted family $\tilde{\mathcal{F}}$ of functions with identical norm, i.e. $\|f\| = v$ for all $f \in \tilde{\mathcal{F}}$. Suppose $\tilde{\mathcal{F}}$ γ-shatters a set of points $\phi_1, \ldots, \phi_k$ in a ball of unit radius. Then the set of functions $\tilde{\mathcal{F}}'$ of functions with unit norm γ-shatters the set of points $\phi'_1, \ldots, \phi'_k$ where $\phi'_i = v\phi_i$, since $\langle f', \phi_i \rangle = \langle f/v, v\phi_i \rangle = \langle f, \phi_i \rangle$ for all ϕ_i. Therefore, since the points $\phi'_1, \ldots, \phi'_k$ lie inside a ball of radius v, we have by theorem 4.18 that

$$\mathrm{fat}_{\tilde{\mathcal{F}}}(\gamma) \leq \mathrm{fat}_{\tilde{\mathcal{F}}'}(\gamma) \leq \min\{9v^2/\gamma^2, N+1\} + 1. \tag{5.15}$$

To obtain or desired result, first restrict $\phi_1, \ldots, \phi_k$ to have unit norm (i.e. from the unit ball to the unit sphere). Since the fat shattering dimension cannot increase through any restriction on the data points, the bound in eq. 5.15 remains valid. Furthermore, since this bound is monotonically increasing in v, allowing functions with norm $\|f\| < v$ will not increase the fat shattering dimension (since these functions can never shatter more points by the same margin), and hence $\mathrm{fat}_{\mathcal{F}}(\gamma) \leq \mathrm{fat}_{\tilde{\mathcal{F}}}(\gamma)$, which proves our result. Q.E.D.

We thus have the following upper bound on the fat shattering dimension of quantum CLF classifiers.

Theorem 5.9. *Let $\mathcal{H}$ be a set of n-qubit quantum CLF classification functions on density matrices with matrix norm at most ν. Then the fat shattering dimension of $\mathcal{H}$ can be upper bounded by*

$$\mathrm{fat}_{\mathcal{H}}(\gamma) \leq \min\{9\nu^2/\gamma^2, 4^n + 1\} + 1. \tag{5.16}$$

Generally, we can take $\mathcal{H}$ to be a family consisting of observables of rank at most r and with eigenvalues $\lambda_1, \ldots, \lambda_r$ such that $|\lambda_i| \leq |\lambda_{\max}|$ for some $\lambda_{\max}$. In this formulation, we can express this bound in a somewhat more practical way. Namely, for any matrix H, $\|\mathrm{H}\|^2 = \sum_{i=1}^{r} \lambda_i^2$; since $\lambda_i^2 \leq \lambda_{\max}^2$, we have $\|\mathrm{H}\|^2 \leq r\lambda_{\max}^2$ and thus

$$\mathrm{fat}_{\mathcal{H}}(\gamma) \leq \min\{9r\lambda_{\max}^2/\gamma^2, 4^n + 1\} + 1. \tag{5.17}$$

It is natural to consider observable families whose largest (absolute) eigenvalues are bounded. For example, in the work of Havlíček et al., the observable family has only eigenvalues $+1$ and -1, and thus the fat shattering dimension of this family is upper bounded by $\mathrm{fat}_{\mathcal{H}}(\gamma) = O(\min\{r/\gamma^2, 4^n\})$ (in fact, this family also has a fixed rank $r = 2^n$).

Moreover, we can prove that this upper bound is tight, which we shall do by showing a corresponding lower bound. The main idea behind the proof of the lower bound is that we can adapt Vapnik's result [36] (see theorem 4.16), which was shown by using the vertices of a symmetric simplex as data points, to quantum CLF learning through the fact that orthogonal pure states form the vertices of a symmetric simplex in complex space. From this connection we can directly show the lower bound by applying a statement by Burges [37, theorem 6], which is a more accessible formulation of Vapnik's result.

Lemma 5.10. *Let $\phi_1, \ldots, \phi_N$ be points of unit norm in $\mathbb{R}^{N-1}$, with $N = 2^n$, which form the vertices of a symmetric $(N-1)$-simplex. Suppose the set $\mathcal{F}$ of linear functions $f \in \mathbb{R}^{N-1}$ with norm v γ-shatters the points $\phi_1, \ldots, \phi_k$ where $k \leq N$. Denote $u_N = 1/\sqrt{1-1/N}$. Then the set $\mathcal{H}$ of n-qubit observables with norm $u_N v$ γ-shatters the set of n-qubit pure states $\{\rho_i\}_{i=1}^k$ where $\rho_i = |i\rangle\langle i|$.*

Proof. Suppose a function $f \in \mathcal{F}$ separates $\phi_1, \ldots, \phi_k$ by margin γ under some assignment y, that is

$$\langle f, \phi_i \rangle \begin{cases} \leq s_i - \gamma & \text{if } y_i = -1, \\ \geq s_i + \gamma & \text{if } y_i = +1. \end{cases} \tag{5.18}$$

We can embed the symmetric $(N-1)$-simplex in $\mathbb{R}^N$ through the bijection $\phi_i \leftrightarrow e_i$ where the e_i are standard basis vectors of $\mathbb{R}^N$. Now consider the set $\mathcal{F}'$ of linear functions $f \in \mathbb{R}^N$ of norm $u_N v$, and take the vector $f' \in \mathbb{R}^N$ with entries $f'_i = \langle f, \phi_i \rangle$. It can be readily seen that f' has the correct norm, from the fact that the vertices ϕ_i satisfy $\langle \phi_i, \phi_j \rangle = -1/(N-1)$ for $i \neq j$. Furthermore, f' separates $e_1, \ldots, e_k$ by margin γ under y, because $\langle f', e_i \rangle = \langle f, \phi_i \rangle$ for $i = 1, \ldots, N$.
Lastly, consider the set $\mathcal{H}$ of n-qubit observables with norm $u_N v$, and note that $\mathcal{F}'$ and $\mathcal{H}$ can be directly related through the bijections $e_i \leftrightarrow \rho_i$ and

$$f' \leftrightarrow \mathrm{H} = \mathrm{diag}(f'_1, \ldots, f'_N). \tag{5.19}$$

Then $\|\mathrm{H}\| = \|f'\| = u_N v$ and H separates $\rho_1, \ldots, \rho_k$ by margin γ under y, since $\mathrm{tr}[\mathrm{H}\rho_i] = \langle f', e_i \rangle$ for $i = 1, \ldots, N$. Therefore, if $\mathcal{F}$ γ-shatters $\phi_1, \ldots, \phi_k$, then $\mathcal{H}$ γ-shatters $\rho_1, \ldots, \rho_k$. Q.E.D.

Theorem 5.11. *The fat shattering dimension of the set $\mathcal{H}$ of n-qubit observables* H *with $\|\mathrm{H}\| \geq \nu$ shattering n-qubit pure states satisfies*

$$\mathrm{fat}_{\mathcal{H}}(\gamma) \geq \min\{\nu^2(1 - 2^{-n})/\gamma^2, 2^n\}. \tag{5.20}$$

Proof. The result as described by Burges [37, theorem 6] states that the set $\mathcal{F}$ of functions in $\mathbb{R}^{N-1}$ with unit norm γ-shatters k vertices of a symmetric $(N-1)$-simplex, lying on a sphere of radius v, if $v^2/\gamma^2 \geq k-1$. Thus the set $\mathcal{F}'$ of functions with norm v γ-shatters k unit-norm vertices of a symmetric $(N-1)$-simplex under the same condition. Therefore, by lemma 5.10, k orthogonal n-qubit pure states are γ-shattered by the set $\tilde{\mathcal{H}}$ of n-qubit observables with norm $\nu = u_N v$ if $\nu^2/u_N^2\gamma^2 \geq k-1$; in other words, the maximum number of pure states that can be γ-shattered by $\tilde{\mathcal{H}}$ can be written $\lfloor \nu^2/u_N^2\gamma^2 \rfloor + 1$. Since we can have at most $N = 2^n$ orthogonal n-qubit pure states, we find that the fat shattering dimension of $\tilde{\mathcal{H}}$ is lower bounded by

$$\mathrm{fat}_{\tilde{\mathcal{H}}}(\gamma) \geq \min\{\nu^2(1-2^{-n})/\gamma^2, 2^n\}. \tag{5.21}$$

Since allowing observables with norm larger than ν does not decrease the fat shattering dimension (the enlarged set still contains the observables of norm ν that realise a separation), this directly implies the theorem. Q.E.D.

We can also write this bound in the fashion of eq. 5.17. This time, we note that $\|\mathrm{H}\| \geq r\lambda_{\min>0}^2$, where $\lambda_{\min>0}$ is the smallest nonzero absolute eigenvalue, and hence

$$\mathrm{fat}_{\mathcal{H}}(\gamma) \geq \min\{r\lambda_{\min>0}^2(1-2^{-n})/\gamma^2, 2^n\}. \tag{5.22}$$

What exactly can be learnt from these bounds? Firstly, we note that the set of unrestricted quantum CLF classifiers is very expressive, in that it can shatter exponentially many states in the number of qubits. After all, the rank of general n-qubit observables may scale as $O(2^n)$. The other side of the coin is that, at first sight, this also seems to imply poor generalisation performance: looking at expression 4.35, we obtain a bound scaling exponentially in n when we insert the lower bound on the fat shattering dimension of theorem 5.11 (since any training set must be of size $O(\mathrm{poly}(n))$). In this sense however, quantum CLF classifiers are no worse than classical CLF classifiers that operate in spaces of exponential dimensionality; as such, we should look deeper to find a distinction between classical and quantum CLF classifiers in terms of generalisation.

Let us have a closer look at theorem 4.19. It states, in the quantum setting, that if some observable H from a family $\mathcal{H}$ classifies a set of T points correctly with margin γ, then we obtain an upper bound for the generalisation risk through eq. 4.35 using this particular γ and the fat shattering dimension of $\mathcal{H}$. Note that the requirement to classify a set of T points is much less strict than the requirement to shatter these points: correct classification demands

a separation for only one, fixed label assignment, while shattering demands a separation for all possible label assignments. We should therefore relax the condition of shattering to compare generalisation performance of quantum and classical CLF learning; this allows us to cleverly pick a useful training set *and* a label assignment for which one can construct a separation such that the ratio between the fat shattering dimension and the training set size is more favourable.

