Master Thesis

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Control Engineering

Quantum machine learning

Bc. Jan Svoboda

Supervisor: Mgr. Jakub Mareček Ph.D.
Field of study: Cybernetics and robotics
January 2025

Acknowledgements

I would like to thank here, in the last phase of my student life, the people who have supported me unconditionally during my studies. First and foremost, I must mention my family, who gave me the opportunities to study and develop in the fields of my own choice and never forced me to do anything, without them I would never have made it to my Master's degree. I would also like to thank my friends, both those I have met outside of school and those I have met within it. Even in moments when I didn't feel like singing, I always had people around me who helped me, gave me advice, or were just there for me. Finally, I can't forget my best friend, whom I met at CTU and since then she has been the biggest support I have had in my life. In moments when I didn't even believe in myself she put her trust in me, she was able to make me laugh and gave me the desire to improve and keep working.

So thank you all, it was worth it.

Declaration

I declare that the presented work is solely mine and that I have cited all the used literature.

In Prague, 5. January 2025

Abstract

Although quantum computing is not being used in the real world yet, it is important to study it in order to understand its concepts and expand our knowledge of the possibilities it offers us. This paper compares quantum machine learning methods with kernel methods, as the two have much in common. It is our intention to devise a methodology for the generation of quantum kernels in such a way that classical simulation of them is $\#P$ hard. The aforementioned methodologies are subsequently tested on real data sets, with somewhat disappointing results.

Keywords: quantum computing, quantum kernels, kernel target alignment, kernel methods, $\#P$ hardness

Supervisor: Mgr. Jakub Mareček Ph.D.

Abstrakt

Přestože se kvantové počítání zatím v reálném světě nepoužívá, je důležité jej studovat, abychom pochopili jeho koncepty a rozšířili své znalosti o možnostech, které nám nabízí. Tento článek porovnává metody kvantového strojového učení s kernelovými metodami, protože tyto dvě metody mají mnoho společného. Naším záměrem je navrhnout metodiku generování kvantových kernelů tak, aby jejich simulace pomocí klasických počítačových systémů byla $\#P$ složitá. Výše uvedené metody následně testujeme na reálných souborech dat, přičemž výsledky jsou poněkud neuspokojivé.

Klíčová slova: kvantové počítání, kvantové kernely, cílové kernelové zarovnání, kernelové metody, $\#P$ složitost

Překlad názvu: Kvantové strojové učení

Contents

Figures

Tables

Chapter 1

Introduction

Over the last few years, quantum computing has gained more and more scientific focus, lending itself to be the revolutionizer across various artificial intelligence fields including machine learning. While machine learning is obviously one of the fastest growing fields of artificial intelligence, it is not disregarded by quantum computing with much desired properties, with exponential speedup being one of them. Exponential speedup using quantum computation is the result of quantum phenomena called entanglement and superposition. These make it possible to perform computations exponentially faster compared to classical computers. This may accelerate not only training but also classification. However, quantum computation does not only bring benefits. Several disadvantages need to be pointed out, such as hardware limitations. Although there have been rapid advances, up-to-date quantum computers still face problems of low computational power stemming from coherence time, gate fidelity, and limited qubit connectivity. These factors not only make quantum computers noisy and prone to errors but most importantly, they impose limits on what problems can and cannot be solved by quantum computers today. Correcting errors and filtering out noise can be done, but the utilization of more qubits is needed, making the computation demanding on more resources, thus inefficient. Another problem with quantum computing is the same phenomenon that quantum mechanics is based on: that measurement is part of an experiment. With qubits living in their own Hilbert space, mapping the results of quantum computation back to classical data for it to be used by scientists immensely affects the qubit state, and the development of trustworthy methods for state readout is still a problem. In order to fully exploit the quantum computational power once large-scale quantum computers are designed, all of the obstacles listed above must be tackled. With the rise of quantum computing, several questions have arisen. One of the most questioned ones is the existence of quantum advantage, meaning whether there are tasks that can be done on quantum computers that couldn't be done on classical computers. While evaluating, several qualitative measures are taken into consideration, such as time of computation, resources needed for the task, etc. In this work, we are introducing quantum kernels that are #P hard to evaluate classically. These findings are then justified by experiments that aim to show that these kernels are indeed useful and not only an example of a rather useless tool with hard simulability properties.

Chapter 2

Quantum computing

As this thesis revolves around quantum computers, introducing the concept of quantum computing in general is necessary. Currently, all computation is performed by classical computers, which are the result of the tireless development of technological companies. This development has driven the evolution of computers to a mind-boggling state, where computers and the models implemented on them are more powerful than ever, affecting our lives in almost sci-fi ways. This is the big picture; however, companies that produce computers are on the verge of an interesting event. The problem that Gordon Moore predicted back in 1965 [?] and revisited in 1975 relates to the well-known Moore's Law. Moore's Law essentially states that the number of components per integrated circuit doubles over a period of time—initially stating that annually (1965), later on biennially (1975). The length of this time period is not really important for this thesis, but rather the fact that some progress is being made. But a more important implicit consequence is the size of the circuit components. The majority of the components in an integrated circuit are transistors. The process of increasing the number of components in an integrated circuit comes with a reduction in their size. Up until now, more powerful computers were built simply by adding more smaller transistors into the same size circuit. We have come to a moment where the size of the transistors is atomic and it cannot be reduced any further. On this scale, the laws of classical mechanics cease to make sense, and quantum mechanics becomes more relevant. This may suggest that by decreasing the size of transistors, a classical computer will become a quantum computer, but this is not the case. A quantum computer is a totally different device, and the transition to using quantum computers from classical ones is not continuous, but rather a great leap in information technology.

2.1 The fundamentals

2.1.1 Qubit

The most basic unit of information stored in a classical computer is a bit, which stores either a 1 or a 0. The quantum equivalent is called a qubit. However, a qubit is not just an ordinary unit for storing information. A qubit obeys quantum mechanical laws, enabling it to be not only 1 or 0 but also any state in between. It can be in a superposition of these two states simultaneously. We often define two orthogonal states, which are also referred to as the computational basis:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad |1\rangle = \begin{pmatrix} 0 \\ 1. \end{pmatrix}$$

A quantum state $|\psi\rangle$ is part of the two-dimensional complex Hilbert space $\mathcal{H}^2$

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$$

with $|\alpha|^2 + |\beta|^2 = 1, \alpha, \beta \in \mathbb{C}$. This representation is sufficient if we discuss pure states. However, quantum computers can also exploit mixed states. In such cases, it is more advantageous to use the density matrix $\rho = \langle\psi| |\psi\rangle$ when representing qubits.

2.1.2 Quantum circuit

A quantum circuit connects a set of qubits, which means that it captures the initial circuit state $|\psi\rangle = \bigotimes_{i=1}^{q} |\psi_i\rangle$, where $|\psi_i\rangle$ are the states of each individual qubit. In this work, we assume that each qubit is initialized to the state $|0\rangle$. Also worth mentioning is the dimensionality of this state—since it is the tensor product of q 2-dimensional Hilbert spaces, the resulting space is a Hilbert space $\mathcal{H}^{2^q}$ with dimension 2^q. The quantum circuit then performs the computation via unitary matrices $U \in \mathbb{C}^{2^q \times 2^q}$. Here, q represents the number of qubits in the circuit. A unitary matrix is any matrix that satisfies $UU^\dagger = \mathbb{I}$, where $U^\dagger$ is the Hermitian adjoint and $\mathbb{I}$ is the identity matrix. A quantum circuit then prepares the quantum state:

$$|\psi\rangle = U |0\rangle . \tag{2.1}$$

However, in real applications, the circuit's unitary matrix results from applying many unitary matrices, each targeting specific qubits. To clarify, we often visualize a quantum circuit using a diagram, which adds clarity to the circuit's operations. This diagram arranges qubits on horizontal lines, with interactions between the qubits represented by gates. The relevant gates' unitaries are introduced in 7; however, for now, the reader should simply be aware of the quantum circuit diagram representation, as shown in example 2.1.

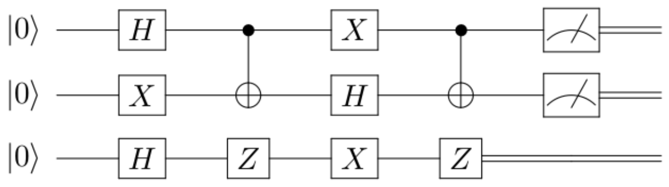


Figure 2.1: Quantum circuit example with $q = 3$ qubits and depth 4

2.1.3 Measurement

In the previous subsection, we introduced the terms quantum circuit and quantum state $|\psi\rangle$, which is prepared by the quantum circuit. One might wonder how we leverage this state in real-world applications. We have not mentioned the last step of quantum computation yet. It is evident that the state of a quantum circuit, even for a small number of qubits, may exist in dimensions far beyond human comprehension. To utilize this state and the computation it represents, we need to measure it. In quantum mechanics, measurement is a critical component of its theory. Once a measurement is applied, the entire state collapses to the measured state. In the diagram, measurement is even shown as the last gate applied to qubits 1 and 2. This measurement is realized using a Hermitian matrix M, and the possible outcomes of the measurement are the eigenvalues of the operators. Throughout this work, we assume measurement on a computational basis $|0\rangle, |1\rangle$, although other bases are possible as well.

2.2 Quantum computing specifics

Quantum computing research is extensive and led by technological giants such as Google and IBM, but what fuels their excitement? Quantum computing offers an undeniable speed-up that is tempting. The main challenge lies in the hardware, which lacks the desired properties. In this section, we explore the possibilities of quantum computing as well as its significant challenges.

2.2.1 Advantages

We have already mentioned some of the advantages of quantum computing. One of them has to be the superposition of states. Qubits can be at the superposition of two states. With a growing number of qubits the dimension of space that can be searched using quantum computers grows exponentially. Another significant concept is the quantum entanglement which cannot be omitted from a thesis regarding quantum computing. Quantum entanglement is a physical phenomenon occurring when two or more particles become linked in such a way that we cannot describe one particle independently of the others. Entangled particles then obtain specific behavior such that the measurement of one particle determines the state of the other particle. This phenomenon shatters our thinking of classical mechanics, and even the greatest physicians, such as Albert Einstein, were not sure about this idea. But entanglement exists, and we can make use of it, which most quantum algorithms offering advantage do. Superposition and entanglement hand in hand enable the quantum computer to make computations of many states all at once - providing parallelism. This is something we need to consider. There are some tasks, for example, finding an element in an array, that require looking at every element separately. Quantum computers offer a remarkable possibility - to look at every element in an array in a single computational task. This unlocks possibilities of quantum speed-up, such as Lov Grover's algorithm, which takes advantage of parallelism and shrinks time complexity for searching for an element in an unsorted array from classical $\mathcal{O}(N)$ to quantum $\mathcal{O}(\sqrt{N})$.

2.2.2 Challenges

Until now, we have discussed only the advantages and possibilities of quantum computing. However, as intriguing as these may seem, quantum computing still lacks real-world applications, since creating a reliable quantum computer may be one of the most challenging engineering tasks of the 21st century. Since the first proposal by Richard Feynman in 1981, the quantum computing industry has come a long way, yet we are not there yet. We are on the verge of what some call the NISQ era, with NISQ standing for Noisy Intermediate-Scale Quantum. The intermediate scale is often defined as a range from 50 up to a few hundred qubits that may be used in a quantum computer. Even though there are computers with more than 50 qubits already, it is still the lowest limit, and we are still discovering what quantum computers are capable of. The term 'Noisy' may be even more determining with respect to the power of quantum computers. Even with the limited number of qubits we are able to construct, we cannot do so reliably. Qubits suffer from several deficiencies such as error per gate, gate execution time, qubit initialization, connectivity of qubits, etc. Error per gate is one of the properties that are primary for optimization. We need to ensure that we apply certain quantum gates with high accuracy, but even in the best hardwares, the error per gate remains above 1%. On one hand, with the growing number of qubits q, we may use these qubits for error correction. On the other hand, this approach will backfire due to the qubit connectivity problem. Connecting all the qubits in a quantum computer is demanding, and to use error correction effectively, we assume the qubit connection graph to be dense, which is a challenging engineering problem. Gate execution time is important as well, with time often being one of the complexity arguments that determines the algorithm's performance. The last challenge we mention is qubit initialization, as even initializing a quantum computer to the $|0^q\rangle$ state needs to be considered as a part of quantum computation.

2.3 Current state

Currently, there are several approaches to creating general quantum computers, but we will only mention two of the most widely used ones.

2.3.1 Superconducting qubits

Superconducting qubits use a circuit of superconducting materials, giving them the property of carrying an electric current with zero resistance. This comes with a trade-off of having to operate at extremely low temperatures close to absolute zero, which imposes challenging conditions on cooling the system down. A qubit is represented as a Cooper pair - a pair of electrons and gates are applied to qubits via microwave pulses. This approach offers great speed with circuits operating at GHz frequencies. Also, it is believed that the number of qubits can scale up easier than other techniques with the IBM Condor computer having 1121 qubits. The biggest disadvantage is the decoherence of qubits due to the interactions with the environment, which may ruin experiments. For example, error rates per single gate are generally around 0.1%. This procedure is mainly used by IBM and Google. Since this work experiments are run using Qiskit, IBM's quantum computing library, this is the approach used in this thesis as well.

2.3.2 Trapped ions

Another approach to creating quantum computers is trapped ions. Where superconducting qubits come up short, trapped ions excel and vice versa. This approach allows for high-fidelity and longer coherence times. Trapped ions offer the lowest error rates in the quantum computing industry with companies using them claiming an error rate of 0.001%. The disadvantage is operation times - for one-qubit gate up to $20ns$, and for two-qubit gates even $200ns$. Also, scalability appears to be harder than on superconducting qubits, with the highest number of qubits at a trapped ions computer right now 56. The leading companies using this technique are IonQ and Quantinuum.

2.4 Possibilities

As mentioned earlier, the Hilbert space of a quantum computer grows exponentially with the number of qubits. With the most powerful computer of IBM harnessing a power of more than 1200 qubits [Bro24], quantum circuit state lives in a dimension that is higher than the number of all atoms in the known universe. We are dealing with an extremely complex quantum world, but this does not necessarily ensure that quantum computers are more powerful. In this section, we show several applications of quantum computing that may lead to quantum computing being advantageous compared to classical computers.

2.4.1 Quantum Fourier transform

A gateway to harnessing quantum advantages is the Quantum Fourier Transform (QFT) [Cop02]. Like its classical counterpart, it transforms data from the computational basis into the Fourier basis. A QFT circuit with q qubits exploits parallelism by computing Fourier coefficients simultaneously, using $O(q^2)$ gates, compared to the classical $O(q2^q)$, thus offering exponential speedup. Assuming that the quantum computer natively implements the controlled phase gate, the complexity can be further reduced to $O(q\log(q))$ [HH00]. The QFT, while advantageous on its own, merely serves as a gateway, with many algorithms linked to this foundational algorithm.

2.4.2 Quantum phase estimation

Quantum phase estimation (QPE) appears to be the next step to obtaining quantum advantage. QPE is used to estimate the eigenvalues of unitaries U, with quantum computers capitalizing on the fact that all eigenvalues of unitaries are of the norm $\|\lambda_i\| = 1$. Estimating eigenvalues is thus equivalent to finding their phases. QPE uses two registers, the first being the output register, and the second being the input. The first register is an applied set of Hadamard gates, then controlled unitaries U^{2^j} are applied to the second register with q_j being the control qubit. After the controlled operations are done, inverse QFT is applied and the results are obtained by using measurement on the first register. QPE is a backbone of more complex quantum algorithms such as Shor's algorithm, HHL algorithm, etc. [NC10]

2.4.3 Shor's algorithm

We recognize that one of the most important algorithms fully utilizing quantum computers is Shor's algorithm [Sho94]. This algorithm solves the problem of finding the prime factors of a large integer S , which is widely known as the key to modern-day encryption systems [RSA78]. The best-known non-quantum algorithm for this task is a general number field sieve that works in subexponential time $\mathcal{O}(e^{1.9(log(S))^{1/3}(log(log(S)))^{2/3}})$. Shor's algorithm offers the speed-up of $O\Big((\log S)^2(\log\log S)\Big)$ using the fastest multiplication algorithms. [HvdH21]. Shor's algorithm leverages quantum superposition by raising an initial guess a up to S powers at once. The Quantum Fourier transform [CEMM98] then finds a period r in the resulting function $f(x) = a^x \bmod S$ present in the superposition. With a recently established r, we may improve the initial guess to $a^{r/2} \pm 1$. This updated guess, together with the original number, S gives the desired factors. For finding the factors, a classical computer is used utilizing common algorithms such as a general number field sieve.

2.4.4 HHL algorithm

The Harrow-Hassidim-Lloyd algorithm [HHL09] solves the problem of finding a solution to the system of linear equations $Ax = b$ with N variables. The HHL algorithm, under the assumption that the matrix A is sparse and well conditioned, i.e., has a low number κ, offers an exponential speed-up with time complexity being $O(\log(N)\kappa^2)$, as opposed to classical algorithms $O(N\kappa)$. HHL uses the aforementioned QFT and QPE to approximate the result x by preparing an inverse matrix A^{-1} and measuring on an ancilla qubit. The measuring happens to be a crucial part since to get full information about $x$the HHL algorithm needs to be run $\mathcal{O}(n)$ times, which ruins the exponential speed-up. However, mostly only some traits of the solution are needed, or we only need a sample. In this case, HHL may lead to new research in physics, chemistry, computational science, and more.

2.5 Future

We have only mentioned a minor part of quantum computing possibilities, but we need to mention that all of them require advanced quantum computer technology with better decoherence times and error correction, a technology that will be more reliable overall. IBM has already announced their goal for the next 10 years - to create a quantum computer with more than 10^5 [Rev23] qubits. Recent improvements show that they may be able to do that. Maybe not in ten years, but someday, it will certainly happen. In the meantime, our work is to find what we could do with reliable quantum computers once we have them. We believe that quantum computers will not replace classical computers anytime soon but rather become another layer of computation with its own properties. Methods that shall excel will be doing so by using a partnership between classical and quantum computation, as does Shor's algorithm for example. We take a similar approach in this work, using quantum computers to define a function that no classical computer can reliably estimate and using classical computers to utilize this function in a supervised machine learning task. The settings and further description of such tasks are defined in the next chapter.

Chapter 3

Quantum kernels

Quantum supervised machine learning can be described in many ways. In this paper, we rely on the strong similarity to kernel methods, which we will be introducing in this section. Even though the usage of kernel methods is on its decline with neural networks being vastly used across artificial intelligence fields, its advantages are still undeniable. In addition, quantum neural networks (QNN) are set to be more prone to suffer from the phenomenon of barren plateaus than their classical counterparts [MBS$^+$18]. These indications can cause us to witness kernel methods become significantly more efficient and frequently used in quantum computing. Let's introduce some important concepts that will help us understand more deeply the similarities between quantum-supervised machine learning and kernel methods and how we can leverage these similarities in the applications we need. The previous chapter aimed to introduce quantum computing in general. In this chapter, we focus on more specific tasks of quantum computing - supervised machine learning in the quantum domain. More specifically, we lay the foundations on which the main part of this thesis lies. The main concept of this thesis is to show that quantum computers can be handy when classifying data. Furthermore, we wish to show that quantum computing can not only be useful but, more importantly, more advantageous over conventional methods. The first step is to define the problem that we want the quantum computer to solve.

3.1 Initial problem

The initial problem is a notoriously known data classification problem. We are given training data $\mathcal{X}_{train} = \{x_1, \ldots, x_n\}$ from data space Ω^d with its assigned labels $\mathcal{Y}_{train} = \{y_1, \ldots y_n\}$. Data space Ω^d can be any space, but in this thesis, we assume it to be real $\mathbb{R}^d$ with dimension d, and the labels are from binary space $\mathbb{B}$ with slight notation change $\mathbb{B} = \{-1, 1\}$ since binary classification in machine learning often uses ± 1 labels. Generally, the label-assigning function $f : \mathbb{R}^d \to \mathbb{B}$ is not known, but we are certain it satisfies $f(\mathcal{X}) = \mathcal{Y}$. The goal for the algorithm is to learn the classifier utilizing an approximate map $\hat{f}$so that when the classifier sees new data (often called test data), $\mathcal{X}_{test} = \{x_{1t}, \ldots, x_{nt}\}$ it classifies this data with labels $\hat{f}(\mathcal{X})$ that match the real labels $f(\mathcal{X}_{test}) = \mathcal{Y}_{test} = \{y_{1t}, \ldots, y_{nt}\}$ with high probability. Model performance is often evaluated using generalization error.