The question then becomes whether quantum CLF learning offers a higher degree of freedom in the construction of such a separation, and therefore a higher likelihood of low generalisation risk. There are two indications that this is indeed the case. The first of these is that, besides free scaling of the observable which one can also apply in classical CLF learning, quantum observables have another property which we can use as a regularisation parameter: the rank. It is a free parameter, since it is part of the choice of the observable family, and it directly affects the fat shattering dimension as can be seen from eqs. 5.17 and 5.22. Interestingly, we can produce some intuition that a higher rank can lead to a lower margin. Namely, consider the restriction of observables to those of unit rank and norm ν. Imagine we are given T feature states $\rho_\Phi(x)$; in the worst case, these states may be (approximately) orthogonal. For orthogonal states, the maximum margin γ achieved is ν/T: we can see this by noting that the maximum margin is the maximum smallest inner product $\mathrm{tr}[\mathrm{H}\rho_\Phi(x)]$ with any feature state $\rho_\Phi(x)$, which is found when

$$\mathrm{H} = \nu|\mathcal{T}_+\rangle\langle\mathcal{T}_+| \tag{5.23}$$

with

$$|\mathcal{T}_+\rangle = T^{-1/2}\sum_{x\in\mathcal{T}_+}|\Phi(x)\rangle \tag{5.24}$$

where $\mathcal{T}_+ = \{(x,y)\in\mathcal{T} : y = +1\}$. Secondly, consider observables with rank at most T (and norm ν). Again, the T states which can be shattered with maximum margin might be mutually orthogonal. Now for any label assignment – i.e. every partition of the training set $\mathcal{T}$ into the $+1$ class $\mathcal{T}_+$ and the -1 class $\mathcal{T}_-$ – there is a straightforward choice for an observable which correctly classifies these points, namely

$$\mathrm{H} = \nu\Bigg(\sum_{x\in\mathcal{T}_+}\rho_\Phi(x) - \sum_{x'\in\mathcal{T}_-}\rho_\Phi(x')\Bigg). \tag{5.25}$$

Because the states $\rho_\Phi(x)$ are orthogonal, this observable classifies the training set with margin 2ν, which is an improvement by a factor $2T$. For this case,

an increase in rank resulted in an increase in margin; however it remains to be determined whether this also applies to observables of general rank. Furthermore, in order to assess the value of quantum CLF classifiers in this regard, one needs to know the precise relationship between the rank and the achievable margin – that is, whether down the line a higher rank contributes to a lower generalisation risk bound or not. This question we leave open for future investigation.

The second indication that quantum CLF classifiers may exhibit better generalisation performance is that likely, quantum computers offer more freedom in the choice of the feature map for which the computation of the classifier is tractable. So far, we have disregarded the feature map in the study of generalisation performance, considering only the most general arrangements of points in quantum Hilbert space. But when our task is to classify a set of points with a single label assignment, the feature map becomes crucial to arrange the points in such a way that a separation can be made with a high margin on many points. Here, a quantum computer could offer an advantage: while it is known that quantum computers can mimic all classical computations (as we will discuss in the next chapter), and therefore a quantum computer can reproduce all classical feature maps, the discorvery of quantum speedups [5, 9] suggests that there may be additional feature maps for which a quantum computer can efficiently compute expectation values. For these reasons, quantum feature maps have become a focal point for the investigation of quantum learning advantage [48]. Still, little is known about how such classically intractable feature maps influence generalisation performance under regularisation conditions, and we leave this for future work. For now, we shall focus our attention on the precise description of classical hardness in the quantum CLF learning setting, which we study in the next chapter.

Chapter 6

A path to quantum advantage

In this chapter, we consider the possibility for quantum advantage in learning using quantum CLF classifiers in place of their classical counterparts, elaborating on the claim by Havlíček et al. that an advantage could be provided by quantum feature maps with induced kernels that are hard to estimate. To this end, we first set up a framework of quantum complexity theory (section 6.1); with these notions laid out, the first step in the discussion of advantage is to give a rigorous definition of computational hardness for quantum classification functions (section 6.2). From such a definition, it will turn out straightforward to formally prove the equivalence between computational hardness of evaluating quantum classification functions (both explicit and implicit), and hardness of computing the predicted label from corresponding classifiers. We subsequently show that hardness of the kernel is sufficient to guarantee hardness of quantum CLF classifiers (section 6.3), thus verifying the claim. In section 6.4, we connect this result to previous quantum complexity considerations, and discuss possible hardness of learning procedures involving quantum classifiers, which is a different matter – and in fact a more general question – than mere evaluation of quantum classifiers.

6.1 Quantum complexity

In chapter 2, we precisely established a quantum computational model: given some input state, a quantum computer capable of implementing quantum gates performs operations on the input state and returns an output upon measurement. This is similar to a Turing machine – the most common theoretical model used to describe computing machines –, which is given a bit string as input, performs a number of operations on the input, and either

accepts or rejects. The capabilities of computing machines, then, are defined by the decision problems they can solve; a set of such problems is called a *complexity class*. For example, the set of decision problems that can be solved – more precisely, the languages $L \subseteq \{0,1\}^*$ that can be decided – by a deterministic Turing machine in time polynomial in the size of the input, is denoted P. It is appropriate, in order to compare the capabilities of classical and quantum computers, to consider similar concepts for quantum computers. The most common complexity class used to describe quantum computers is the class BQP. It is the set of languages that can be decided probabilistically (as a quantum computer is an inherently probabilistic machine) within bounded error, using polynomial-time quantum circuits.

Definition 6.1. A uniform family of polynomial-time quantum circuits is a set of quantum circuits $\{\mathrm{U}_n\}$, such that U_n acts on n qubits, and there exists a (classical) Turing machine which on all inputs $n \geq 1$ computes a description of U_n in time polynomial in n.

Definition 6.2. BQP is the set of languages $L \subseteq \{0,1\}^*$ for which there exists a uniform family of polynomial-time quantum circuits $\{\mathrm{U}_n\}$ such that for all $z \in \{0,1\}^n$,

$$\begin{cases} |\langle z|\mathrm{U}_n|1\rangle|^2 \geq 1/2 + \delta & \text{if } z \in L, \\ |\langle z|\mathrm{U}_n|1\rangle|^2 \leq 1/2 - \delta & \text{if } z \notin L. \end{cases} \tag{6.1}$$

for some $0 < \delta \leq 1/2$ with $\delta = \Omega(1/\mathrm{poly}(n))$, where $|1\rangle$ denotes the 1 state on the first qubit.

We can alter this definition a bit: note that we can write $|z\rangle = \mathrm{U}_z|0^n\rangle$, where U_z consists of Pauli X gates applied to the i-th qubit if $z_i = 1$. Since the circuit implementing U_z can be constructed trivially in time $O(n)$, we may absorb U_z into U_n, and the resulting set $\{\mathrm{U}_n\}$ remains a uniform poly-time family, for any $z \in \{0,1\}^n$. This, then, changes the language in question: we're now deciding languages over the set of uniformly poly-time quantum circuits[1] U_n – or rather, the bitstrings that describe these circuits – instead of bitstrings $z \in \{0,1\}^n$. Hence we may regard BQP as the set of quantum circuit languages L such that

$$\begin{cases} |\langle 0^n|\mathrm{U}_n|1\rangle|^2 \geq 1/2 + \delta & \text{if } \mathrm{U}_n \in L, \\ |\langle 0^n|\mathrm{U}_n|1\rangle|^2 \leq 1/2 - \delta & \text{if } \mathrm{U}_n \notin L. \end{cases} \tag{6.2}$$

[1]Here we mean that if we take one circuit U_n from this set for every size n, then the resulting set $\{\mathrm{U}_n\}_{n\geq 1}$ must be a uniform family of poly-time circuits.

Note that, strictly speaking, we have defined a *promise class*, since we explicitly excluded circuits for which $1/2 - \delta < |\langle 0^n|\mathrm{U}_n|1\rangle|^2 < 1/2 + \delta$. To reflect this, the class is sometimes (more appropriately) referred to as PromiseBQP. We shall however refer to it as simply BQP by abuse of notation.

Now, since our quantum computer is probabilistic in nature, this decision problem generally cannot be solved with certainty; the best we can do is sample from the distribution given by $|\langle 0^n|\mathrm{U}_n|1\rangle|^2$, and output the majority answer. The probability, then, that the output is the correct answer, can be found using the following theorem.

Theorem 6.3 (Chernoff-Hoeffding bound)**.** *Consider a set of K independent random variables $\{X_1, \ldots, X_K\}$. If we know $X_i \in [a, b]$ then let $\Delta = b - a$. Let $M = K^{-1}\sum_{i=1}^{K} X_i$. Then*

$$\mathbb{P}[|M - \mathbb{E}(M)| \leq \epsilon] \geq 1 - 2\exp\left(-\frac{2K\epsilon^2}{\Delta^2}\right). \tag{6.3}$$

We say that an expectation $\mathbb{E}(M)$ is *estimated to additive error* ϵ if we can sample M such that $|M - \mathbb{E}(M)| \leq \epsilon$.

Now, let X_i be the outcome after the i-th time we run the quantum circuit and measure the output qubit; if we are promised that either $|\langle 0^n|\mathrm{U}_n|1\rangle|^2 \geq 1/2 + \delta$ or $|\langle 0^n|\mathrm{U}_n|1\rangle|^2 \leq 1/2 - \delta$, then the majority answer will be correct if $|M - \mathbb{E}(M)| \leq \delta$. Hence, if we take $K = \Omega(1/\delta^2)$ (note that $\Delta = 1$ since the outcome is either 0 or 1), we can be certain to obtain a correct answer with at least constant probability (with respect to δ)[2]. Therefore, we shall express BQP as the set of decision problems as in eq. 6.2 which can be decided with at least constant probability in time $O(\mathrm{poly}(n, 1/\delta)) = O(\mathrm{poly}(n))$.

From the description of quantum circuits, we have that P $\subseteq$ BQP: after all, one can construct the so-called *Toffoli gate*,

$$|x\ y\ z\rangle \mapsto |x\ y\ \ z \oplus (x \wedge y)\rangle \tag{6.4}$$

from a constant number of single-qubit unitaries and CNOT gates [49]; by setting $z = 1$, one obtains a NAND gate, which is universal for classical computation. Therefore, by restricting itself to computational basis states (giving measurement probabilities of either 0 or 1, thus becoming deterministic), a quantum computer can mimic all classical computations.