Definition 3.1 (Generalization error). Generalization error or the expected loss of a model $\hat{f}$ is

$$\mathcal{E}_g(\hat{f}) = \int_{\nleq \times \mathbb{B}} \mathcal{L}(\hat{f}(x), y) p(x, y) \mathrm{d}x \mathrm{d}y, \tag{3.1}$$

where $\mathcal{L}(\hat{f}(x), y)$ is the loss function which outputs the loss of the model's classification - in this thesis, we will use *hinge-loss* $= \mathcal{L}_h(\hat{f}(x), y) = \max(0, 1 - \hat{f}(x)y)$ and $p(x, y)$ is the joint probability distribution of the data sample x and its label y.

In real-world application, we often do not have access to the exact joint probability distribution. That is why instead of generalization error, empirical error is used.

Definition 3.2 (Empirical risk). Given t data points, the empirical risk of a model $\hat{f}$ is

$$\mathcal{E}_e(\hat{f}) = \frac{1}{t} \sum_{i=1}^{t} \mathcal{L}(\hat{f}(x_i), y_i) \tag{3.2}$$

3.2 Support vector machine

To explain this next part, we will assume that the data are separable. We will use this example to explain how the support vector machine (SVM) works. We will then extend the principle to linearly non-separable data.

3.2.1 Linearly separable data

Given the setup as in the initial problem, we assume an $\mathcal{X}$ that is linearly separable. The optimal model $\hat{f}$ is classifying the data based on a hyperplane

$$wx - b = 0, \tag{3.3}$$

where $w \in \mathbb{R}^n$ is the normal vector to the hyperplane, called the weight vector. $b \in \mathbb{R}$ defines a hyperplane's offset - bias. This definition of a hyperplane is still not sufficient since we want the best classifier possible. To achieve this, we want to create two hyperplanes

$$y(wx - b) \geq 1, \tag{3.4}$$

where $y \in \mathbb{B}$ so this equation really represents two hyperplanes. In order to have the best classifier possible, maximizing the margin between these two parallel hyperplanes is necessary. The margin between the two hyperplanes is geometrically given by $\frac{2}{|w|}$. Maximizing the margin is the same as minimizing the inverted value of a margin, so we obtain an optimization problem:

$$\begin{aligned} w^*, b^* = && \arg\min_{w,b} \frac{\|w^2\|}{2} && (3.5) \\ \text{subject to} && y_i(wx_i - b) \geq 1. && (3.6) \end{aligned}$$

The minimizer that solves this problem is called the Hard margin support vector machine. The classifier of this type splits the space $\mathbb{R}^n$ into two subspaces and finds in which subspace the data lies: $\hat{f}(x) = \text{sign}(w^* x - b^*)$.

3.2.2 Linearly non-separable data

In the case of data that is linearly non-separable, we extend the Hard margin SVM using the previously mentioned hinge loss. This loss is useful because it gives us information about the distance from the margin. We want to minimize

$$\|w\|^2 + C[\frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_h(wx_i - b, y_i)], \tag{3.7}$$

where $C > 0$ is the hyperparameter defining the trade-off between forcing the x_i to lie on the correct side of the hyperplane and increasing the margin size. We further deconstruct the hinge-loss and introduce the slack variable ς_i to define the update primal optimization problem:

$$\begin{aligned} w^*, b^*, \varsigma^* = && \arg \underset{w,b,\varsigma}{min} \|w\|_2^2 + C\sum_{i=1}^{n} \varsigma_i && \\ \text{subject to} && y_i(wx_i - b) \geq 1 - \varsigma_i, && (3.8) \\ && \varsigma_i \geq 0, \forall_i \in \{1, ..., n\}. && (3.9) \end{aligned}$$

Such minimizer is called the Soft-margin SVM.

3.2.3 Non-linear boundary

The soft-margin SVM is just one step from the approach used in this work later. The primal problem is intuitive but suffers from computational complexity. To overcome these obstacles we further define the Dual problem using Lagrange multipliers. We define Lagrangian:

$$L(w,b,\varsigma,\alpha,\mu) = \frac{\|w\|^2}{2} + C\sum_{i=1}^{n}\varsigma_i - \sum_{i=1}^{n}\alpha_i[y_i(wx_i+b)-1+\varsigma_i] - \sum_{i=1}^{n}\mu_i\varsigma_i. \quad (3.10)$$

Where $\alpha_i, \mu_i \geq 0$ are Lagrange multipliers for constraining hyperplane and slack variables respectively. In order to minimize the Lagrangian we use derivatives $\frac{\partial L}{\partial w}, \frac{\partial L}{\partial b}, \frac{\partial L}{\partial \varsigma_i}$.

$$\begin{aligned}
\frac{\partial L}{\partial w} &= w - \sum_{i=1}^{n}\alpha_i y_i x_i = 0 \\
\frac{\partial L}{\partial b} &= \sum_{i=1}^{n}\alpha_i y_i = 0 \\
\frac{\partial L}{\partial \varsigma_i} &= C - \alpha_i - \mu_i = 0
\end{aligned}$$

Using previous results we can substitute into the 3.10 and obtain the dual problem:

$$\begin{aligned}
\alpha_i^* = \arg\max_{\alpha_i} \sum_i \alpha_i - \frac{1}{2}\sum_{i,j}^{|\mathcal{X}|}\alpha_i\alpha_j y_i y_j k(x_i,x_j) \quad (3.11) \\
\text{subject to } \sum_i y_i\alpha_i = 0 \\
0 \leq \alpha_i \leq C, \forall i..
\end{aligned}$$

An important thing to mention here is the kernel function $k(x_i, x_j)$. Its introduction is seemingly unnecessary, but its rationale is so important that we want to emphasize it. As we mentioned earlier, the data may be linearly separable. In this case, the soft-margin SVM can utilize $k(x_i, x_j) = x_i^T x_j$ - simple inner product. The importance of the kernel function k arises in the case of linearly non-separable data. We can use some transformation to map the data to a higher dimensional feature space, where the data becomes linearly separable - an example of such approach is radial basis function [AC02]. In this higher dimensional space, we can find a hyperplane separating the data. If this hyperplane is mapped back to the original space, it becomes non-linear. This is called the *kernel trick* and it is the key essence of using SVMs. Mapping to a feature space using a feature map is defined in the following section.

3.3 Feature map

The kernel function k that appeared in the previous section is closely related to the feature map, as mentioned in the linearly separable case $k(x_i, x_j) = x_i^T x_j$. But to fully utilize the kernel trick, we wish to create a feature map that will map our data to the higher dimensional feature space $\mathcal{F}$ whose dimension may be arbitrary, even infinite. To achieve this we introduce the feature map $\phi(x) : \mathbb{R}^d \to \mathcal{F}$, the kernel function then corresponds to the inner product in the feature space $k(x_i, x_j) = \langle x_i | x_j \rangle$.

3.3.1 Quantum feature map

The quantum computer steps in right at this time. As we mentioned earlier, the quantum computer state lives in the Hilbert space $\mathcal{H}$ whose dimension grows exponentially with the number of qubits m. To use classical data in a quantum computer, we need to map this data to the quantum circuit. For this purpose, we can incorporate a quantum circuit with the quantum embedding $\phi(x)$. The quantum embedding is further described in section 7. For this moment we only need to know that it is a map $\phi(x) : \mathbb{R}^d \to \mathcal{F}$.

Definition 3.3 (Quantum feature map)**.** Let x be data from data space $\mathcal{X}$ that we want to encode into the quantum circuit. $\mathcal{F}$ is the space of complex density matrices $\mathbb{C}^{2^q \times 2^q}$. The circuit realizing this embedding prepares the quantum computer in the state:

$$|\phi(x)\rangle = E(x) \, |0^q\rangle \tag{3.12}$$

where $E(x)$ is a unitary operator defined based on data x, and q is the number of qubits.

3.4 Quantum kernel

We can further introduce a quantum kernel once we can map classical data to a quantum computer. Quantum kernels are the key components of this work. This feature map gives rise to a kernel-distance metric based on the dot product.

Definition 3.4 (Quantum kernel)**.** Let ϕ be a quantum feature map. Then quantum kernel $\kappa(x_i, x_j) \in \mathbb{R}$ is the fidelity between two data encoding feature vectors $\phi(x_i)$, $\phi(x_j)$:

$$\kappa(x_i, x_j) = \mathrm{tr}[\rho(x_i)\rho(x_j)]^2, \tag{3.13}$$

which for the case of mixed states simplifies:

$$\kappa(x_i, x_j) = \|\langle\phi(x_i)|\phi(x_j)\rangle\|^2. \tag{3.14}$$

Subsequently, we define a kernel matrix K with rows and columns corresponding to fidelity between states raised by individual samples:

$$K(\mathcal{X})_{ij} = \kappa(x_i, x_j). \tag{3.15}$$

This kernel matrix $K(\mathcal{X})$, commonly referred to as Gram matrix, is positive definite.

3.5 Projected quantum kernels

Quantum kernels often exhibit an issue of being exponentially concentrated. [HBM+21] Meaning that the kernel values outside the kernel diagonal are vanishingly small. Support vector machines suffer from this, so they cannot learn the function properly. To ease this problem, a family of projected quantum kernels is used. Projected quantum kernels are an easy concept of measuring only *some part* of qubits. This step is executed by t-reduced density matrix:

$$\rho_t(x) = \mathrm{tr}_t[\rho(x)], \tag{3.16}$$

which can be considered as a measurement on just a subset of qubits $t \subset q$, with tr_t being a partial trace over qubits t. The kernel entries then are:

$$\kappa_t(x_i, x_j) = \mathrm{tr}_t[\rho(x_i)\rho(x_j)]. \tag{3.17}$$

Results from these projected quantum kernels can be used as features in traditional SVMs.

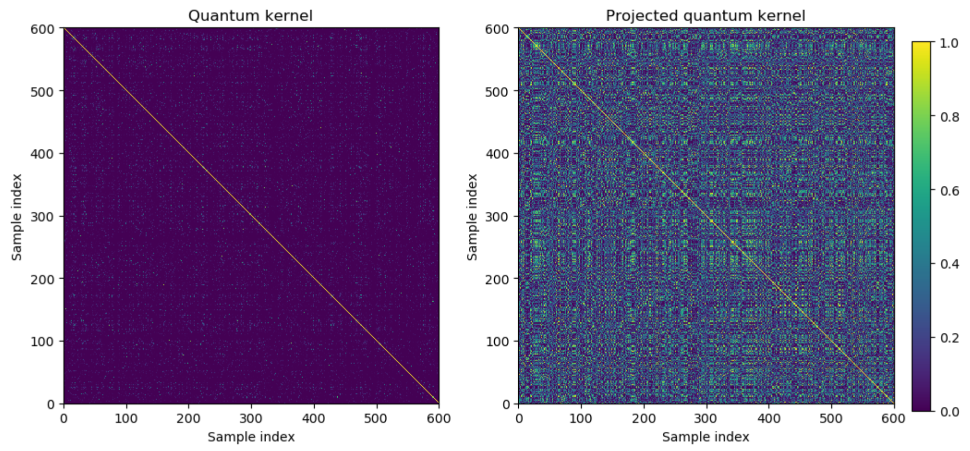


Figure 3.1: Quantum kernel and Quantum projected kernel comparison

3.6 Synthesis

With the quantum kernel $\kappa(.,.)$ being defined we can plug this into the 3.11. As mentioned, $k(x_i, x_j)$ can have numerous forms, let us name the Gaussian kernel function $k(x_i, x_j) = e^{-\gamma||x_i - x_j||_2^2}$, the radial basis function $k(x_i, x_j) = e^{-\frac{||x_i, x_j||^2}{2\sigma^2}}$, or even the linear kernel function $k(x_i, x_j) = \langle x_i | x_j \rangle$. But in the previous section, we showed that a quantum computer armed with data embedding $\phi(x)$ naturally defines its own kernel function $k(x_i, x_j) = \kappa(x_i, x_j) = \langle \psi(x_i) | \psi(x_j) \rangle$. The subject of the next section is the use of quantum kernel and its properties.

Chapter 4

Kernel quality measures

An essential part of this thesis is determining the quality of a kernel. We don't want to jump forward, but the main idea is to basically generate a lot of kernels and then keep just a fraction of those that acquire a certain quality. This chapter dives into the realm of kernel quality measures and it introduces principles on which we decide whether the kernel we have is useful or not. We use the kernel target alignment [CSTEK01], model complexity [HBM$^+$21], the asymmetric geometric difference [HBM$^+$21] , and the eigenvalues ratio [LL15].

4.1 Kernel target alignment

Definition 4.1 (Kernel target alignment)**.** The kernel target alignment $KTA(k, f)$ is then defined as follows:

$$KTA(k,f) = \frac{\langle k, f \otimes f \rangle}{\langle k, k \rangle^{1/2} \langle f \otimes f, f \otimes f \rangle^{1/2}} = \frac{\sum_i \gamma_i \alpha_i^2}{(\sum_i \gamma_i)^{1/2} \sum_i \alpha_i^2}, \tag{4.1}$$

where k is the quantum kernel as defined and f is the target function.

Since this measure may be the sum of infinite dimensions we can use a more straightforward definition that utilizes the Gramm matrix K belonging to a kernel, and the so-called target matrix K^*

$$K^*_{ij} = y_i y_j. \tag{4.2}$$

Kernel target alignment then takes the form of:

$$KTA(K, K^*) = \frac{\langle K|K^*\rangle_F}{\sqrt{\langle K^*|K^*\rangle_F \langle K|K\rangle_F}}, \tag{4.3}$$

which is the notion that will be used throughout this work. $\langle K_1, K_2\rangle_F = \sum_{i,j=1}^{n} K_1(x_i, x_j)K_2(x_i, x_j)$ is the Frobenius inner product of two matrices. Kernel target alignment can be interpreted as a measure of similarity based on the cosine of the angle. From such knowledge we already know the range of output for general matrices - resulting in $KTA(A, B) \in [0, 1]$.

4.2 Asymmetric geometric difference

Asymmetric geometric difference is the first obstacle a quantum kernel needs to overcome in order to achieve an advantage over its classical counterpart [HBM+21]. It is defined as follows:

$$g_{ab} = g(K_a||K_b) = \sqrt{||\sqrt{K_b}K_a^{-1}\sqrt{K_b}||_\infty}, \tag{4.4}$$

where $||\cdot||_\infty$ is the spectral norm of a matrix. K_a and K_b are corresponding kernels. It is important to mention the assumption that $Tr(K_a) = Tr(K_b) = N$, which can be enforced using regularization. The first sign of a step towards quantum advantage is when g_{cq} (the asymmetric geometric difference between a classical K_c and a quantum kernel K_q) is proportional to $q_{cq} \propto \sqrt{n}$.

4.3 Model complexity

Model complexity is bound to the asymmetric geometric difference. If the condition of $q_{cq} \propto \sqrt{n}$ is met, we wish to examine each of the kernels separately by finding its model complexity:

$$s_K(n) = \sum_{i,j=1}^{N} \frac{t(x_i)t(x_j)}{K_{ij}}, \tag{4.5}$$

where $t(x) = Tr(O\rho(x))$. The models' complexity is an important measure since smaller values demonstrate a better generalization to new data, while bigger values indicate overfitting. Informally, model complexity tells us whether the distance in feature space is connected to the kernel function value $k(x_i, x_j)$. Quantum kernels exhibit advantage when the asymmetric geometric difference is proportional to the training dataset n, the quantum model complexity is much smaller $s_Q \ll n$, and the classical model complexity is proportional to $s_C \propto n$ [HBM+21].

4.4 Eigenvalues ratio

The eigenvalues ratio is of simple definition, but it helps find the best possible kernel [LL15]. We define t-th eigenvalues ratio β_t:

$$\beta_t(K) = \frac{\sum_{i=1}^{t} \lambda_i}{\sum_{i=t+1}^{n} \lambda_i}, \tag{4.6}$$

where λ_i is the i-th eigenvalue of the Gramm matrix K corresponding to the kernel we wish to inspect, we assume the set of eigenvalues $\{\lambda_i\}_i$ to be of ascending order. The eigenvalues ratio is the ratio between the main eigenvalues and the tail values. Lower values of $\beta_t(K)$ indicate better performance of a particular kernel. Of course, during optimization, we need to find the optimal value t which will also be part of the optimization.

Chapter 5

Hardness of random circuits

Before we dive into the next section, which acquaints us with the main idea of this work and our way of thinking about quantum advantage, let us build the last pillar of its foundations - hard classical simulability. More precisely, the hard classical simulability of a quantum kernel $\kappa(x_i, x_j)$. We want a quantum computer to have a head start over a classical computer while evaluating kernel function. We take advantage of using random unitaries and their capacity of being hardly simulable using classical computing. In this section, we propose findings from [Mov20]. Let us introduce several concepts that are the foundation of the theorems that are utilized later.

5.1 $\#P$ hardness

We later classify that quantum kernel $\kappa(x_,x_j)$ is formidably hard to simulate on classical computers. The task of simulating kernel output lies in the $\#P$ complexity class, which is part of NP. The complexity class NP is defined as the set of decision problems for which a solution can be verified in polynomial time by a deterministic Turing machine. Roughly said, the NP complexity class asks whether a solution exists to our problem. $\#P$ complexity class goes deeper and answers the question of *how many* solutions exist.

Definition 5.1 ($\#P$ [AB06])**.** $\#P$ is the set of all functions $f : 0, 1^* \to \mathbb{N}$ such that there is a polynomial time nondeterministic Turing machine T for all $x \in 0, 1^*$, $f(x)$ equals the number of accepting branches in T's computation graph on x.

Intuitively, this complexity set lies in NP. Later on, we will prove that estimating a quantum kernel, which uses random unitaries, is $\#P$ hard for classical computers.

5.2 Circuit architecture

Circuit architecture is mentioned throughout this section and is essential to define. First, we assume that each circuit has q qubits. 1-qubit and 2-qubit gates can be applied to these qubits. Regarding the fact that any quantum computation can be translated to a circuit using solely universal gates, [Wil11], we can see this model as a general instance of a quantum computing model. Architecture $\mathcal{A}$ refers to an arbitrary layout of m gates in the circuit. At this point, we define gates only as black boxes, and when we mention architecture $\mathcal{A}$, we don't consider the gates' definition. So when talking about architecture $\mathcal{A}$, we mean something like in figure 5.1, some sort of a blueprint. We can say that architecture $\mathcal{A}$ with a description of each gate defines a particular quantum circuit. The architecture becomes a circuit

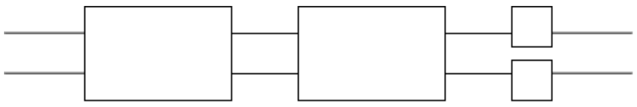


Figure 5.1: Circuit architecture $\mathcal{A}$ blueprint

once the gates are defined using unitaries. The circuit itself can be described by a single unitary given the equation:

$$C = C_m C_{m-1} \dots C_1, \tag{5.1}$$

Where C_i is a single gate. If a gate is applied only to 1 qubit, the unitary is $C_i = C_{\hat{i}} \otimes \mathbb{I}$, where $C_{\hat{i}}$ is simply the 1-qubit gate, which means it does nothing to the second qubit.

5.3 Worst-case and average-case circuits

In the literature [TD04] it has been proven that for architecture with depth $m = 4$, a worst-case circuit exists, for which to classically approximate a quantity $p = |\langle 0^n|C|0^n\rangle|^2$ is #P-hard up to constant relative error. However, talking about worst-case circuits is not sufficient to achieve quantum supremacy. Quantum supremacy expects that estimating probability distribution induced by the random circuit is computationally hard for any classical algorithm that inputs the classical description of the gates. [Mov20] We need to extend this finding to *most* circuits. Thus, we introduce the idea of an average case circuit. This circuit is generated completely at random by the QR decomposition of a random Gaussian matrix.

5.4 Haar random circuit distribution

The Haar random circuit distribution is defined on architecture over circuits $\mathcal{A}$. We have the Haar random circuit distribution $\mathcal{H}_{\mathcal{A}}$ over circuits $\mathcal{A}$ whose local gates are independently drawn from the Haar measure.

5.4.1 Sampling from Haar measure

For generating random unitaries $U \in \mathbb{C}^{2^q \times 2^q}$ QR decomposition approach is utilized [Mez07]. The procedure is as follows:

1. Generate $2^q \times 2^q$ matrix Z with complex entries such that both real and complex parts are normally distributed with mean 0 and variance 1
2. Compute a QR decomposition of $Z = QR$
3. Generate diagonal matrix $\Lambda = \operatorname{diag}(R_{ii}/|R_{ii}|)$
4. Compute $U = Q\Lambda$, which is a random unitary.