Besides P, there is a class of problems that can be viewed as the classical counterpart to BQP. It is called BPP, and it contains problems of the form

[2]That is, if we take $K \geq c/\delta^2$ for some constant c, we can make the probability of error arbitrarily small by choosing c arbitrarily large.

in definition 6.2, except the outcomes must be generated by a classical probabilistic Turing machine. By probabilistic we mean that the machine has access to a "coin flipper" at any point in the computation. While it is easy to see that $\mathsf{P} \subseteq \mathsf{BPP}$, the converse is unknown; nonetheless, it is assumed to be true, and this assumption is strengthened by the fact that the construction of (a specific kind of) pseudorandom generators is sufficient to show $\mathsf{P} = \mathsf{BPP}$ [50]. In any case, it turns out that a quantum computer can solve all problems in BPP: simply put, every time the probabilistic classical Turing machine flips a coin, the quantum computer uses a Hadamard gate to create randomness. Therefore we have $\mathsf{P} \subseteq \mathsf{BPP} \subseteq \mathsf{BQP}$. Again, it is unknown whether $\mathsf{BQP} \subseteq \mathsf{BPP}$; however, since the discovery of quantum algorithms with no known classical (probabilistic) counterpart, such as Shor's factoring algorithm [5], it is widely believed that this is not the case. Throughout the rest of this chapter, we shall assume that $\mathsf{BQP} \nsubseteq \mathsf{BPP}$.

As discussed in the previous chapters, though, we are interested in computing expectation values, which are real numbers. While we have so far assumed the computation of expectation values to be a naturally feasible task for quantum computers, we have never shown formally that this is indeed the case; doing so will be the objective of the remainder of this section. Besides tying loose ends together, this analysis will also ease the discussion of computational hardness in the context of quantum learning.

When we speak of computing expectation values, we wish to obtain the value $\langle\psi|\mathrm{O}|\psi\rangle$ given an observable O and a state $|\psi\rangle$. However, just as with BQP problems, we cannot expect to do this for *any* O and $|\psi\rangle$; like unitary circuits, there must be a description that is uniformly poly-time (in the number of qubits) for both. To make more sense of this, let us write down the usual decomposition, where $|\psi\rangle$ is prepared from $|0\rangle$ by a unitary U and O eigendecomposes as $\mathrm{W}^\dagger\mathrm{D}\mathrm{W}$, in the computational basis:

$$\langle\psi|\mathrm{O}|\psi\rangle = \langle 0|\mathrm{U}^\dagger\mathrm{W}^\dagger\mathrm{D}\mathrm{W}\mathrm{U}|0\rangle. \tag{6.5}$$

We see that at the very least, both U and W must be taken from a uniform family of poly-time quantum circuits for the expectation value $\langle\psi|\mathrm{O}|\psi\rangle$ to be efficiently computable on a quantum computer. But there is another point: since the outputs from our quantum circuits are computational basis states, we need to be able to efficiently map these states to their respective eigenvalues, in order to compute any expectation value. These two requirements lead to the definition of uniformly poly-time observables.

Definition 6.4. A uniform family of polynomial-time observables is a set of n-qubit observables $\{\mathrm{O}_n\}$ with eigendecompositions $\mathrm{O}_n = \mathrm{U}_n\mathrm{D}_n\mathrm{U}_n^\dagger$ such that

(1) there exists a deterministic Turing machine which on all inputs $n \geq 1$ computes a description of U_n in time $O(\mathrm{poly}(n))$; and

(2) the function λ_n which maps $z \in \{0,1\}^n$ to the z-th diagonal element of D_n (i.e. the z-th eigenvalue of O_n in the computational basis) is computable in time $O(\mathrm{poly}(n))$ on all inputs $z \in \{0,1\}^n$.

For convenience, we have absorbed W into U as both are unitary and necessarily uniformly poly-time circuits.

In order to connect the task of computing expectation values to the quantum complexity class BQP, we define a class of decision problems called "expectation-BQP", denoted EBQP, which we shall express in terms of languages over observables in analogy to eq. 6.2.

Definition 6.5. EBQP is the set of all languages L over the set of uniformly poly-time observables such that

$$\begin{cases} \langle 0^n|\mathrm{O}_n|0^n\rangle \geq +\delta & \text{if } \mathrm{O}_n \in L, \\ \langle 0^n|\mathrm{O}_n|0^n\rangle \leq -\delta & \text{if } \mathrm{O}_n \notin L \end{cases} \tag{6.6}$$

for some strictly positive $\delta = \Omega(\Delta_{\mathrm{O}_n}/\mathrm{poly}(n))$, where Δ_{O_n} is the difference between the largest and smallest eigenvalue of O_n.

The intuition for this restriction on δ stems from the fact that, considering the Chernoff-Hoeffding bound, the allowed error should scale at least with the outcome gap Δ in order to achieve constant success probability with sufficiently few measurements. (Note in the definition of BQP, we imposed this restriction with $\Delta = 1$.)

Let us now consider the relationship between BQP and EBQP. First, let $L \in \mathsf{BQP}$, let $\{\mathrm{U}_n\}$ be as in eq. 6.2, and identify

$$\mathrm{O}_n = \mathrm{U}_n|1\rangle\langle 1|\mathrm{U}_n^\dagger - 1/2; \tag{6.7}$$

then

$$|\langle 0^n|\mathrm{U}_n|1\rangle|^2 = 1/2 + \langle 0^n|\mathrm{O}_n|0^n\rangle. \tag{6.8}$$

Thus, if we define a language L' as in eq. 6.6 with the same δ, then $\mathrm{O}_n \in L'$ if and only if $\mathrm{U}_n \in L$. Note that $\{\mathrm{O}_n\}$ is a uniform family of poly-time observables since $\{\mathrm{U}_n\}$ is a uniform family of poly-time circuits, and the eigenvalue function λ_n corresponding to the diagonal operator $|1\rangle\langle 1| - 1/2$ can be computed in time $O(1)$. Since $\lambda_n(z) \in [-1/2, 1/2]$, we have $\Delta_{\mathrm{O}_n} = 1$, and thus $\delta = \Omega(\Delta_{\mathrm{O}_n}/\mathrm{poly}(n))$, which implies that $L' \in \mathsf{EBQP}$. Hence L is

decided in polynomial time by a machine that decides all languages in EBQP, and therefore BQP $\subseteq$ EBQP.

We can reverse the argument. Let $\{\mathrm{O}_n\}$ be as in eq. 6.6, and let $L \in$ EBQP. Note that any poly-time observable O_n can be expressed in its eigendecomposition

$$\mathrm{O}_n = \sum_{z\in\{0,1\}^n} \lambda_z \mathrm{U}_n|z\rangle\langle z|\mathrm{U}_n^\dagger \tag{6.9}$$

where U_n is a uniformly poly-time quantum circuit and λ_z is computable from z in polynomial time. This means that

$$\langle 0^n|\mathrm{O}_n|0^n\rangle = \sum_{z\in\{0,1\}^n} \lambda_z\langle 0^n|\mathrm{U}_n|z\rangle\langle z|\mathrm{U}_n^\dagger|0^n\rangle = \sum_{z\in\{0,1\}^n} \lambda_z|\langle 0^n|\mathrm{U}_n|z\rangle|^2. \tag{6.10}$$

Because U_n is uniformly poly-time, we can use a BQP machine (a machine that decides all languages in BQP, i.e. a quantum computer) to sample from the distribution

$$\mathbb{P}(z) = |\langle 0^n|\mathrm{U}_n|z\rangle|^2 = \langle 0^n|\mathrm{U}_n|z\rangle\langle z|\mathrm{U}_n^\dagger|0^n\rangle \tag{6.11}$$

by measuring every qubit (which requires time $O(n)$). Then, we can compute an estimation of the expectation value $\langle 0^n|\mathrm{O}_n|0^n\rangle$ by sampling many multi-qubit outcomes, computing the eigenvalue corresponding to the outcome and averaging the result. According to the Chernoff-Hoeffding bound, and the fact that the eigenvalues $\lambda_n(z)$ can be computed in time $O(\mathrm{poly}(n))$, this estimate can be made up to additive error at most δ with at least constant probability in time $O(\mathrm{poly}(n, \Delta_{\mathrm{O}_n}/\delta))$; since $L \in$ EBQP requires that $\delta = \Omega(\Delta_{\mathrm{O}_n}/\mathrm{poly}(n))$, the time needed is $O(\mathrm{poly}(n))$. Given the promise that the expectation value be either above $+\delta$ or below $-\delta$, L is thus decided by the BQP machine with at least constant probability in polynomial time by accepting if the result is positive, and rejecting if it is negative. As such, $L \in$ BQP, which implies that EBQP $\subseteq$ BQP; combined with the other direction, we find that EBQP = BQP.

From the above reasoning, it is clear that all problems which ask to estimate expectation values up to some additive error in polynomial time, without deciding whether they fall above or below some threshold, can be solved by a machine that solves all problems in BQP. After all, given a procedure that solves a decision problem as in eq. 6.6, a classical deterministic machine can compute the expectation value $\langle 0^n|\mathrm{O}_n|0^n\rangle$ through binary search with an initial guess g by invoking this procedure $O(\log(|g - \langle 0^n|\mathrm{O}_n|0^n\rangle|/\delta))$ times; and since all numbers used in the computation must be stored in

at most $O(\mathrm{poly}(n))$ bits for the computation to be feasible in polynomial time, it is guaranteed that $\log(|g - \langle 0^n|\mathrm{O}_n|0^n\rangle|/\delta) = \log(2^{O(\mathrm{poly}(n))}/\delta) = O(\mathrm{poly}(n) + \log(1/\delta)) = O(\mathrm{poly}(n, 1/\delta))$. Hence, a deterministic Turing machine that has oracle access to a machine that solves all BQP problems, also solves all such estimation problems. In the following, we shall make a slight abuse of notation, and consider BQP in the sense of real-valued estimation problems instead of decision problems.

6.2 Defining computational hardness

We shall now proceed to give appropriate definitions of computational hardness for quantum CLF classification functions and classifiers. To this end, we shall firstly revisit the construction of quantum CLF classifiers as defined in chapter 5, in order to better fit them into the context of quantum complexity as discussed in the previous section, which is required to properly define computational hardness.

Let us begin with quantum explicit classification functions. We shall formally consider such functions in terms of *classification function descriptions*, one for each qubit number n. We define a description $\mathcal{F}_n$ on n qubits to be a 4-tuple $(\phi, \mathcal{U}, \mathcal{W}, \mathrm{D})$ containing

- a feature angle vector map $\phi : \mathcal{X} \to [0, 2\pi]^m : x \mapsto \boldsymbol{\phi}(x)$,
- a feature unitary operator map $\mathcal{U} : [0, 2\pi]^m \to U(2^n) : \boldsymbol{\phi} \mapsto \mathrm{U}(\boldsymbol{\phi})$,
- a variational unitary operator map $\mathcal{W} : [0, 2\pi]^p \to U(2^n) : \boldsymbol{\theta} \mapsto \mathrm{W}(\boldsymbol{\theta})$,
- a real computational basis diagonal observable $\mathrm{D} \in H(2^n)$.