5.5 Cayley transformation and Cayley path

The Cayley transformation is a projective map

$$f(\theta) : \mathbb{R} \to \mathbb{C} : \frac{1 + i\theta}{1 - i\theta} \tag{5.2}$$

that maps a line of points to a unit circle. This function is used when transforming between two unitary matrices through a path $\theta \in [0, 1]$. Following the definition of Cayley transformation, we further introduce:

$$H = \tau(h) = \sum_{\alpha=1}^{N} f(\lambda_\alpha) \langle \psi_\alpha | \psi_\alpha \rangle \,, \tag{5.3}$$

where H is a unitary matrix generated from a hermitian matrix h, λ_α and ψ_α are the eigenvalues and eigenvectors of h. The generated unitary matrix H is of Haar measure and it represents the previously mentioned average case circuit. Lastly, we want to define the Cayley path, which is parameterized by the parameter $\theta \in [0, 1]$:

$$C(\theta) = W\tau(\theta h), \tag{5.4}$$

where W is the fixed worst-case circuit and h is a randomly generated hermitian matrix generating $H = \tau(\theta h)$ which is of average case circuit instance. From this equation, we can point out the fact that this Cayley path truly oscillates between the worst-case circuit W and the average-case circuit with $W\tau(0) = W\mathbb{I} = W$ and $W\tau(1) = WH$. $W\tau(1)$ is a random Haar unitary due to the left-translation invariance, thus an instance of an average case circuit. This is an important concept worth stressing. What Cayley

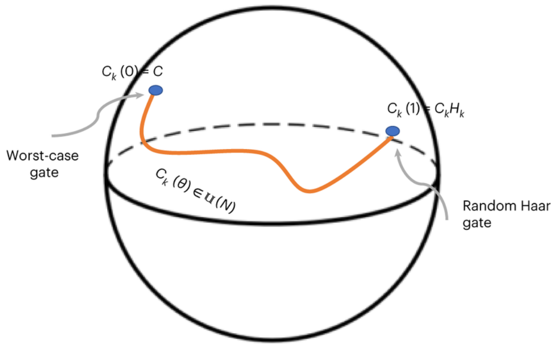


Figure 5.2: Cayley path

transformation does is that it makes the worst-case circuit and random-case circuit defined on the architecture $\mathcal{A}$ equivalent. The equation for a circuit unitary becomes:

$$C(\theta) = W_m \tau_m(\theta) W_{m-1} \tau_m(\theta) \ldots W_1 \tau_1(\theta). \tag{5.5}$$

We assume that $\tau_i(1)$ is a unitary matrix according to the Haar measure. Then, the distribution over $W_m \tau_i(\theta)$ for $|1 - \theta| \leq \Delta \ll 1$ is $\mathcal{O}(\Delta)$-close to the Haar in total variation distance.

5.6 Theorems

I will introduce the main results presented in [Mov20].

Theorem 5.2 (Hardness of random quantum circuits (informal) [Mov20])**.** *Suppose there exists an architecture $\mathcal{A}$ for which it is $\#P$ hard to compute arbitrary output probabilities within a small multiplicative error. Then it is $\#P$ hard to calculate the probability amplitude for most circuits with the same architecture within $\epsilon = 2^{-\Omega(m)}$ where m is the number of gates.*

This states that, indeed, most circuits have the property that their amplitudes are $\#P$ hard to calculate. The reasoning behind this theorem is that if there exists a classical algorithm that efficiently computes $p_0(\theta) = |\langle 0^n|C|0^n\rangle|^2$ for $\theta \approx 1$ - an instance of an average-case circuit, then we could call this algorithm $poly(n)$ times on θ_i, and with the use of the Berlekamp-Welch algorithm we obtain $p_0(\theta)$,$\forall\theta$ [Mov20]. However, this would cause a collapse of the polynomial hierarchy, which means that there cannot be such a classical algorithm. This theorem is built upon the foundation of the two following ones. It merges both of them while making them more tractable.

Theorem 5.3 ($\#P$ hardness of Haar random circuits [Mov20])**.** *Let $\mathcal{A}$ be an architecture such that computing $p_0 = |\langle 0^n|C|0^n\rangle|^2$ is $\#P$ hard in the worst-case. Then, it is $\#P$ hard to output $|\langle 0^n|C|0^n\rangle|^2$ with the probability $\alpha = \frac{3}{4} + \frac{1}{poly(n)}$ over the choice of circuits $H \in \mathcal{H}_{\mathcal{A}}$.*

The proof of this theorem is based on the fact that given the Berlekamp-Welch algorithm, outputting the rational function $p_0(\theta)$ is possible when given a sufficient number of measurements $|\theta_i|_i = poly(n)$. However, this fact implies that BPP $= \#P$. This is highly unlikely. To mention the second theorem once again, we need to define a new concept, which is the classical algorithm $\mathcal{U}$. This algorithm has the following property:

$$Pr[|\mathcal{U}(C(z_i)) - p_0(z_i) \leq \epsilon] = 1 - \frac{1}{poly(n)}; |z_i| \leq \Delta, \tag{5.6}$$

where $z_i = 1 + \theta_i$, a parameter defining Cayley path and Δ is defined as the upper bound on the interval on which we take θ_i: $|1 - \theta_i| \in [0, \Delta]$.

Theorem 5.4 (Robustness of $\#P$ hardness [Mov20])**.** *Assuming access to an oracle $\mathcal{U}$ as described above, it is $\#P$ hard to compute $p_0(C(\theta))$ over $\mathcal{H}_{\mathcal{A}}$ within $\epsilon = 2^{-\Omega(m^2)}$ additive error.*

5.7 Summary

In this section, we have highlighted the results [Mov20], which show that using random circuit sampling, we can generate circuits that are $\#P$ hard to simulate classically. The importance of the aforementioned is depicted in the next section, where we finally get to the core of the thesis.

Chapter 6

Main concept

Moving from the previous section, we can finally show the main concept of this work. The key idea is to construct the kernel $\kappa_R(x_i, x_j)$ such that the kernel output is $\#P$ hard to evaluate on a classical machine. We intend to use this kernel output with a classical support vector machine and solve the initial problem 3.1. Hence, the procedure employs a quantum computer to evaluate the kernel $\kappa_R(x_i, x_j)$ and a classical computer to solve the optimization task 3.11.

6.1 Possible advantages

Our approach to this problem offers not only one but four advantages.

6.1.1 Kernel simulability

A kernel is constructed as follows:

$$\kappa_R(x_i, x_j) = \langle \psi(x_i) | U | \psi(x_j) \rangle \,, \tag{6.1}$$

with U being a randomly generated unitary. We claim that simulating the output of this kernel on a classical computer is $\#P$ hard. As proof, we use the previously mentioned theorems 5.3 and 5.4. Since these proofs take into account the estimating probabilities of $|\langle 0^n|C|0^n\rangle|^2$, we need to extend it to our case, where we wish to estimate the probabilities $|\langle\psi(x)|C|\psi(y)\rangle|^2$. The proof is fairly simple. If we assume the fact that estimating $|\langle 0^n|C|0^n\rangle|^2$ is $\#P$ hard, we can rewrite the

$$\begin{aligned} |\langle\psi(x)|C|\psi(y)\rangle|^2 &= |\langle 0^n|E^\dagger(x)CE(y)|0^n\rangle|^2 \qquad (6.2)\\ &= |\langle 0^n|C|0^n\rangle|^2. \qquad (6.3)\end{aligned}$$

This means that only by applying unitary operations (which we know from the data, and we can easily compute their conjugate transpose) we can translate our problem to estimating $|\langle 0^n|C|0^n\rangle|^2$. If estimating $|\langle\psi(x)|C|\psi(y)\rangle|^2$ wasn't $\#P$ hard, estimating $|\langle 0^n|C|0^n\rangle|^2$ would not be $\#P$ hard either.

6.1.2 Time complexity

In addition to the kernel being $\#P$ hard to evaluate classically, it also enables exponential speed-up. If we are able to prepare a quantum system $|\psi(x)\rangle$ in linear time $\mathcal{O}(n)$, we can take advantage of the ability of quantum computers to compute an inner product in constant time $\mathcal{O}(1)$ and therefore evaluate the Gram matrix $K(\mathcal{X})$ in time $\mathcal{O}(n)$ instead of $\mathcal{O}(n^2)$ as on a classical computer. [Sch21]

6.1.3 Projected quantum kernels

On top of that, we may employ the projected kernels κ_R^P with corresponding Gram matrices $\mathcal{K}(\mathcal{X})$ having eigenvalues satisfying these rules [KBS21]:

- One eigenvalue $2^{-p} + \mathcal{O}(2^{-2q})$ with constant eigenfunction
- $2^{2p} - 1$ eigenvalues $2^{-p-q} + \mathcal{O}(2^{-2d})$,

where p and q follow notation and represent the number of measured qubits in the projected kernel and the number of qubits in the circuit, respectively.

6.1.4 Random sampling

Using a sampling of U, which is a random unitary, we wish to achieve much better spectral properties 4. The main idea is to sample random Haar unitaries for a sufficient number of trials to strictly improve kernel qualitative measurements such as kernel target alignment or eigenvalue ratio. We use rejection sampling, throw away unitaries that do not improve the model's performance, and keep only the promising ones.

6.2 Experiment algorithms

We prepared several experiments to support our findings. The experiment procedure goes as follows:

Algorithm 1 Classifying data

```
                         Inputs:
                  Labelled data $(x_i, y_i)$
                  Unlabelled data $(\hat{x}_i)$

                         Output:
        Unitary $U$ increasing accuracy of prediction
                Classified data $(\hat{x}_i, \hat{y}_i)$
procedure TRAINING
    for all $x_i$ do
        $\psi_i \leftarrow E(x_i)\,|0^n\rangle$          ▷ We obtain state vectors of mapped data
    end for
    $t \leftarrow 0$
    while $t = maxiter$ do                 ▷ Boolean variable for train ending
        $C \leftarrow$ random unitary
        for all $\psi_i$,$\psi_j$ do
            $K_{ij} \leftarrow \langle\psi_i|C|\psi_j\rangle$                        ▷ We get kernel matrix
        end for
        $KTA_C \leftarrow KTA(K, y)$             ▷ Evaluate kernel-target alignment
        $MC_C \leftarrow KTA(K, y)$                  ▷ Evaluate model complexity
        $AGD_C \leftarrow KTA(K, y)$   ▷ Evaluate asymmetric geometric distance
        $ER_{kC} \leftarrow KTA(K, y)$               ▷ Evaluate $k$-eigenvalue ratio
        $t \leftarrow t + 1$
    end while
end procedure
procedure EVALUATING
    for all $\hat{x}_i$ do
        $\hat{\psi}_i \leftarrow E(\hat{x}_i)\,|0^n\rangle$
        $\hat{y}_i \leftarrow 0$
        for all $\psi_j$ do
            $\hat{y}_i \leftarrow \hat{y}_i + \langle\psi_j|U|\hat{\psi}_i\rangle$
        end for
        $\hat{y}_i \leftarrow \text{sign}\,\hat{y}_i$
    end for
end procedure
```

Chapter 7

Embeddings

Embedding plays a key role in mapping classical data to a quantum circuit. For each of the embeddings, we describe its functionality and also present the kernel it naturally generates. All the approaches mentioned below will then be used in the experiments in the next section. It is important to mention that the role of embedding is often overlooked, and not many works are dedicated to this issue. Among the exceptions, we should mention for example: Maria Schuld. We have drawn on the insights contained in the aforementioned works in the construction of this chapter. We believe that testing embeddings against each other on different types of data can help future work push quantum computing towards better results.

7.1 Basis embedding

One of the simplest embeddings is basis embedding. Basis embedding takes a discrete input and encodes it in the quantum circuit. In the literature [**Sch21**] it is often defined as binary basis embedding but can, in general, use any numeral system. The idea behind basis embedding is simple - each digit is mapped to its predefined quantum state. The only restriction is that the set of quantum states we map to is orthogonal. For example, we can use the aforementioned binary basis embedding. We choose an arbitrary orthogonal basis $\{|0\rangle, |1\rangle\}$. The binary string $x \in \mathbb{B}^n$ is then mapped to the quantum state:

$$\psi(x) = \bigotimes_{i=1}^{N} m(x_i), \tag{7.1}$$

where $m(x_i)$ is the mapping function:

$$m(x_i) = \begin{cases} |0\rangle & \text{if } x_i = 0, \\ |1\rangle & \text{if } x_i = 1. \end{cases} \tag{7.2}$$

As mentioned, we can use an arbitrary set of orthogonal basis states, e.g. $\{|+\rangle, |-\rangle\}$, $\{|l\rangle, |r\rangle\}$. What is more, we are not bound only to binary string inputs but can choose other numeral systems with a simple example being the quaternary system - with $x = \{0,1,2,3\}^n$, and the orthogonal basis being $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$. Basis embedding generates a kernel corresponding to Kronecker delta:

$$\kappa(x, x') = \delta_{x,x'}. \tag{7.3}$$

7.2 Amplitude embedding

Another possible, and a bit more sophisticated, embedding is amplitude embedding [Sch21]. It maps N-dimensional data $\mathcal{D}(x) \in \mathbb{C}$ such that each dimension of input gets mapped to a quantum state. Mapping can be defined via the equation:

$$|\psi(x)\rangle = \frac{1}{|x|} \sum_{i=1}^{N} x_i |\phi_i\rangle, \tag{7.4}$$

where ϕ_i is from an orthogonal basis of the quantum space. For example, data $x = (0,3,0,4)$ then gets mapped to the state $\psi(x) = \frac{1}{\sqrt{5}}(3|01\rangle + 4|11\rangle)$. Amplitude embedding generates a kernel corresponding to the absolute square of a linear kernel:

$$\kappa(x, x') = |x^\dagger x||^2 \tag{7.5}$$

Since this embedding is not defined by a set of gates, but rather a unitary operator, obtaining a unitary that maps state $R|0^n\rangle \to |\psi(x)\rangle$ is important. Generating such unitary is done via the Householder transformation [Mez07]. The equation of the projector R is:

$$R = \mathbb{I} - 2|v\rangle\langle v| \tag{7.6}$$

where $|v\rangle = |0^n\rangle - |\psi(x)\rangle$ is the difference between the input and the desired state.

7.3 Rotation embedding

Rotation encoding can encode data $\mathcal{X} \in \mathbb{R}^n$ [Sch21]. The data should be precomputed in a way that each $x_i \in \mathcal{X}$ is in the interval $\langle 0, 2\pi)$. For this data, each i-th component x_i gets mapped to the superposition of $|0\rangle$ and $|1\rangle$ regarding its value. The embedding can be described by the equation:

$$|\psi(x)\rangle = \bigotimes_{i=1}^{N} \cos(x_i)\,|0\rangle + \sin(x_i)\,|1\rangle\,. \tag{7.7}$$

Again, as in the example of basis embedding, this mapping can be slightly changed, and the resulting state can be a superposition of different basis ($\{|+\rangle, |-\rangle\}$, etc.) The corresponding kernel of rotation embedding is the cosine kernel:

$$\kappa(x, x') = \prod_{i=1}^{n=|X|} |\cos(x_i - x_i')|^2 \tag{7.8}$$

7.4 Pauli feature map

A much more sophisticated encoding feature map is a family of Pauli feature maps. These transform data $x \in \mathbb{R}^n$, where n is the feature dimension, as:

$$U_{\Phi(x)} = exp(i \sum_{S \in \mathcal{I}} \psi_S(x) \prod_{i \in S} P_i), \tag{7.9}$$

where S is a set of qubit indices that describes the connections in the feature map (e.g. ZZ, Y, ZY), $\mathcal{I}$ is a set containing all these index sets and $P_i \in \{I, X, Y, Z\}$. The data mapping ψ_s is defined as:

$$\psi(x) = \begin{cases} x_i & \text{if } S = \{i\} \\ \prod_{j \in S}(\pi - x_j) & \text{if } |S| > 1 \end{cases} \tag{7.10}$$

Some examples include using $P = Y$ or $P = Z$ in one qubit setting. These map the interval $[-\pi, \pi]$ to the corresponding states visualized on the Bloch sphere:

The corresponding colors of state vectors correspond to the gradient between green ($x = -\pi$) and blue ($x = \pi$) color.

7.5 ZZ-feature map

A very famous embedding throughout the quantum computing community is the 2-qubit ZZ-feature map [HCT+19]. This embedding is part of the Pauli

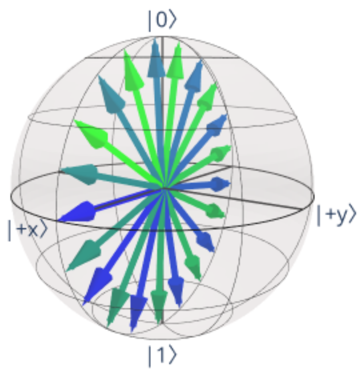


Figure 7.1: Pauli Y feature map

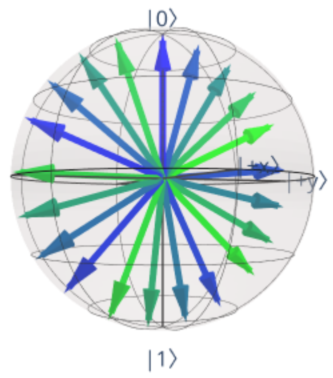
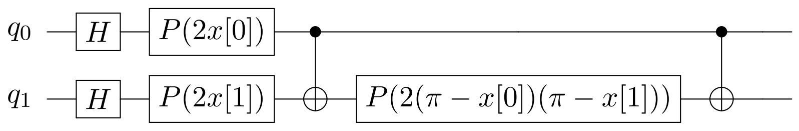


Figure 7.2: Pauli Z feature map

feature map family, it maps the data with the set of single qubit rotations $P(\lambda)$ about the Z axis depending on data and a set of $CNOT$ gates. $P(\lambda)$ is defined as:

$$P(\lambda) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\lambda}. \end{pmatrix} \tag{7.11}$$

A circuit realizing a ZZ-feature map is then depicted in the figure below:



(7.12)

with $(x_0, x_1) = x \in \mathcal{X}$.

7.6 Repeated embedding

Another option is to use repeated embeddings. These work simply by applying embedding circuits one after the other. The resulting kernel corresponds to the original kernel raised to the power of the number of repetitions. For example, considering an amplitude embedding with 3 repetitions, the resulting kernel is:

$$\kappa(x, x') = (|x^{\dagger}x\|^2)^3 \tag{7.13}$$

Chapter 8

Experiment

In this section, we introduce findings based on experiments. Experiments were run on two datasets - one trivial, the Moon dataset from *sklearn* [PVG+11] and one rather challenging, the Pima indians diabetes dataset [Rep16], with the best algorithms performing only a $\approx 82\%$ success rate of classifying the data. The goal of the experiments was to use a quantum computer to evaluate kernels and a classical computer to use these kernels to train an SVM classifier. The obtained quantum kernels were then classified based on quality measures introduced in 4. The SVM classifier was run with parameter 10^4 iterations. All experiments were run in the *Qiskit Aer* simulator.

8.1 Preproccessing

Since the Pima dataset sometimes contains *NaN* values, we have replaced those with mean values of corresponding features, which we believe does not harm the experiments but enables encoding data into the circuit.

In case of amplitude encoding, each sample was normalized. While using Pauli feature maps, we scaled the training dataset across all features to the interval $[0, 2\pi]$ since otherwise encoding data may suffer from being periodic.

8.2 Moons dataset

The Moons dataset consists of samples with 2 features - x and y coordinates. Our training and test datasets consisted of 70 and 30 samples respectively. We used several approaches for embedding the data into the circuit:

- XY-feature map
- XY-feature map with two repetitions
- ZZ-feature map
- ZZ-feature map with two repetitions
- Amplitude encoding

Because the Moons dataset can be encoded into the circuit of just 2 qubits, we were able to run 50 experiments for each embedding, with each experiment using a different random unitary U. Since we used only 2 qubits we did not use projected kernels as they should not be behaving much differently. From the results 8.1 we can see that there is a clear connection between kernel target alignment and kernel accuracy, but we observe the fact that not all embeddings offer the same efficiency. Another measure we evaluated was

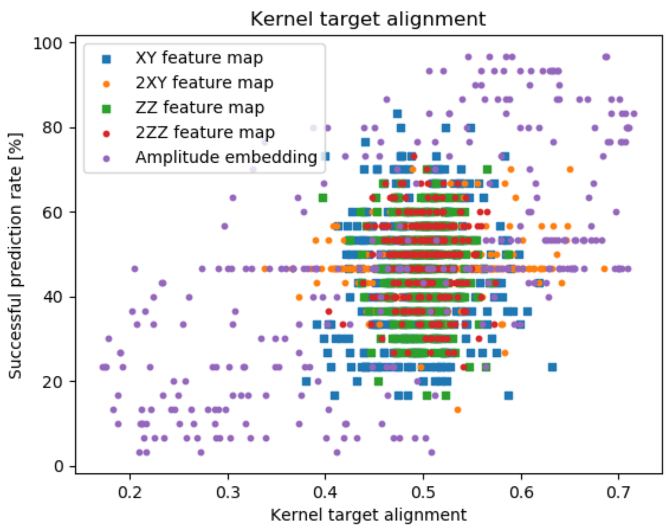


Figure 8.1: Kernel target alignments for Moons dataset

t-th eigenvalue ratio of matrix $K^{70\times70}$. For this, we used several settings ending with $t = 10$, which exhibited the biggest impact across all possible