We demand that the set $\{\mathcal{F}_n\}_{n\geq 1}$ be uniformly poly-time computable, i.e. there must exist a Turing machine which computes the maps ϕ, $\mathcal{U}$ and $\mathcal{W}$ on input n and the respective map inputs in time $O(\mathrm{poly}(n))$. Furthermore, we require that the diagonal entries of D be computable in time $O(\mathrm{poly}(n))$. Then, a *classification function instance* $f_{\boldsymbol{\theta}}(\cdot)$ of description $\mathcal{F}_n$ is a function

$$f_{\boldsymbol{\theta}}(x) = \langle 0^n|\mathrm{U}^\dagger_{\boldsymbol{\phi}(x)}\mathrm{W}^\dagger_{\boldsymbol{\theta}}\,\mathrm{D}\,\mathrm{W}_{\boldsymbol{\theta}}\mathrm{U}_{\boldsymbol{\phi}(x)}|0^n\rangle \tag{6.12}$$

such that $\boldsymbol{\phi}(x) = \phi(x)$, $\boldsymbol{\theta} \in [0, 2\pi]^p$, $\mathrm{U}_{\boldsymbol{\phi}} = \mathcal{U}(\boldsymbol{\phi})$ and $\mathrm{W}_{\boldsymbol{\theta}} = \mathcal{W}(\boldsymbol{\theta})$. Since $\{\mathcal{F}_n\}_{n\geq 1}$ is uniformly poly-time computable, the problem of approximately evaluating $f_{\boldsymbol{\theta}}(x)$ for *any* x and $\boldsymbol{\theta}$ is in BQP.

In order to be able to think about classical hardness of producing such estimates, we first need to define precisely a *family* of classification functions.

Definition 6.6. Let $\mathcal{F}_n$ be a quantum explicit classification function description for all $n \geq 1$, and let F_n be the set of all classification function instances of $\mathcal{F}_n$. Then a *family* of quantum explicit classification functions is the set $\{F_n\}_{n\geq 1}$.

We shall now define classical hardness of families of explicit classification functions.

Definition 6.7. A family $G = \{F_n\}$ of quantum explicit classification functions described by descriptions $\{\mathcal{F}_n\}$ is *classically hard* if there exists no classical algorithm which, for all $n \geq 1$, can evaluate all classification function instances $f(\cdot) \in F_n$ on all inputs $x \in \mathcal{X}$ up to additive error ϵ in time $O(\text{poly}(n, \Delta_{\text{D}}, 1/\epsilon))$.

In other words, a quantum explicit classification function family G is classically hard if the problem of estimating *any* evaluation of a function in G is in $\mathsf{BQP} \setminus \mathsf{BPP}$.

For the hardness definition of implicit classifiers, we can make an almost carbon copy of the definitions for explicit classifiers; the main difference is the classification function description. In particular, we define a quantum implicit classification function description $\mathcal{F}_n$ on n qubits to be a 4-tuple $(\phi, \mathcal{U}, \mathcal{T}, \boldsymbol{\alpha})$, containing

- a feature angle vector map $\phi : \mathcal{X} \to [0, 2\pi]^m : x \mapsto \boldsymbol{\phi}(x)$,
- a feature unitary operator map $\mathcal{U} : [0, 2\pi]^m \to U(2^n) : \boldsymbol{\phi} \mapsto \text{U}(\boldsymbol{\phi})$,
- a training set $\mathcal{T} \subseteq \mathcal{X} \times \mathcal{Y}$ of cardinality $O(\text{poly}(n))$,
- a weight vector $\boldsymbol{\alpha} \in \mathbb{R}^{|\mathcal{T}|}$.

which again must be uniformly poly-time computable.

Then, a classification function instance $f(\cdot)$ of description $\mathcal{F}_n$ is a function

$$f(x) = \sum_{x' \in \mathcal{T}} \alpha_{x'} \, |\langle 0^n | \text{U}^\dagger_{\boldsymbol{\phi}(x')} \text{U}_{\boldsymbol{\phi}(x)} | 0^n \rangle|^2 \tag{6.13}$$

$$= \langle 0^n | \text{U}^\dagger_{\boldsymbol{\phi}(x)} \text{O}_{\mathcal{T},\boldsymbol{\alpha}} \text{U}_{\boldsymbol{\phi}(x)} | 0^n \rangle \tag{6.14}$$

where

$$\text{O}_{\mathcal{T},\boldsymbol{\alpha}} = \sum_{x' \in \mathcal{T}} \alpha_{x'} \text{U}_{\boldsymbol{\phi}(x')} | 0^n \rangle \langle 0^n | \text{U}^\dagger_{\boldsymbol{\phi}(x')}. \tag{6.15}$$

such that $\boldsymbol{\phi}(x) = \phi(x)$ and $\text{U}_{\boldsymbol{\phi}} = \mathcal{U}(\boldsymbol{\phi})$.

Note that the gap Δ appearing in definition 6.7 is given for implicit classifiers, depending on the training set feature maps $\mathrm{U}_{\phi(x')}$, by

$$\Delta_{\mathrm{O}_{\mathcal{T},\boldsymbol{\alpha}}} \leq \alpha_+ - \alpha_- = \sum_{x' \in \mathcal{T}} |\alpha_{x'}|, \tag{6.16}$$

where α_+ is the sum of all positive $\alpha_{x'}$, and α_- is the sum of all negative $\alpha_{x'}$. The extreme case $\Delta_{\mathrm{O}_{\mathcal{T},\boldsymbol{\alpha}}} = \alpha_+ - \alpha_-$ occurs when, for all states $\mathrm{U}_{\phi(x')}|0^n\rangle$,

$$\begin{cases} |\langle 0^n|\mathrm{U}_{\phi(x'_i)}\mathrm{U}_{\phi(x'_j)}|0^n\rangle|^2 = 1 & \text{if } \operatorname{sgn}(\alpha_{x'_i}) = \operatorname{sgn}(\alpha_{x'_j}), \\ |\langle 0^n|\mathrm{U}_{\phi(x'_i)}\mathrm{U}_{\phi(x'_j)}|0^n\rangle|^2 = 0 & \text{if } \operatorname{sgn}(\alpha_{x'_i}) \neq \operatorname{sgn}(\alpha_{x'_j}), \end{cases} \tag{6.17}$$

since in this case the observable $\mathrm{O}_{\mathcal{T},\boldsymbol{\alpha}}$ becomes

$$\mathrm{O}_{\mathcal{T},\boldsymbol{\alpha}} = \alpha_+ \mathrm{U}_+|0^n\rangle\langle 0^n|\mathrm{U}_+^\dagger + \alpha_- \mathrm{U}_-|0^n\rangle\langle 0^n|\mathrm{U}_-^\dagger \tag{6.18}$$

for some $\mathrm{U}_+, \mathrm{U}_-$ such that $\mathrm{U}_+|0^n\rangle$ and $\mathrm{U}_-|0^n\rangle$ are orthogonal. In all other cases, nonorthogonality between the states $\mathrm{U}_{\phi(x')}|0^n\rangle$ implies a smaller gap $\Delta_{\mathrm{O}_{\mathcal{T},\boldsymbol{\alpha}}}$. Further note that, if we first estimate the kernels $|\langle 0^n|\mathrm{U}^\dagger_{\boldsymbol{\phi}(x')}\mathrm{U}_{\boldsymbol{\phi}(x)}|0^n\rangle|^2$ to some additive error $\tilde{\epsilon}$ and subsequently compute the weighted sum (eq. 6.13), the resulting error is bounded by $\tilde{\epsilon}\Delta_{\mathrm{O}_{\mathcal{T},\boldsymbol{\alpha}}}$; as such we can estimate $f(x)$ to additive error ϵ by estimating all individual kernels to error $\tilde{\epsilon} = \epsilon/\Delta_{\mathrm{O}_{\mathcal{T},\boldsymbol{\alpha}}}$, which requires $O(\mathrm{poly}(\Delta_{\mathrm{O}_{\mathcal{T},\boldsymbol{\alpha}}}, 1/\epsilon))$ measurements per kernel. This is precisely in line with the fact that, provided ϕ and $\mathcal{U}$ are uniformly poly-time computable, $\mathrm{O}_{\mathcal{T},\boldsymbol{\alpha}}$ is a uniformly poly-time observable (and so is $\mathrm{U}^\dagger_{\boldsymbol{\phi}(x)}\mathrm{O}_{\mathcal{T},\boldsymbol{\alpha}}\mathrm{U}_{\boldsymbol{\phi}(x)}$), which assures that the problem of estimating $f(x)$ is in BQP.

A family of quantum implicit classification functions is defined in a similar fashion to families of quantum explicit classifiation functions, only using an implicit classification description. This straightforwardly gives a definition of classical hardness with regard to implicit classification functions: like in the explicit case, a quantum implicit family G is classically hard if the task of estimating any evaluation of a function in G is in $\mathsf{BQP} \setminus \mathsf{BPP}$.

Now, in order to obtain a classification result from the evaluation of a classification function, we require a function σ which takes the output of the classification function and maps it onto one of the labels -1 or $+1$; the composition $\sigma \circ f$ then forms the classifier from a classification function. We require that σ be efficiently computable on a classical machine. In accordance with the previous chapters, we consider threshold functions of the form

$$\sigma(f(x)) = \operatorname{sgn}(f(x) - d) \tag{6.19}$$

for some offset $d \in \mathbb{R}$, which are clearly efficiently computable classically. This modification from a real-valued to a binary output changes the nature of the problem: instead of an estimation problem, we are now concerned with a decision problem. For this reason, it is in order to give separate definitions of classifier families and classifier hardness. After all, while the two types of problems are closely related, they are not equivalent.

Definition 6.8. Given a (either explicit or implicit) classification function family $G = \{F_n\}_{n\geq 1}$, a classifier family corresponding to G is a family of pairs $C = \{(F_n, \sigma_n)\}_{n\geq 1}$, where σ_n is a threshold function with some offset $d \in \mathbb{R}$.

It is necessary to specify a separate threshold function for every n, since definition 6.6 (and similarly for implicit classification functions) allows different codomains of the functions $f(\cdot) \in F_n$ for different n. After all, if the offset d lies outside the codomain of a function $f(\cdot)$ (i.e. $d > f(x)\ \forall x$ or $d < f(x)\ \forall x$), the decision problem of computing $\sigma(f(x))$ becomes trivial: the answer is either $+1$ or -1, for all x. For this reason, we shall call a classifier family *well-formed* if, for all n, the offset d of σ_n lies in the codomain of the functions $f(\cdot) \in F_n$.

The intuition for the definition of classifier hardness stems from the (by now well-known) process for obtaining a classification result: a quantum computer computes an estimate of an expectation value up to some error and feeds this into the threshold function σ. For a classifier family to be classically hard, there should exist no classical algorithm which can simulate this entire process efficiently for all inputs x and classification function $f(\cdot)$. We formalise this in the following definition.

Definition 6.9. Let C be a classifier family corresponding to classification function family $G = \{F_n\}$. Then C is *classically hard* if there exists no classical algorithm which, for all $n \geq 1$ and $f(\cdot) \in F_n$ with $f(x) \in [a, b]\ \forall x \in \mathcal{X}$, can compute $\sigma(\tilde{f}(x))$ such that $\tilde{f}(x) \in [a, b]$ and $|\tilde{f}(x) - f(x)| \leq \epsilon$ on all inputs $x \in \mathcal{X}$, in time $O(\text{poly}(n, \Delta, 1/\epsilon))$, where $\Delta = b - a$.

Given uniformly poly-time computability properties of the classification function description which gives rise to a classification function family G, we expect a quantum computer to be able to compute $\tilde{f}(x)$ in the sense of the above definition, and therefore to compute the classification result $\sigma(\tilde{f}(x))$. Therefore, the condition that C is classically hard again entails that the problem of computing $\sigma(\tilde{f}(x))$, for any x and f with respect to C, lies in $\mathsf{BQP} \setminus \mathsf{BPP}$. Note that, for the reasons discussed above, C must be well-formed in order to be classically hard.

6.3 Connecting classification functions, classifiers and kernels

With precise hardness definitions in place, we are now in a position to continue the analysis of quantum advantage in the context of quantum CLF classifiers, by first laying the connection between classification functions and their corresponding classifiers. Through the following theorem, we show that classical hardness of classification function families and classifier families is equivalent, which is a step usually glanced over in contemporary literature, including the work of Havlíček et al.

Theorem 6.10. *Let G be a family of explicit or implicit classification functions and C a well-formed family of classifiers corresponding to G. Then C is classically hard* if and only if *G is classically hard.*

Proof. If G is not classically hard, then we can use a classical computer to directly calculate any $f(x)$, to within ϵ additive error in time $O(\text{poly}(n, \Delta, 1/\epsilon))$ and use it as the input to σ. Since σ is classically efficiently computable, so is $\sigma(f(x))$ up to the same error, and hence C is not classically hard. Inversion of this statement yields the *only if* direction.

The other direction follows from the fact that any machine which can compute $\sigma(\tilde{f}(x))$ such that $|\tilde{f}(x) - f(x)| \leq \epsilon$ in time $O(\text{poly}(n, \Delta, 1/\epsilon))$, can decide the following comparison problem, given the threshold $d \in [a, b]$ of σ:

$$\begin{cases} \text{if } f(x) + \epsilon > d & \text{then YES} \\ \text{if } f(x) - \epsilon < d & \text{then NO} \end{cases} \tag{6.20}$$

If a classical machine can efficiently find a solution to the above decision problem for some $d \in [a, b]$, it can do so for all $d \in [a, b]$ simply by shifting $f(x)$. Therefore, one can then compute an estimate of $f(x)$ up to additive error ϵ through binary search, with at most $O(\log(\Delta/\epsilon))$ such comparisons[3]. Since $O(\text{poly}(n, \log(\Delta/\epsilon)))$ can be absorbed into $O(\text{poly}(n, \Delta, 1/\epsilon))$, we find that $f(x)$ can be efficiently classically estimated. In conclusion, we observe that G is not classically hard if C is not classically hard, and hence by inversion of this statement, we proved the *if* direction. Q.E.D.

Informally, this result states that if one cannot efficiently classically estimate certain quantum CLF classification functions up to some error, one

[3]If $f(x) \in [a, b]$ with $\Delta = b - a$, then in the worst-case, the initially picked d and $f(x)$ are at most Δ apart. Then, the number of comparisons K required to find $\tilde{f}(x)$ such that $|f(x) - \tilde{f}(x)| \leq \epsilon$ is found from the condition $2^{-K}\Delta \leq \epsilon$, which implies $K = O(\log(\Delta/\epsilon))$.

cannot classically efficiently compute the corresponding classifier up to that error either.

To show that classical hardness of estimating quantum kernels provides classical hardness of computing quantum CLF classifiers, we need to take one last step. Here, by kernel hardness we mean the existence of a classically hard family of quantum implicit classification functions with a single term (which are kernels multiplied by a scalar). Now, the above analysis only shows that kernel hardness implies hardness of implicit "classifiers" whose training set contains a single point; however, a problem with only one point to classify is not a well-defined classification problem. As such, we need to show that under the condition of quantum kernel hardness, there exist families of quantum implicit classifiers induced by training sets with at least two points, which are classically hard.

To this end, take a training set $\mathcal{T} = \{(x_0, +1), (x_1, -1)\}$ and an induced classifier

$$c(x) = \mathrm{sgn}(\alpha_0 \,\mathrm{tr}[\rho(x_0)\rho(x)] + \alpha_1 \,\mathrm{tr}[\rho(x_1)\rho(x)] - d). \tag{6.21}$$

For correct labelling, we require that $\alpha_0 > \alpha_1$ and $\alpha_1 < d < \alpha_0$. Note that the second requirement follows immediately if this family of classifiers is well-formed. If the requirements are met, there always exist inputs x such that either $\alpha_0 \,\mathrm{tr}[\rho(x_0)\rho(x)]$ or $\alpha_1 \,\mathrm{tr}[\rho(x_1)\rho(x)]$ equals d; this brings us back to the single-point case, which we assumed was hard. As such, the family containing classifiers of this type is classically hard if and only if the family of kernels $\mathrm{tr}[\rho(x_i)\rho(x)]$ is classically hard. Now, we point out that a similar argument can be given for classifiers induced by training sets with more than two points: it may still occur that all but one of the terms sum to d, which gives the same hardness property. Lastly noting that, as we argued in chapter 5, implicit classifiers are a special case of explicit classifiers, we can summarise the analysis in the following statement.

Corollary 6.11. *There exist families of quantum CLF classifiers (either explicit or implicit) that are classically hard, if and only if there exists a family of quantum kernels which cannot be evaluated efficiently classically.*

As a final observation, we remark that for a family of quantum explicit classifiers to be possibly classically hard contingent on kernel hardness, the observable $\mathrm{H}_{\boldsymbol{\theta}} = \mathrm{W}_{\boldsymbol{\theta}}^{\dagger}\mathrm{D}\mathrm{W}_{\boldsymbol{\theta}}$ must act on the space of the feature states for at least one $\boldsymbol{\theta}$, i.e. $\exists x \in \mathcal{X}, \boldsymbol{\theta} \in [0, 2\pi^m] : \langle\Phi(x)|\mathrm{H}_{\boldsymbol{\theta}}|\Phi(x)\rangle \neq 0$. But since this is required for the explicit classifier to be able to classify any set of feature vectors at all, this is not an issue.

6.4 Hardness of learning

Through the complexity theoretic study of the previous sections, we have worked out a possible source of quantum advantage in using quantum CLF classifiers: namely, if it can be shown that $\mathsf{BQP} \nsubseteq \mathsf{BPP}$, we can be certain that there is no classical algorithm which estimates all quantum kernels efficiently, and therefore it will be impossible to evaluate all quantum CLF classifiers efficiently on a classical machine. After all, if one can efficiently estimate the overlap between any two (efficiently preparable) states in a classical manner, from the descriptions of the circuits that generate them, then $\mathsf{BQP} = \mathsf{BPP}$. But is this really an advantage for learning? Or, in other words, does such "quantum advantage" provide any motivation for using quantum CLF classifiers at all? In machine learning, we're not interested in classifiers simply for the sake of evaluating them, but for the reason that they solve classification problems. As such, the question we should ask is: can quantum classifiers efficiently solve classification problems that classical classifiers cannot? Do there exist classification problems which are so difficult they force us to use a quantum classifier? Havlíček et al. do not explicitly consider this question in their paper, but given its importance in motivating the use of quantum classifiers, we will modestly discuss the topic in this section.

To be able to say anything meaningful on this matter, we first need to define precisely what we mean by a classification problem. Typically, such problems are described by the model of *probably approximately correct learning*, or PAC learning for short [51]. Let $\mathcal{X}$ be an input set and $\mathcal{Y} = \{+1, -1\}$ a label set. In the PAC learning context, a *concept class* $\mathfrak{Y}$ is a set of functions $y : \mathcal{X} \to \mathcal{Y}$, one of which is the unknown ground truth to be learnt. Then, a *learner* $\mathcal{L}$ for $\mathfrak{Y}$ is an algorithm which outputs a *hypothesis* $c : \mathcal{X} \to \mathcal{Y}$. This learner $\mathcal{L}$ is said to be a (ϵ, δ)-PAC learner for $\mathfrak{Y}$ if, for every $\epsilon, \delta \in [0, 1]$, $y(\cdot) \in \mathfrak{Y}$ and every distribution $\mathcal{D}$ over $\mathcal{X}$, when given a training set $\mathcal{T} \subset \mathcal{X} \times \mathcal{Y}$ with x drawn from $\mathcal{D}$ as input[4], $\mathcal{L}$ outputs with probability at least $1 - \delta$ a hypothesis $h(\cdot)$ such that

$$\mathbb{P}_{x \sim \mathcal{D}}[h(x) \neq y(x)] \leq \epsilon \tag{6.22}$$

with at most $O(\text{poly}(n, 1/\delta, 1/\epsilon))$ examples. Informally, a hypothesis that is the output of an (ϵ, δ)-PAC learner is allowed to err, as long as it does so sufficiently infrequently – preferably on points that occur with low probability.

This definition, however, is not compatible with quantum classification complexity as discussed in the previous sections, since we considered quantum

[4]Strictly speaking, $\mathcal{D}$ is an *ensemble* $\{D_n\}_{n \geq 1}$ of distributions over n-bit strings. As such, in eq. 6.27, x is drawn from the n-bit distribution D_n for every n.

classifiers in the context of worst-case complexity classes, and PAC learnability is not a worst-case statement. As such, let us proceed with a nonstandard, worst-case definition of learning: as a special case of PAC learning, we shall require a worst-case learner to output a hypothesis that classifies all new points correctly with certainty, i.e. $\epsilon, \delta = 0$, using $O(\text{poly}(n))$ examples drawn from any distribution $\mathcal{D}$.