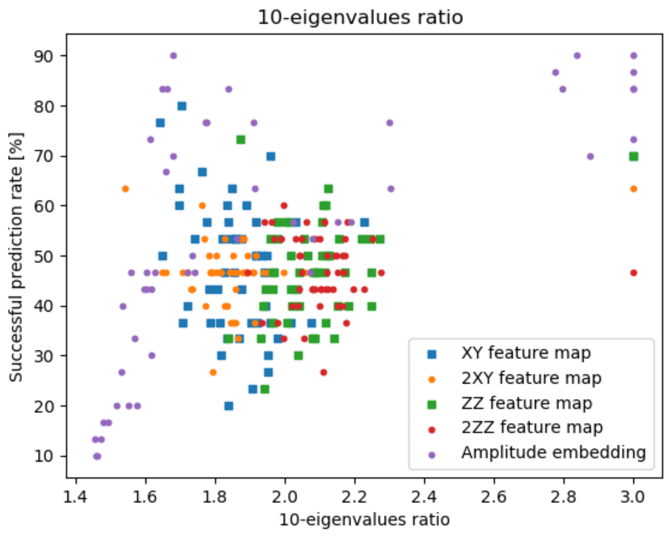


Figure 8.2: t-th eigenvalue ratio for Moons dataset

options, but still lacked the desired behavior of determining kernel capabilities definitely. We then used the classical radial basis function SVM to evaluate the optimal 'classical' Gramm matrix. The classical SVM was able to classify all samples in the test dataset correctly, so we could compare this kernel to the quantum ones. Unfortunately, we did not observe any correlation between the quality measures from [HBM+21] and kernel's performance.

8.3 PIMA dataset

As the second dataset we chose the Pima indians diabetes dataset. This dataset consists of 768 samples with 8 features per sample. We chose this dataset because it is hardly classifiable and online resources [Rep16] claim best performance of classical methods $\approx$ 82% offering enough space for improvement. We split the dataset into a 70% training and a 30% testing dataset, and tested several embedding approaches.

- XY-feature map
- ZZ-feature map
- Amplitude encoding
- Projected XY-feature map
- Projected amplitude encoding

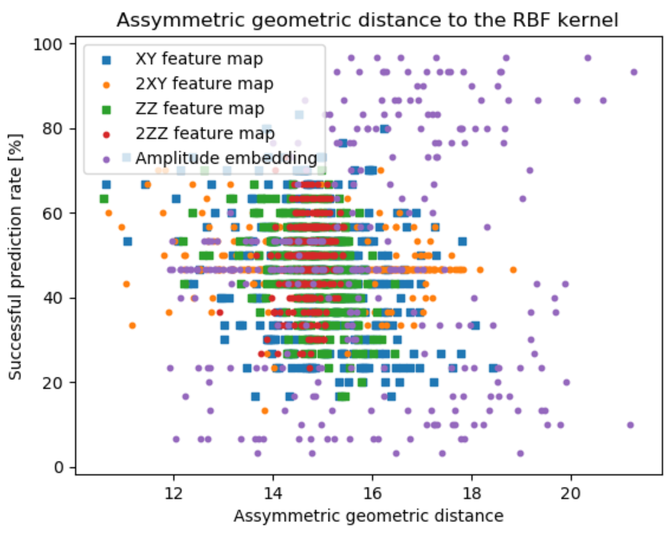


Figure 8.3: Asymmetric geometric distance of Kernels

With Pauli embeddings executed on 8 qubits, we can see that the embeddings that are not projected lack in performance and are overtaken by projected kernels. We emphasize this since it is the result of the kernel matrix being concentrated around the matrix diagonal as mentioned in 3.1. Results of using t-th eigenvalue ratio were disappointing as the performance was not correlated to the measure at all. Using the asymmetric geometric distance and model complexity did not yield good results either.

8.4 Discussion

In this section, we have introduced our experiments that have showed somewhat sobering results. From the introduced quality measures that could be used to randomly sample kernels using a quantum computer, only kernel target alignment and t-th eigenvalue ratio appeared to be of any use. However, it is at least clear from the experiments that different embeddings will be ideal for different datasets. In our case, the amplitude embedding was the best choice for both datasets, but this may not always be the case. We should also note that the experiments were not as extensive as would have been desirable, and we can assume that with an increased number of samples we would have found more random unitary matrices that would have improved the kernel performance. Nevertheless, it is important to point out that random unitary operations can indeed be useful, as they can turn the kernel structure into a position where the kernel shows signs of high performance.

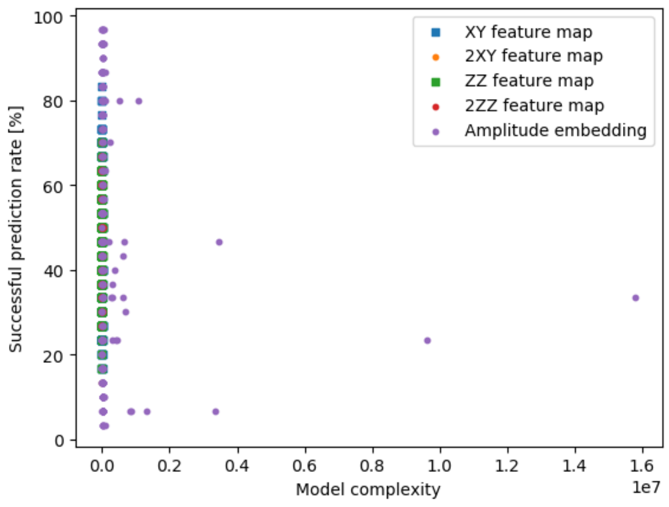


Figure 8.4: Model complexity of each kernels

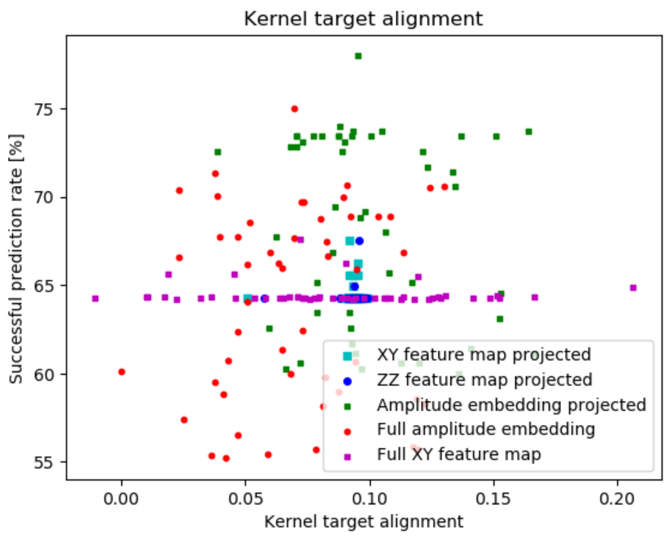


Figure 8.5: Kernel target alignments for Pima dataset

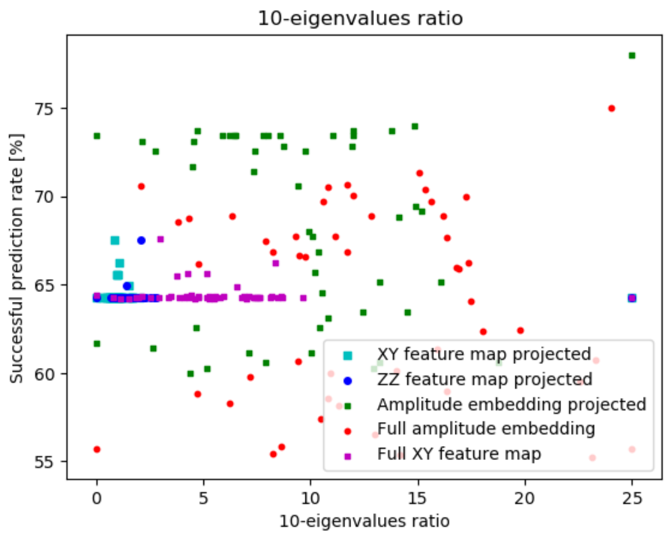


Figure 8.6: t-th eigenvalue ratio for Pima dataset

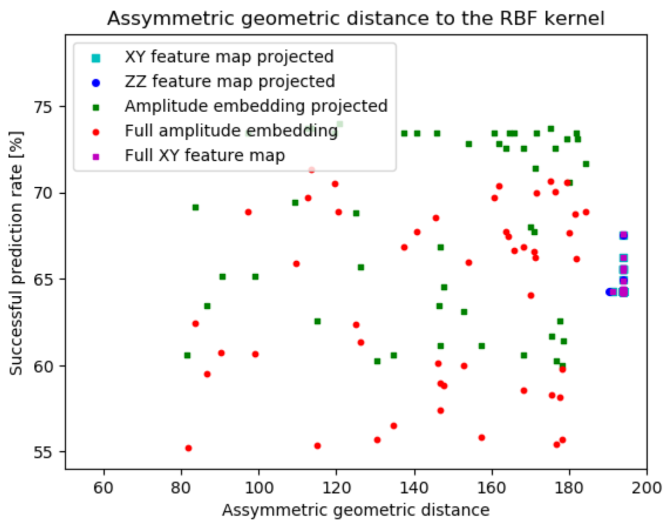


Figure 8.7: Asymmetric geometric distance of Kernels

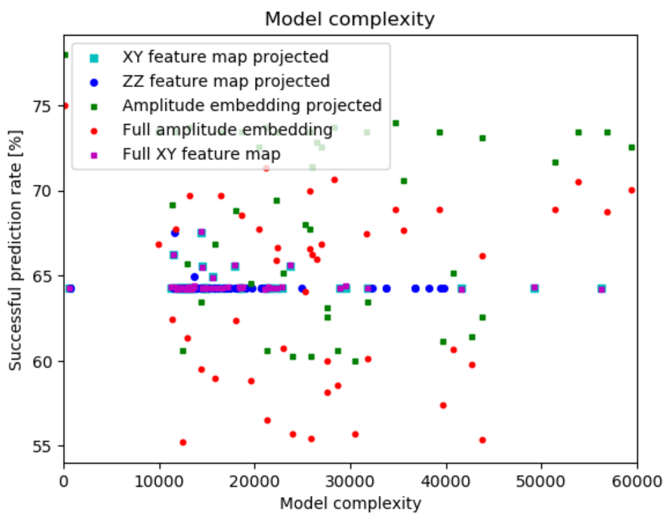


Figure 8.8: Model complexity of each kernels

Chapter 9

Conclusion

In this paper, we focused on the use of quantum kernels to classify classical data. After a general introduction to quantum computing and a description of the current challenges of quantum computing, we focused on the similarity of quantum machine learning to classical kernel methods. We have extended the original idea of quantum kernels to include the possibility of these kernels being P hard in terms of computational complexity. We attempted to apply these results to the real data classification problem. However, in light of the experiments we have carried out, we regret to point out that we have been unsuccessful and in our experiments we have not proved that hard-to-simulate kernels can be useful. We assume that the classification problems stemmed from the inappropriate use of the embeddings we used, and for future efforts to improve the experiment, we recommend extending the work to thoroughly investigate the compatibility of different embeddings depending on the type of data we are trying to encode. On the other hand, we have to mention the fact that the kernels we used took advantage of quantum computing, such as the principle of superposition, quantum entanglement and parallelism, so that on real quantum hardware, evaluating these kernels would result in an exponential speedup.