When comparing worst-case computational capabilities of quantum and classical computers, one typically considers the complexity classes BQP and BPP, as we have done in section 6.1. However, because classification problems allow the use of training sets containing oracle-given examples, it is more intuitive, when asking how quantum and classical computers compare in terms of learning, to consider a complexity class that takes into account such examples. One such complexity is P/poly: this is the class of problems solvable by a classical machine, in polynomial time, which besides the input is given an $O(\text{poly}(n))$-sized *advice string* for each instance size n that may not depend on the input. For classification problems, we can identify the training set, which we require to be of polynomial size, as the advice. Here, it is worthwhile to note that, in fact, BPP/poly = P/poly. The direction P/poly $\subseteq$ BPP/poly is trivial; the other direction follows from a result known as Adleman's theorem [52, 53], which states that BPP $\subseteq$ P/poly. It is easy to see that a P/poly machine M which decides any language $L \in$ BPP also decides all languages $L' \in$ BPP/poly, because the advice string given to the BPP/poly machine M' that decides L' is of polynomial size, and can therefore be appended to the advice string given to M.

In order to compare quantum and classical worst-case learning, then, we should consider how BQP and P/poly relate. Of interest here are problems that are BQP-*complete*: these are the problems in BQP such that, if some machine M can decide one such problem, then a P machine with access to the oracle M can solve all problems in BQP. One such problem is the approximation of the Jones polynomial, which plays a big role in the simulation of topological quantum field theory [54–57]. More precisely, there exist uniform families of polynomial-time quantum circuits $\{\mathrm{U}_n\}$ (cf. definition 6.1), such that the problem of deciding whether $|\langle z|\mathrm{U}_n|1\rangle|^2$ is greater or smaller than $1/2 \pm \delta$ for all $z \in \{0,1\}^n$ is BQP-complete; therefore, if a classical polynomial-time algorithm can decide this problem with $O(\text{poly}(n))$-sized advice, it can do so for all uniform families of polynomial-time quantum circuits, and hence BQP $\subseteq$ P/poly[5].

[5] BQP $\subseteq$ P/poly is in fact believed not to hold. One argument is that it would imply Factoring $\in$ P/poly, and would thus break most cryptography schemes, which are defined against P/poly adversaries [58].

With this in mind, we can determine the relationship between BQP and P/poly on the condition that a classical computer is capable of learning everything (in the sense of the worst-case definition given above) that a quantum computer can learn. The latter is precisely the set of ground truths $y(\cdot) = c^n_{\boldsymbol{\theta}^*}(\cdot)$ where $c^n_{\boldsymbol{\theta}^*}(\cdot)$ is some quantum CLF classifier from a classifier family C, one for every qubit number n, such that the set comprising the underlying classification functions $\{f^n_{\boldsymbol{\theta}^*}(x)\}_{n\geq 1}$ can be realised as a uniform family of polynomial-time observables for every x. Indeed, if C is chosen arbitrarily from the set of all possible quantum CLF classifier families, this is the best a quantum computer can do: by theorem 6.10, evaluating a classifier $c(x)$ is precisely as hard as evaluating the underlying classification function $f(x)$; and since the choice of $c^n_{\boldsymbol{\theta}^*}(x)$ is arbitrary, there exist classifier choices such that estimating expectation values for the family $\{f^n_{\boldsymbol{\theta}^*}(x)\}$ is a BQP-complete problem. After all, we can identify $f^n_{\boldsymbol{\theta}^*}(x) = \langle x|\mathrm{U}_n|1\rangle\langle 1|\mathrm{U}_n^\dagger|x\rangle = |\langle x|\mathrm{U}_n|1\rangle|^2$, where $\{\mathrm{U}_n\}$ describes a BQP-complete quantum circuit family. Therefore, if there exists a classical poly-time algorithm which can worst-case learn anything that a quantum computer can possibly learn – that is, it can learn to simulate any poly-time quantum circuit without being given the circuit description – using polynomially many samples as advice, then $\mathsf{BQP} \subseteq \mathsf{P/poly}$.

To a careful reader however, worst-case classification of the concept containing all quantum circuits, using polynomially many examples drawn from any distribution, may seem like a very tall order. Is it possible at all? If not, then this invalidates the above reasoning, claiming quantum learning supremacy on the condition that $\mathsf{BQP} \nsubseteq \mathsf{P/poly}$. This question is answered by a result from the work of Arunachalam et al. [59]: namely, in their paper they show that, under the condition that the *learning with errors* problem LWE [60] is not in BQP, which is widely believed to be true, there exists no quantum polynomial-time PAC learner for the boolean circuit class TC^2_0.

Let us decompose this statement. First, TC^2_0 is the set of boolean functions $y : \{0,1\}^n \to \{0,1\}$ that can be computed by a circuit B which satisfies the following:

- B contains $O(\mathrm{poly}(n))$ gates;
- each gate is an unbounded fan-in NOT, AND, OR or majority gate MAJ, where $\mathrm{MAJ}(x) = 1$ if and only if $\sum_i x_i \geq n/2$;
- B is of depth at most 2.

Here, TC^2_0 is the concept class in the context of PAC learning. Furthermore, a quantum PAC learner, as defined in the paper, is a learner that has access

to a quantum computer, as well as a *quantum training set* of the form

$$\sum_x \sqrt{p(x)}\,|x, c(x)\rangle. \tag{6.23}$$

Then, the authors state that, conditioned on $\mathsf{LWE} \notin \mathsf{BQP}$, there exists no quantum learner which runs in time $O(\text{poly}(n))$ and is a PAC learner for TC^2_0. The consequence of this is that, under the made assumption, the concept class of all quantum circuits is not classically efficiently PAC learnable. After all, quantum learning with quantum examples is no harder than classical learning with classical examples, and TC^2_0 is a restricted class of *classical* circuits, which is no harder to learn than an unrestricted class of quantum circuits. Yet, there exists no poly-time quantum PAC learner which can perform this task, which implies that learning the class of all possible quantum circuits is certainly out of reach for classical poly-time algorithms. This stringent result therefore breaks the above argument for quantum learning supremacy conditioned on $\mathsf{BQP} \nsubseteq \mathsf{P/poly}$, and shows that the concept class that captures all possible quantum CLF classifiers is not a suitable candidate to consider in the quest for such supremacy (either in a PAC or worst-case learning context).

The comparison between classical and quantum learning changes however when we restrict the concept class from the set of all quantum circuits to something smaller. Consider a concept class $\mathfrak{C}$ that contains the following two concepts: $c(\cdot) : \{0,1\}^n \to \{0,1\}$ which is expressible as a BQP-complete circuit, and $\bar{c}(\cdot)$ which is the negation of $c(\cdot)$. A quantum computer can straightforwardly PAC learn (or even zero-error learn) this concept in polynomial time with a single example $(x, y(x))$, by simply evaluating $c(x)$ and outputting the hypothesis $c(\cdot)$ if $y(x) = c(x)$ and $\bar{c}(\cdot)$ otherwise. For this class to be efficiently classically zero-error learnable however, there would have to exist a classical algorithm which, with a set of $O(\text{poly}(n))$ examples as advice, evaluates $c(x)$ correctly in polynomial time for any input x. The existence of such an algorithm would imply $\mathsf{BQP} \subseteq \mathsf{P/poly}$, since the evaluation of $c(x)$ is BQP-complete; as such we find that, if $\mathsf{BQP} \nsubseteq \mathsf{P/poly}$ – which is generally believed to be true – then there exist concept classes which can be efficiently zero-error learnt quantumly, but not classically.

Still, zero-error performance is an unreasonable requirement for learning, and it is more useful to consider this class from a PAC viewpoint. Now, efficient classical PAC learnability of this class requires an advised classical algorithm which evaluates $c(x)$ correctly *on average*. One way to describe average-case complexity is to follow (an equivalent of) Levin's definition [61, 62]. It defines an algorithm A to be on-average polynomial-time with respect to a distribution $\mathcal{D}$ if there exists a polynomial $P(n)$ and constant $c > 0$ such

that, for all $t > 0$,

$$\mathbb{P}_{x\sim\mathcal{D}}[t_A(x) \leq t] \geq 1 - P(n)/t^c, \tag{6.24}$$

where $t_A(x)$ is the time needed for the algorithm to process x. An algorithm which satisfies this property in fact has median polynomial runtime. By rearranging terms, we may also write the condition as

$$\mathbb{P}_{x\sim\mathcal{D}}[t_A(x) \leq (P(n)/\epsilon)^{1/c}] \geq 1 - \epsilon \tag{6.25}$$

for all $\epsilon > 0$. With this in mind, imagine a set $\mathcal{A}$ of classical algorithms $A_\mathcal{T}$, one for each training set $\mathcal{T}$, which map $x \in \{0,1\}^n$ to 0 or 1. Assume that, for all $\mathcal{D}$, if $\mathcal{T}$ is sampled according to $\mathcal{D}$, then with probability at least $1-\delta$ the algorithm $A_\mathcal{T}$ correctly evaluates a ground truth $y(\cdot) \in \mathfrak{C}$ and satisfies the average-case polynomial-time condition in eq. 6.25. Then consider the set $\mathcal{A}'$ which contains, for every $A_\mathcal{T}$, a truncated algorithm $A'_\mathcal{T}$, which returns the answer of $A_\mathcal{T}$ if $A_\mathcal{T}$ runs within time $(P(n)/\epsilon)^{1/c}$, and a generic "incorrect answer" if $A_\mathcal{T}$ exceeds this runtime. We find that

$$\mathbb{P}_{x\sim\mathcal{D}}[A'_\mathcal{T}(x) \neq y(x)] \leq \epsilon \tag{6.26}$$

(cf. eq. 6.22) and thus a learner $\mathcal{L}$ which samples $\mathcal{T}$ and maps $\mathcal{T}$ to $A'_\mathcal{T}$ is an (ϵ, δ)-PAC learner for $\mathfrak{C}$, which implies that $\mathfrak{C}$ is efficiently classically PAC-learnable.

If we could reverse this argument, we would have a complexity-theoretic condition for quantum learning supremacy: namely if there exists no classical on-average polynomial-time algorithm which correctly evaluates $y(\cdot) \in \mathfrak{C}$, then $\mathfrak{C}$ is not classically PAC learnable and hence there exists a class that can be PAC learnt with a quantum algorithm but not classically. However, if we wish to show that PAC learnability requires the existence of an on-average poly-time algorithm, we run into a problem: namely, an on-average polynomial-time algorithm must compute $c(x)$ *correctly* for all x, regardless of the time needed to do so. What's in the way here is the fact that we cannot verify the correctness of a hypothesis returned by a PAC learner (assuming $\mathsf{BQP} \neq \mathsf{BPP}$). If we could, we could also construct an algorithm which runs the PAC learning algorithm, keeps its output if it is correct, and otherwise runs a brute-force simulation of the quantum circuit (which is guaranteed to output the correct result); and this algorithm would be an on-average poly-time algorithm. But since this is not possible, the requirement that an on-average poly-time algorithm exist is in fact too strong for PAC learning. Therefore, the negated requirement, namely that no such algorithm exist, is

too weak to claim quantum learning supremacy in the PAC context, and we should look for an alternative definition.

The average-case complexity definition that best fits the PAC context is that of *heuristic schemes* [61]. If L is a language and $\mathcal{D}$ a distribution over inputs, then an algorithm A is a heuristic scheme for $(L, \mathcal{D})$ if

- for all n, $x \in \{0,1\}^n$ and $\epsilon > 0$, A runs in time $O(\text{poly}(n, 1/\epsilon))$ on input x;
- for all n and $\epsilon > 0$,

$$\mathbb{P}_{x\sim\mathcal{D}}[A(x) \neq L(x)] \leq \epsilon \tag{6.27}$$

  where $L(x) = \mathbb{1}_{x\in L}$.

Furthermore, HeurP is the set of pairs $(L, \mathcal{D})$ for which there exists a heuristic scheme that can be computed by a deterministic Turing machine (and similarly for the class HeurBPP). Thus, the difference between the class of on-average polynomial algorithms and algorithms for HeurBPP is that the latter are allowed to err with a probability ϵ under $\mathcal{D}$ and must run in polynomial time, while the former can run in superpolynomial time, but must return a correct answer for every input.

We can now use the same reasoning as with on-average poly-time algorithms. Define the language $L = \{x : y(x) = 1\}$, and let $\mathcal{B}$ be a set of algorithms $B_{\mathcal{T}}$. Then if for all $\mathcal{D}$,

$$\mathbb{P}(B_{\mathcal{T}} \text{ is a heuristic scheme for } (L, \mathcal{D})) \geq 1 - \delta \tag{6.28}$$

with $\mathcal{T}$ sampled according to $\mathcal{D}$, then a learner $\mathcal{L}$ which samples $\mathcal{T}$ and maps $\mathcal{T}$ to $B_{\mathcal{T}}$ is an (ϵ, δ)-PAC learner for $\mathfrak{C}$.

Interestingly, with this definition of average-case complexity, we can reverse the argument: suppose there exists a distribution such that there exists no classical algorithm B which is a heuristic scheme for $(L, \mathcal{D})$, i.e. $(L, \mathcal{D}) \notin$ HeurBPP. Then the assumption that $\mathfrak{C}$ is classically PAC learnable leads to a contradiction. Namely, it requires that for all distributions and all concepts $c(\cdot) \in \mathfrak{C}$ (including the concept c such that $c(x) = L(x)$) there exists a classically computable hypothesis h which satisfies eq. 6.22 (and which a learner must be able to find with probability at least $1 - \delta$). But this is impossible assuming $(L, \mathcal{D}) \notin$ HeurBPP. Therefore, under this assumption, $\mathfrak{C}$ is not classically PAC learnable. As such, if one were to find a language $L \in$ BQP and a distribution $\mathcal{D}$ such that $(L, \mathcal{D}) \notin$ HeurBPP, then this would be sufficient to claim quantum PAC learning supremacy. After all, in this

case one could construct a concept class $\mathfrak{C}_L = \{L(\cdot), \bar{L}(\cdot)\}$ that is efficiently PAC learnable with a quantum learning algorithm, but not classically.

The question remains then, whether there exists a pair $(L, \mathcal{D})$ for which the above condition holds. In particular, it is likely that we cannot link the existence of such a pair to a worst-case complexity statement such as $\mathsf{BQP} \subseteq \mathsf{P/poly}$ in the same straightforward manner. After all, it is known that the reductions from worst-case hardness to average-case hardness would imply a collapse of the polynomial hierarchy for many complexity classes, including NP [63], and the situation may be equally complex in the quantum case [64]. Thus the statement $\mathsf{BQP} \nsubseteq \mathsf{P/poly}$ is unlikely to imply anything about the existence of concept classes that are efficiently PAC learnable with a quantum computer, but not classically. Nonetheless, based on recent results showing average-case hardness of quantum circuit sampling [65, 66] and an average-case separation between quantum and classical shallow circuits [67], we conjecture that no efficient classical algorithm exists which decides a BQP-complete problem with examples, and hence there is a BQP-complete language L such that $(L, \mathcal{D}) \notin \mathsf{HeurBPP}$ for some distribution $\mathcal{D}$. Under this condition, there exist concept classes that are quantumly PAC learnable, but not classically, without additional assumptions about average-case hardness.

In conclusion, we find that the problem of characterising PAC learning of quantum CLF classifiers is twofold. Firstly, as we have seen, statements providing an average-case hardness of BQP-complete problems for classical machines would have direct implications for a separation between quantum and classical PAC learning – that is, it would prove a directly applicable form of quantum learning supremacy[6]. Secondly, the mere existence of quantumly but not classically learnable concept classes does not give any indication about the properties of these classes. That is, more research would be needed to precisely characterise the border between the classical and the quantum learnable; insights in this direction would give a clear motivation for the physical realisation of quantum learners, as well as precise guidelines on which classification problems quantum learning is bound to provide an advantage. All in all, it is apparent that there will be ample opportunity for future research that addresses the power of quantum computing in machine learning problems.

[6] During the time of the writing of this thesis, a result was shown by Liu et al. [68], which proves a separation between classical and quantum PAC learning with CLF classifiers under the condition that the discrete logarithm problem, which is contained in BQP, cannot be efficiently solved on a classical computer.

Chapter 7

Conclusion and outlook

In this thesis, we have extended the work of Havlíček et al., scrutinising the properties of the quantum counterpart to SVM learning which they propose. We have worked out this quantum learning model by precisely defining the explicit and implicit formulations of quantum CLF classifiers, with discussions on the feasibility of implementing of both types and the restrictions one encounters in doing so. We have compared quantum explicit and implicit classifiers, including their training set classification performance under a regularised risk functional. Further, we have characterised the generalisation performance of these quantum CLF classifiers in terms of the fat shattering dimension, as well as possible sources for quantum supremacy with such classifiers, which we have discussed in a learning complexity setting. In section 5.1 we have seen that, while the set of quantum explicit classifiers is richer than the set of quantum implicit classifiers, and therefore offers more freedom in the choice of classifiers, this freedom can be harnessed only in a restricted way, since the most general explicit classification function evaluations require runtime exponential in the number of qubits. Implicit classifiers, on the other hand, only depend on the underlying kernel, which must be evaluated at most polynomially many times in the size of the training set. Section 5.2 has shown, by invoking the representation theorem, that quantum implicit classifiers induced by a training set can always achieve a minimum regularised risk value on this training set, while for certain restricted sets of explicit classifiers, this is not the case (theorem 5.6). What makes explicit classifiers interesting, however, is that the greater richness of such classifier sets allows one to simulate other quantum learning algorithms, such as fixed-hamiltonian optimisation procedures. Moreover, explicit classifiers offer different methods for model simplification than implicit classifiers, namely through constraints on the parametrised circuit, which may further

reduce overfitting.

Next, in section 5.3, we have given a bound on generalisation performance of quantum CLF classifiers, by showing a tight upper bound on their fat shattering dimension (theorems 5.9 and 5.11). We found that the fat shattering dimension of general quantum CLF classifiers with norm at most ν classifying pure states by margin γ is upper bounded by

$$\mathrm{fat}_{\mathcal{H}}(\gamma) \leq \min\{9\nu^2/\gamma^2, 4^n + 1\} + 1 \tag{7.1}$$

which is tight, following from the lower bound

$$\mathrm{fat}_{\mathcal{H}}(\gamma) \geq \min\{\nu^2(1 - 2^{-n})/\gamma^2, 2^n\} \tag{7.2}$$

for observables with norm at least ν. We point out that the use of quantum CLF classifiers as opposed to classical ones introduces a new regularisation parameter, the observable rank, and that we have indications that observables of higher rank have the possibility to correctly classify more points or with higher margin, promising additional freedom in achieving better generalisation performance with regard to theorem 4.19. Furthermore, we argue that a greater richness of quantum feature maps could enhance generalisation performance as well.

Besides these properties of quantum CLF classifiers, we have considered computational hardness of evaluating quantum CLF classifiers. Using precise definitions of computational hardness in this learning setting, we have shown that, both for explicit and implicit classifiers, computational hardness of evaluating the underlying expectation values and kernels respectively implies hardness of computing the classifier itself, and vice versa – a property that is often assumed in literature without proof (section 6.3). Furthermore, this shows that the existence of quantum kernel families which are classically hard to evaluate, implies the existence of quantum CLF classifier families that are classically hard.

However, this fact alone does not imply any form of quantum learning supremacy, as we point out in section 6.4. While we argue that necessarily $\mathsf{BQP} \subseteq \mathsf{P/poly}$ if there exist classical learning algorithms which can solve all classification problems that can be solved by quantum learning algorithms (which is a worst-case statement), this is a requirement too strong to reflect common practice in machine learning, which asks only for good classification and generalisation performance in average case learning, as described by the PAC learning principle. In this context, we note that there must exist a language $L \in \mathsf{BQP}$ and a distribution $\mathcal{D}$ such that $(L, \mathcal{D}) \notin \mathsf{HeurBPP}^{\mathcal{O}_{L,\mathcal{D}}}$, where $\mathcal{O}_{L,\mathcal{D}}$ is an oracle that returns random examples $(x, \mathbb{1}_{x\in L})$ drawn from

$\mathcal{D}$, to guarantee the existence of a concept class which can be efficiently PAC-learnt on a quantum computer, but not using a classical computer.

We leave open a number of questions for future work. Firstly, while we discussed generalisation performance of quantum CLF classifiers through the fat shattering dimension, it is still not completely clear what implications this has on generalisation performance of quantum CLF classifiers in practice, as compared to that of classical SVMs. More precisely, it would be fruitful to have a clear picture of the influence that the observable rank, acting as a regularisation parameter, has on the margin and the size of training sets that can be classified correctly. The same question holds for quantum feature maps, and whether there exist feature maps which guarantee better training set classification in the same time than classical feature maps.

Next, in the context of quantum learning supremacy, it would be very valuable to gain more knowledge about the question whether the complexity theoretic statement revolving around $\mathsf{HeurBPP}$ which we formulated holds, as this answers the question whether there exists a form of quantum learning supremacy. However, seeing as similar questions, such as whether $\mathsf{BPP} = \mathsf{BQP}$ and $\mathsf{P} = \mathsf{NP}$, have to this day not been answered, we suspect this to be a very difficult open problem that will not see a solution soon. Lastly, extending the question, it would be fruitful to be able to precisely draw the boundary between classically and quantumly efficiently PAC learnable problems. After all, this would provide key motivations to employ quantum algorithms for learning in practical settings. However, as we mentioned, this is a highly complex topic, and decisive results may remain out of reach in the near future.

References

[1] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, *Supervised learning with quantum-enhanced feature spaces*, Nature (2019).

[2] P. Benioff, *The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines*, Journal of Statistical Physics (1980).

[3] R. P. Feynman, *Simulating physics with computers*, International Journal of Theoretical Physics (1982).

[4] L. K. Grover, *A fast quantum mechanical algorithm for database search*, in *Proceedings of the Annual ACM Symposium on Theory of Computing*, 1996.

[5] P. W. Shor, *Algorithms for quantum computation: discrete logarithms and factoring*, in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[6] E. Bernstein and U. Vazirani, *Quantum complexity theory*, SIAM Journal on Computing (1997).

[7] T. F. Rønnow, Z. Wang, J. Job, S. Boixo, S. V. Isakov, D. Wecker, J. M. Martinis, D. A. Lidar, and M. Troyer, *Defining and detecting quantum speedup*, Science (2014).

[8] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, *Quantum machine learning*, 2017.

[9] A. W. Harrow, A. Hassidim, and S. Lloyd, *Quantum algorithm for linear systems of equations*, Physical Review Letters (2009).

[10] S. Lloyd, M. Mohseni, and P. Rebentrost, *Quantum algorithms for supervised and unsupervised machine learning*, (2013).

[11] P. Rebentrost, M. Mohseni, and S. Lloyd, *Quantum support vector machine for big data classification*, Physical Review Letters (2014).

[12] J. Preskill, *Quantum computing in the NISQ era and beyond*, Quantum (2018).

[13] D. Willsch, M. Nocon, F. Jin, H. De Raedt, and K. Michielsen, *Gate-error analysis in simulations of quantum computers with transmon qubits*, Physical Review A (2017).

[14] K. Temme, S. Bravyi, and J. M. Gambetta, *Error mitigation for short-depth quantum circuits*, Physical Review Letters (2017).

[15] C. Cortes and V. Vapnik, *Support vector networks*, Machine Learning (1995).

[16] V. N. Vapnik, *Statistical learning theory*, Wiley-Interscience, 1998.

[17] D. J. Griffiths and D. F. Schroeter, *Introduction to Quantum Mechanics*, Cambridge University Press, 3rd edition, 2018.

[18] V. V. Shende, S. S. Bullock, and I. L. Markov, *Synthesis of quantum-logic circuits*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (2006).

[19] R. Iten, R. Colbeck, I. Kukuljan, J. Home, and M. Christandl, *Quantum circuits for isometries*, Physical Review A (2016).

[20] H. B. Curry, *The method of steepest descent for non-linear minimization problems*, Quarterly of Applied Mathematics (1944).

[21] J. Schmidhuber, *Deep learning in neural networks: an overview*, 2015.

[22] T. Howley, M. G. Madden, M. L. O'Connell, and A. G. Ryder, *The effect of principal component analysis on machine learning accuracy with high-dimensional spectral data*, Knowledge-Based Systems (2006).

[23] D. Needell and J. A. Tropp, *CoSaMP: Iterative signal recovery from incomplete and inaccurate samples*, Applied and Computational Harmonic Analysis (2009).

[24] A. Peruzzo, J. McClean, P. Shadbolt, M. H. Yung, X. Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, *A variational eigenvalue solver on a photonic quantum processor*, Nature Communications (2014).

[25] M. Schuld and N. Killoran, *Quantum Machine Learning in Feature Hilbert Spaces*, Physical Review Letters (2019).

[26] M. Rötteler, *Quantum algorithms for highly non-linear Boolean functions*, in *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, 2010.

[27] D. K. Park, C. Blank, and F. Petruccione, *The theory of the quantum kernel-based binary classifier*, Physics Letters, Section A: General, Atomic and Solid State Physics (2020).

[28] C. Blank, D. K. Park, J. K. K. Rhee, and F. Petruccione, *Quantum classifier with tailored quantum kernel*, npj Quantum Information (2020).

[29] G. A. Garreau and W. Rudin, *Real and complex analysis*, The Statistician (1987).

[30] D. Sejdinovic and A. Gretton, *What is an RKHS?*, Technical report, University of Oxford, 2012.

[31] J. Mercer, *Functions of positive and negative type, and their connection with the theory of integral equations*, Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character (1909).

[32] J. Vert, K. Tsuda, and B. Schölkopf, *A primer on kernel methods*, Kernel Methods in Computational Biology, 35-70 (2004) (2004).

[33] B. Schölkopf, R. Herbrich, and A. J. Smola, *A generalized representer theorem*, in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2001.

[34] V. Vapnik and A. Chervonenkis, *Theory of pattern recognition*, Nauka, Moscow, 1974.

[35] T. M. Mitchell, *Machine learning*, McGraw-Hill, New York, 1997.

[36] V. N. Vapnik, *The nature of statistical learning theory*, Springer, 2000.

[37] C. J. C. Burges, *A tutorial on support vector machines for pattern recognition*, Data Min. Knowl. Discov. **2**, 121 (1998).

[38] J. Mount, *How sure are you that large margin implies low VC dimension?*, Technical report, 2015.

[39] J. Shawe-Taylor, P. L. Bartlett, R. C. Williamson, and M. Anthony, *Structural risk minimization over data-dependent hierarchies*, IEEE Transactions on Information Theory (1998).

[40] A. Knutson and T. Tao, *Honeycombs and sums of Hermitian matrices*, Notices Amer. Math. Soc. **48** (2000).

[41] E. Farhi and H. Neven, *Classification with quantum neural networks on near term processors*, (2018).

[42] K. Beer, D. Bondarenko, T. Farrelly, T. Osborne, R. Salzmann, D. Scheiermann, and R. Wolf, *Training deep quantum neural networks*, Nature Communications **11**, 808 (2020).

[43] G. Vidal, *Class of quantum many-body states that can be efficiently simulated*, Physical Review Letters (2008).

[44] E. Grant, M. Benedetti, S. Cao, A. Hallam, J. Lockhart, V. Stojevic, A. G. Green, and S. Severini, *Hierarchical quantum classifiers*, npj Quantum Information (2018).

[45] E. Farhi, J. Goldstone, and S. Gutmann, *A quantum approximate optimization algorithm*, page 1 (2014).

[46] S. Aaronson, *The learnability of quantum states*, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences (2007).

[47] A. Ambainis, A. Nayak, A. Ta-Shma, and U. Vazirani, *Dense quantum coding and quantum finite automata*, Journal of the ACM (2002).

[48] S. Lloyd, M. Schuld, A. Ijaz, J. Izaac, and N. Killoran, *Quantum embeddings for machine learning*, (2020).

[49] V. V. Shende and I. L. Markov, *On the CNOT cost of Toffoli gates*, Quantum Information and Computation (2009).

[50] O. Goldreich, *In a world of P=BPP*, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (2011).

[51] L. G. Valiant, *A theory of the learnable*, Communications of the ACM **27**, 1134 (1984).

[52] L. Adleman, *Two theorems on random polynomial time*, in *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, 1978.

[53] C. Bennetts and J. Gill, *Relative to a random oracle A,* $\mathsf{P}^A \neq \mathsf{NP}^A \neq \mathsf{coNP}^A$ *with probability 1*, SIAM J. Comput. (1981).

[54] D. Aharonov, V. Jones, and Z. Landau, *A polynomial quantum algorithm for approximating the Jones polynomial*, Algorithmica (New York) (2009).

[55] M. H. Freedman, A. Kitaev, and Z. Wang, *Simulation of topological field theories by quantum computers*, Communications in Mathematical Physics (2002).

[56] M. H. Freedman, M. Larsen, and Z. Wang, *A modular functor which is universal for quantum computation*, Communications in Mathematical Physics (2002).

[57] E. Witten, *Quantum field theory and the Jones polynomial*, Communications in Mathematical Physics (1989).

[58] S. R. Buss, *Bounded arithmetic, cryptography and complexity*, Theoria **63**, 147 (2008).

[59] S. Arunachalam, A. B. Grilo, and A. Sundaram, *Quantum hardness of learning shallow classical circuits*, (2019).

[60] Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé, *Classical hardness of learning with errors*, in *Proceedings of the Annual ACM Symposium on Theory of Computing*, 2013.

[61] A. Bogdanov and L. Trevisan, *Average-case complexity*, Foundations and Trends in Theoretical Computer Science (2006).

[62] L. A. Levin, *Average case complete problems*, SIAM Journal on Computing (1986).

[63] A. Bogdanov and L. Trevisan, *On worst-case to average-case reductions for NP problems*, in *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, 2003.

[64] N.-H. Chia, S. Hallgren, and F. Song, *On basing one-way permutations on NP-hard problems under quantum reductions*, Quantum **4**, 312 (2020).

[65] M. J. Bremner, A. Montanaro, and D. J. Shepherd, *Average-case complexity versus approximate simulation of commuting quantum computations*, Physical Review Letters (2016).

[66] A. Bouland, B. Fefferman, C. Nirkhe, and U. Vazirani, *Quantum supremacy and the complexity of random circuit sampling*, in *Leibniz International Proceedings in Informatics, LIPIcs*, 2019.

[67] F. Le Gall, *Average-case quantum advantage with shallow circuits*, in *Leibniz International Proceedings in Informatics, LIPIcs*, 2019.

[68] Y. Liu, S. Arunachalam, and K. Temme, *A rigorous and robust quantum speed-up in supervised machine learning*, (2020).

[69] H. Buhrman, R. Cleve, J. Watrous, and R. de Wolf, *Quantum fingerprinting*, Physical review letters (2001).

[70] S. Hanneke and A. Kontorovich, *Optimality of SVM: novel proofs and tighter bounds*, Theoretical Computer Science (2019).

[71] H. Nishimura, *Quantum Computation with Supplementary Information*, Ipsj Digital Courier **1**, 407 (2005).