Despite the failed experiments, we want to mention that quantum machine learning currently represents a great opportunity to push computer science further towards possibilities we cannot imagine. Governments and companies are investing resources in quantum computing, which can only accelerate the progress in computer sciences. Quantum computers are increasing their qubit count almost every year, and it can be assumed that this trend will not stop, but rather the opposite. With more computing power, quantum computing will become more relevant and it will offer countless possibilities. Although the focus is currently on crypto-security, for which quantum computers pose a significant risk, it would be a mistake to give this issue all the attention. I believe that in the future, quantum computers and the algorithms developed on them will be used to enhance not only information sciences but also physics, chemistry, drug discovery, materials science, finances, healthcare and many more, with an incredible potential to improve the quality of our lives in general.

Appendix A

Bibliography

[AB06] Sanjeev Arora and Boaz Barak, *Complexity of counting*, `https://www.cs.princeton.edu/courses/archive/spring06/cos522/count.pdf`, 2006, Accessed: 2025-01-07.

[AC02] First Author and Second Coauthor, *Title of the paper*, Complex Systems **16** (2002), no. 3, 321–336.

[Bro24] Michael Brooks, *Bring on the noise*, MIT Technology Review **127** (2024), no. 1, 50.

[CEMM98] Richard Cleve, Artur Ekert, Chiara Macchiavello, and Michele Mosca, *Quantum algorithms revisited*, Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences **454** (1998), no. 1969, 339–354.

[Cop02] Don Coppersmith, *An approximate fourier transform useful in quantum factoring*.

[CSTEK01] Nello Cristianini, John Shawe-Taylor, André Elisseeff, and Jaz S. Kandola, *On kernel-target alignment*, Neural Information Processing Systems, 2001.

[HBM+21] Hsin-Yuan Huang, Michael Broughton, Masoud Mohseni, Ryan Babbush, Sergio Boixo, Hartmut Neven, and Jarrod R. McClean, *Power of data in quantum machine learning*, Nature Communications **12** (2021), no. 1, 21–23.

[HCT+19] Vojtěch Havlíček, Antonio D. Córcoles, Kristan Temme, Aram W. Harrow, Abhinav Kandala, Jerry M. Chow, and Jay M. Gambetta, *Supervised learning with quantum-enhanced feature spaces*, Nature **567** (2019), no. 7747, 209–212.

[HH00] Lisa Hales and Sean Hallgren, *An improved quantum fourier transform algorithm and applications*, Proceedings 41st Annual Symposium on Foundations of Computer Science (2000), 515–525.

[HHL09] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd, *Quantum algorithm for linear systems of equations*, Physical Review Letters **103** (2009), no. 15, 150502.

[HvdH21] David Harvey and Joris van der Hoeven, *Integer multiplication in time o (n log n)*, Annals of Mathematics **193** (2021), no. 2.

[KBS21] Jonas Kübler, Simon Buchholz, and Bernhard Schölkopf, *The inductive bias of quantum kernels*, Advances in Neural Information Processing Systems (M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, eds.), vol. 34, Curran Associates, Inc., 2021, pp. 12661–12673.

[LL15] Yong Liu and Shizhong Liao, *Eigenvalues ratio for kernel selection of kernel methods*, Proceedings of the AAAI Conference on Artificial Intelligence **29** (2015), no. 1.

[MBS$^+$18] Jarrod R. McClean, Sergio Boixo, Vadim N. Smelyanskiy, Ryan Babbush, and Hartmut Neven, *Barren plateaus in quantum neural network training landscapes*, Nature Communications **9** (2018), no. 1.

[Mez07] Francesco Mezzadri, *How to generate random matrices from the classical compact groups*, 2007.

[Mov20] Ramis Movassagh, *Quantum supremacy and random circuits*, 2020.

[NC10] Michael A. Nielsen and Isaac L. Chuang, *Quantum computation and quantum information*, 10th anniversary edition ed., Cambridge University Press, 2010.

[PVG$^+$11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., *sklearn.datasets.make_moons*, Scikit-learn 1.5 documentation, 2011.

[Rep16] UCI Machine Learning Repository, *Pima indians diabetes database*, Kaggle, 2016.

[Rev23] MIT Technology Review, *Ibm wants to build a 100,000 qubit quantum computer*, `https://www.technologyreview.com/2023/05/25/1073606/ibm-wants-to-build-a-100000-qubit-quantum-computer/`, May 2023, Accessed: 2023-12-31.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Communications of the ACM **21** (1978), no. 2, 120–126.

[Sch21] Maria Schuld, *Supervised quantum machine learning models are kernel methods*, 2021.

[Sho94] Peter W. Shor, *Algorithms for quantum computation: Discrete logarithms and factoring*, Proceedings 35th Annual Symposium on Foundations of Computer Science, IEEE, 1994, pp. 124–134.

[TD04] Barbara M. Terhal and David P. DiVincenzo, *Adaptive quantum computation, constant depth quantum circuits and arthur-merlin games*, 2004.

[Wil11] Colin P. Williams, *Quantum gates*, pp. 51–122, Springer London, London, 2011.

Appendix B

Table of used symbols

$\mathcal{D}(\mathcal{X}, \mathcal{Y})$	Dataset with samples and its labels
$\mathcal{X} \subset \mathbb{R}^d$	Set of data
$\mathcal{Y} \subset \{-1, 1\}$	Labels of data
$\S_{\rangle}$	i-th sample of dataset $\mathcal{X}$
$\dagger_{\rangle}$	Label for a sample $\S_{\rangle}$
$\mathcal{L}$	Loss function
$\mathcal{A}$	Circuit's architecture
$\mathcal{H}_{\mathcal{A}}$	Haar random distribution on architecture $\mathcal{A}$
H	Set of considered strategies for classification
$h, (h^*)$	Single (optimal) strategy of H
w, b	Weight and bias
ς	Slack variable in soft-margin SVM
ϕ	Feature map
$C, E(x),$	Unitaries
n	Number of samples in dataset
q	Number of qubits in quantum circuit
Q	Set of qubits in circuit
$\mathbf{L}(y_i, \hat{y}_i)$	Hinge loss
d	Dimension of sample space $\mathbb{R}^d$
ρ	Density matrix
g_{ab}	Assymetric geometric difference of two kernels K_a, K_b
$KTA(K_a, K_b)$	Kernel target alignment of kernels K_a, K_b
$k(x_i, x_j)$	Kernel value for points x_i, x_j
S	Composite number in Shor's algorithm
a	Initial guess in Shor's algorithm
r	Period of function $f(x) = a^x \text{mod} N$ in Shor's algorithm
N	Dimensionality of problems in HHL algorithm
t	Set of measured qubits in quantum projected kernels

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Svoboda Jan**	Personal ID number:	**474750**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Control Engineering**		
Study program:	**Cybernetics and Robotics**		

II. Master’s thesis details

Master’s thesis title in English:

Quantum machine learning

Master’s thesis title in Czech:

Kvantové strojové učení

Guidelines:

There is much recent interest in the possibility of the use of quantum computers in machine learning. While early papers such as Havlíček et al. (Nature 2019) were broadly positive, more recent papers (e.g., Kübler et al., NeurIPS 2021) present the challenges involved rather clearly. Notably, Kübler et al. introduced a number of conditions that need to be satisfied by the quantum kernel in order to improve the statistical performance compared to kernels computable classically in polynomial time.
In the present dissertation, the student will develop a method for rejection sampling of random unitaries to satisfy the properties of Kübler et al. (NeurIPS 2021). Notably, the process will start with an ensemble of random unitaries that are hard to simulate classically in polynomial time (Movassagh, 2023). Then, the unitaries that do not satisfy the other properties of Kübler et al. (NeurIPS 2021) will be rejected. The properties of such random quantum kernels will be studied. In the computational testing, a variety of encoding of the inputs should be considered, as well as the ML Reproducibility checklist.

Bibliography / sources:

[1] Vojtěch Havlíček, Antonio D. Córcoles, Kristan Temme, Aram W. Harrow, Abhinav Kandala, Jerry M. Chow & Jay M. Gambetta: Supervised learning with quantum-enhanced feature spaces. Nature volume 567, pages209–212 (2019).
https://www.nature.com/articles/s41586-019-0980-2
[2] Jonas M. Kübler, Simon Buchholz, Bernhard Schölkopf: The Inductive Bias of Quantum Kernels. NeurIPS 2021, https://proceedings.neurips.cc/paper/2021/file/69adc1e107f7f7d035d7baf04342e1ca-Paper.pdf
[3] Ramis Movassagh: The hardness of random quantum circuits. Nature Physics 19 (11), 1719-1724
[4] Vojtěch Havlíček et al., https://github.com/qiskit-community/qiskit-machine-learning/blob/main/qiskit_machine_learning/kernels/trainable_fidelity_quantum_kernel.py (2023)
[5] Vyacheslav Kungurtsev, Georgios Korpas, Jakub Marecek, Elton Yechao Zhu: Iteration Complexity of Variational Quantum Algorithms. Quantum 2024. https://arxiv.org/pdf/2209.10615.pdf

Name and workplace of master’s thesis supervisor:

Mgr. Jakub Mareček, Ph.D. Artificial Intelligence Center FEE

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **01.03.2024** Deadline for master's thesis submission: **07.01.2025**

Assignment valid until:
by the end of summer semester 2024/2025

Mgr. Jakub Mareček, Ph.D.	doc. Ing. Zdeněk Hurák, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature