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.
Supervisor–specialist: Ing. Aleš Wodecki Ph.D.
Field of study: Cybernetics and robotics
May 2024

# Acknowledgements

I would like to thank everyone who stood by my side throughout this amazing part of my life, the last chapter of which is this thesis.

# Declaration

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

In Prague, 24. May 2024

# Abstract

Although quantum computing is still not yet used in the real world, it is important to study it in order to understand the concepts of quantum computing and to expand our knowledge of the possibilities of quantum computing. 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 they may be challenging to simulate using classical computer systems. The aforementioned methodologies are subsequently tested on real data sets, with somewhat disappointing results.

**Keywords:** quantum computing, quantum kernels, kernel target alignment, kernel methods

**Supervisor:** Mgr. Jakub Mareček Ph.D.
Ústav X,
Uliční 5,
Praha 99

# Abstrakt

**Klíčová slova:** slovo, klíč

**Překlad názvu:** Moje bakalářka se strašně, ale hrozně dlouhým předlouhým názvem — Cesta do tajů kdovíčeho

# Contents

# Figures

# Tables

# Chapter 1

# Introduction

In recent years, quantum computing has gained more and more scientific focus, lending itself to being the revolutionizer across various artificial intelligence fields, including machine learning. While machine learning is obviously one of the 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 enable us to perform computations exponentially faster than classical computers. This may accelerate not only the training but also classification. However quantum computation does not only carry benefits, several disadvantages need to be pointed out, such as nowadays hardware limitation. Even though rapid advancements are made, up to data quantum computers still faces problems of low computational power which rises from coherence time, gate fidelity and qubit connectivity - these enforce restriction on what problems can and cannot be solved by quantum computers today, making quantum computers noisy and tending to errors. Correcting errors and filtering out noise can be done, but it utilizes another qubit, making the computation inefficient. Another problem with quantum computing is the phenomena that quantum mechanics is based on: measurement is part of an experiment. With qubits living in their own Hilbert space, mapping results of quantum computation back to the classical data so that scientists can utilize it affects the state significantly, and the development of trustworthy methods for state readout is still a simmering problem. Despite these obstacles, inspecting the potential power of quantum computing is a necessary task to harness the quantum computational power once large-scale quantum computers are designed. With the rise of quantum computing, several questions occurred. 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 can't be done on classical computers. Several qualitative measures are assumed, such as time of computation, resources needed for the task, etc. In this work, we introduce 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 examples of a rather useless tool that has the property of being hard to simulate.

# Chapter 2

# Quantum kernels

Machine learning is one of the most growing and focused areas of artificial intelligence studies. This field involves a vast number of methods. It focuses on finding statistical models given data and developing strategies for computational tasks such as data classification, regression and other tasks. In this paper focus is on general soft-margin support vector machine, described as follows:

**Definition 2.1** (Soft-margin SVM)**.** Given data $\mathcal{D}(\mathcal{X}, \mathcal{Y})$, with $\mathcal{X}$ being samples and $\mathcal{Y}$ its labels obtain weight vector $w^*$, bias $b^*$ and slack variable $\varsigma$, so that we minimize:

$$\begin{aligned} &\underset{w,b,\varsigma}{min} ||w||_2^2 + C \sum_{i=1}^{n=|X|} \varsigma_i && (2.1)\\ &\text{subject to } y_i(wx_i - b) \geq 1 - \varsigma_i, \qquad \varsigma_i \geq 0, \forall_i \in \{1, ..., n\}, \end{aligned}$$

Where $C$ is hyperparameter.

This is often called a primal problem. Soft-margin SVM leads to classifying data $\mathcal{X}$ with labels $\hat{\mathcal{Y}}$ (hat over letter corresponds to prediction):

$$\hat{y}_i = \text{sign}(w^* x_i + b^*). \tag{2.2}$$

We further introduce hinge loss:

$$\mathbf{L}(\mathbf{y_i}, \hat{\mathbf{y}}_\mathbf{i}) = \max(0, 1 - y_i\hat{y}_i) = \max(0, 1 - y_i(wx_i + b)), \tag{2.3}$$

outputting 0, then $\hat{y}_i$ is the same as label $y_i$, and a positive number describing the distance from correct labelling otherwise. The optimal solution to soft-margin SVM satisfies

$$\varsigma_i = L(y_i, \hat{y}_i). \tag{2.4}$$

Rewriting original equation **??** then yields:

$$\underset{w,b}{min} \lambda ||w||_2^2 + \sum_{i=1}^{n=|X|} \mathbf{L}(y_i, \hat{y_i}) \tag{2.5}$$

Such an approach can work correctly with the assumption of linearly separable data, which in real-life applications is more than naive. The key is to map data $\mathcal{X}$ to higher-dimensional space, making our data separable with a hyperplane. Data mapping is done using a feature map. This is where the quantum circuit comes in handy. A feature map can generally be seen as a map from Euclidean space to higher dimensional Hilbert space. When speaking about almost anything in the context of quantum computing, it is always thought of as a concept of living in Hilbert space and embedding data to a quantum circuit with a feature map is exactly what needs to be done once we want to map data $\mathcal{X}$ to higher dimensional Hilbert space $\mathcal{H}$.

**Definition 2.2** (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 $\mathbb{C}^{2^n \times 2^n}$. Then, the data encoding feature map is defined as:

$$\phi : \mathcal{X} \to \mathcal{F} \tag{2.6}$$

$$\phi(x) = E(x) |0^n\rangle \tag{2.7}$$

where $E(x)$ is unitary gate defined based on data $x$.

This feature map gives rise to a kernel-distance metric based on the dot product.

**Definition 2.3** (Quantum kernel)**.** Let $\phi$ be defined as above. Then quantum kernel is the inner product between two data encoding feature vectors $phi(x)$, $\phi(x')$ $\{x, x'\} \in \mathcal{X}$:

$$\kappa(x, x') = \langle \phi(x) | \phi(x') \rangle, \tag{2.8}$$

we can also define kernel matrix $K$ with rows and columns corresponding to the inner product between individual samples:

$$K(\mathcal{X})_{ij} = \kappa(x_i, x_j). \tag{2.9}$$

.

This kernel matrix $K(\mathcal{X})$ commonly referred to as Gram matrix is positive definite To some extent, all of quantum computing can be considered an instance of a kernel method since classical kernel methods map to higher dimensions but access higher dimensions only using a kernel. Quantum computers do something similar with accessing quantum Hilbert space using measurements. Thus, all quantum computation using the encoding of classical data appears equivalent to kernel methods. With mapping to higher dimensional Hilbert space, a dual problem can be introduced. Once the features are mapped, the equivalent dual problem to soft-margin SVM is:

$$\begin{aligned} \max_{\alpha_i} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j}^{|\mathcal{X}|} \alpha_i \alpha_j y_i y_j \left\langle \phi(x_i) | \phi(x_j) \right\rangle \\ s.t. \sum_i y_i \alpha_i = 0 \\ 0 \leq \alpha_i \leq \lambda, \forall i. \end{aligned} \tag{2.10}$$

The famous kernel trick states that there exists function $f(x_i, x_j)$ in the Euclidean space, which is equivalent to the dot product $\langle \phi(x_i) | \phi(x_j) \rangle$. Popular functions satisfying this property are Gaussian kernel function $f(x_i, x_j) = e^{-\gamma ||x_i - x_j||_2^2}$, radial basis function $f(x_i, x_j) = e^{-\frac{||x_i, x_j||^2}{2\sigma^2}}$, et cetera. From the above, while thinking about generating quantum kernels, one thing is clear - the generated kernel is very closely linked to the feature map. Examples of feature maps will be presented later. Quantum computers have several properties that suggest quantum advantage, with the leading one being exploring exponentially large spaces using entanglement and interference.

## 2.1 Reproducing kernel Hilbert spaces

Data $\mathcal{X}$ together with embedding $\psi(x)$ generate reproducing kernel Hilbert space.

**Definition 2.4** (Reproducing kernel Hilbert space)**.** Let $\mathcal{X} \neq \emptyset$. The RKHS of a kernel $\kappa$ over $\mathcal{X}$ is the Hilbert space F created by completing the span of functions $f : \mathcal{X} \to \mathbb{R}, f(\cdot) = \kappa(x, \cdot), x \in \mathcal{X}$. For two functions $f(\cdot) = \sum_i \alpha_i \|(x^i, \cdot), g(\cdot) = \sum_j \beta_j \|(x^j, \cdot) \in F$, the inner product is defined as:

$$\langle f | g \rangle_F = \sum_{ij} \alpha_i \beta_j \|(x^i, x^j), \tag{2.11}$$

with $\alpha_i, \beta_j \in \mathcal{R}$.

## 2.2 Spectral decomposition

To use the notion of later kernel target alignment, spectral decomposition needs to be introduced. Let us have a kernel $k(x, y)$, marginal distribution

$\mu(x)$, and integral operator $K$ defined as:

$$Kf(x) = \int k(x, x')f(x')\mu(dx'), \tag{2.12}$$

where $f(x)$ is target function. Due to Mercer's theorem, we can say that if $k$ is a symmetric positive-definite kernel, then there exists a set of eigenfunctions $\psi$ and eigenvalues $\gamma$ such as

$$k(x, x') = \sum_i \gamma_i \psi_i(x)\psi_i(x'), \tag{2.13}$$

and this set may be infinite. This allows us to define spectral decomposition:

$$f(x) = \sum_i \alpha_i \psi_i(x), \tag{2.14}$$

again, there is no upper bound on $i$, and this decomposition may be infinite.

### 2.2.1 Kernel classifier

In order to evaluate our methods, we will use a kernel classifier. Unlabelled data points $x \in \mathcal{X}$ will be labelled with $\hat{y}$ via equation:

$$\hat{y} = f(X) = \sum_{i=1}^{N} \alpha_i k(\hat{x}, x_i), \tag{2.15}$$

where $x_i$ are samples from training data $\mathcal{D}(X, y)$. $\alpha_i$ is sort of a weight which given by $y_i(K(X, U)_{ii} - \lambda\mathbb{I})$ with $\lambda$ being an hyperparameter.

# Chapter 3

# Quantum kernel methods properties

The main idea behind using quantum kernel methods is the fact that quantum computers can do computations that are hard to simulate classically. The very important thing to mention is the data. Loosely put, once a quantum process generates the data, quantum computers are thought to overcome classical ones. On the other hand, once the data are generated classically, the hope for an advantage is much smaller. For example, once we have a quantum circuit which can easily compute function $f$, which requires exponential resources to be approximated classically, once we generate data as

$$\mathcal{Y} = f(x) + \epsilon,$$

quantum kernel $\kappa(x, x') = f(x)f(x')$ then has an exponential advantage for learning $f$ over classical kernel[KBS21]. To bring something to the table, we shall dig much deeper than this trivial example does. We wish to investigate the properties of kernels and separate "good" kernels from "bad" ones. To efficiently separate them, we use two metrics of kernel quality:

- Kernel target alignment
- Asymmetric geometric difference

The first is used in work [KBS21], the latter from [HBM$^+$21]. Kernel target alignment (KTA) may be regarded as the final measure that offers qualitative insight into the calculated kernel and will be utilized to show that the kernel is competitive and even advantageous to its classical opponent. The asymmetric geometric difference have a close relationship with each other and truly describe whether the quantum kernels have an edge over the classical ones.

## 3.1 Kernel target alignment

Kernel target alignment is introduced in [CSTEK01]. It is probably the most used measure when describing the quality of kernels.

**Definition 3.1** (Kernel target alignment)**.** Kernel target alignment $A(k, f)$ is then defined as follows:

$$A(k,f) = \frac{\langle k, f \otimes f \rangle}{\langle k,k\rangle^{1/2} \langle f \otimes f, f \otimes f\rangle^{1/2}} = \frac{\sum_i \gamma_i \alpha_i^2}{(\sum_i \gamma_i)^{1/2} \sum_i \alpha_i^2}, \tag{3.1}$$

where $k$ is the quantum kernel and $f$ is the target function.

However, a rather more understandable definition would be the following: let's have a binary classification task. We can then define an ideal kernel matrix, which is generated from the training dataset labels $y \in \{\pm 1\} \in \mathcal{Y}$:

$$K^*_{ij} = y_i y_j. \tag{3.2}$$

Kernel target alignment is then defined as:

$$A(K, K^*) = \frac{\langle K|K^* \rangle_F}{\sqrt{\langle K^*|K^*\rangle_F \langle K|K\rangle_F}}, \tag{3.3}$$

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 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 $KTA(A, B) \in [-1, 1]$.

## 3.2 Asymmetric geometric difference

Asymmetric geometric difference is defined as follows:

$$g_{ab} = g(K_a||K_b) = \sqrt{||\sqrt{K_b} K_a^{-1} \sqrt{K_b}||_\infty}, \tag{3.4}$$

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 $g_{cq}$ (asymmetric geometric difference between classical and quantum kernel) is proportional to $q_{cq} \propto \sqrt{N}$. Once this is met, the second condition is $s_c \propto N$ while $s_q \ll N$.

## 3.3 Speed-up using quantum kernels

The essential idea of utilizing quantum kernels is that it can produce evaluation speed-up. Evaluating kernel on the classical computer has complexity $n^2$. On the other hand, the situation is different with quantum computers. The time complexity of mapping to feature space is $n$. The inner product is then calculated using a swap test with 1. Thus, there is a quadratic speed-up in evaluating kernels compared to classical computers. [KMS19]

# Chapter 4

# Hardness of random circuits

To accomplish the quantum advantage, the quantum algorithm needs to have one property - it must be hardly simulated classically. This event would mean overturning the Extended Church Turing thesis, an event commonly named Quantum supremacy. Hand in hand with quantum supremacy comes the fact that not even classical supercomputers can outperform quantum computers. While there are several strategies for such a job, random circuit sampling is most promising since it implements universal quantum computation, relies on a minimal number of assumptions and can be tested in larger experiments. Two other ways of representing quantum advantage are Bosson sampling and instantaneous models of quantum computation. In this section, I introduce findings from [Mov20]. Let us introduce several concepts that are the foundation of the theorems that are introduced later.

## 4.1 Circuit architecture

Circuit architecture is mentioned throughout this section and is important to define. First, we assume that each circuit has $n$ qubits. On these qubits can be applied 1-qubit and 2-qubit gates. Regarding the fact that any quantum computation can be translated to a circuit using solely universal gates [Wil11], we can see this model as universal. By circuit architecture $\mathcal{A}$, we mean such ensemble that we apply $m$ gates on these qubits. At this point, we define gates only as blackbox, 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 **??**, some sort of a blueprint. We can say that architecture $\mathcal{A}$

with a description of each gate defines a quantum circuit. The architecture

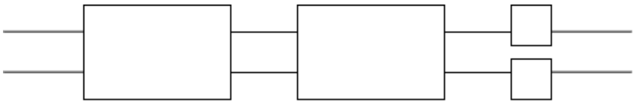


**Figure 4.1:** Circut architecture $\mathcal{A}$ blueprint

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

$$C = C_m C_{m-1} \dots C_1, \tag{4.1}$$

Where $C_i$ is a single gate. If the 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, yielding that it does nothing to the second qubit.

## 4.2 Worst case and average case circuits

In the literature, it is proven [TD04], that for architecture with depth, $n = 4$ exists worst case circuit for which to classically approximate quantity $p = |\langle 0^n|C|0^n\rangle|^2$ is #P-hard up to the constant relative error. However, talking about worst-case circuits is insufficient to achieve quantum supremacy. Quantum supremacy claims that drawing strings from a distribution that mimics the probability distribution induced by the random circuit is computationally hard for any classical algorithm that takes as input 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.

## 4.3 Haar random circuit distribution

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. An algorithm for sampling the Haar measure is given in the chapter devoted to experiments.

### 4.3.1 Cayley transformation and Cayley path

Cayley transformation is a map

$$f(x) : \mathbb{R} \to \mathbb{C} : \frac{1+ix}{1-ix} \tag{4.2}$$

which maps a line of points to a unit circle. We also want to further define $f(-\infty) = -1$. This function is used when transforming between two unitary matrices trough path $\theta \in [0, 1]$. Following the definition of Cayley transformation, we further introduce:

$$H = \tau(h) = \sum_{\alpha=1}^{N} f(\lambda_\alpha) \langle \psi_\alpha | \psi_\alpha \rangle , \tag{4.3}$$

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

$$C(\theta) = W\tau(\theta h), \tag{4.4}$$

where $W$ is the fixed worst case circuit and $h$ is 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 worst-case circuit $W$ and average-case circuit with $W\tau(0) = W\mathbb{I} = W$ and $W\tau(1) = WH$, the later is due to the left-translation invariance again random Haar unitary, thus an instance of an average case circuit. This is really an important concept worth stressing. What Cayley transformation does is that it makes the worst case circuit and random case circuit defined on the architecture $\mathcal{A}$ equivalent. The equation for circuit unitary becomes:

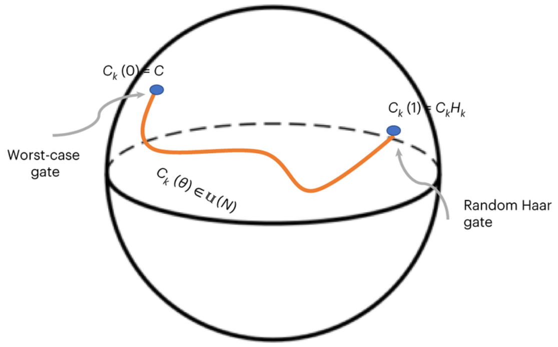


**Figure 4.2:** Cayley path

$$C(\theta) = W_m\tau_m(\theta)W_{m-1}\tau_m(\theta)\dots W_1\tau_1(\theta). \tag{4.5}$$

We then 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.

### 4.3.2 Theorems

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

**Theorem 4.1** (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 to within small multiplicative error. Then it is #P hard to calculate the probability amplitude for most circuits with the same architecture to within $\epsilon = 2^{-\Omega(m)}$ where $m$ is the number of gates.*

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

**Theorem 4.2** (#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 $H \in \mathcal{H}_{\mathcal{A}}$.*

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

$$Pr[|\mathcal{O}(C(z_i)) - p_0(z_i) \leq \epsilon] = 1 - \frac{1}{poly(n)}; |z_i| \leq \Delta, \tag{4.6}$$

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

**Theorem 4.3** (Robustness of #P hardness [Mov20]). *Assuming access to an oracle $O$ as described above, it is #P-hard to compute $p_0(C(\theta))$ over $\mathcal{H}_{\mathcal{A}}$ to within $\epsilon = 2^{-\Omega(m^2)}$ additive error.*

This is an important result since it builds the concept of robustness to the random circuit sampling.

# Chapter 5

# Main concept

From the previous section we can show main concept of this work. Main idea is to construct the kernel $\kappa_R(x_i, x_j)$ such that the kernel is utilizing random unitary. The kernel is constructed as:

$$\kappa_R(x_i, x_j) = \langle\psi(x_i)|U|\psi(x_j)\rangle\,, \tag{5.1}$$

with $U$ being randomly generated unitary. If we sample from unitary distribution for unitaries $U$ for a long enough time, we will find a unitary that strictly improves measures of kernel quality, such as kernel target alignment, etc. For proof, we use previously mentioned theorems 4.2 and 4.3. Since these proofs take into account estimating probabilities of $|\langle 0^n|C|0^n\rangle|^2$, we need to extend it to our case, where we wish to estimate 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

$$\begin{aligned} |\langle\psi(x)|C|\psi(y)\rangle|^2 &= |\langle 0^n|E^\dagger(x)CE(y)|0^n\rangle|^2 \qquad (5.2)\\ &= |\langle 0^n|C|0^n\rangle|^2 \qquad (5.3)\end{aligned}$$

. This means that only by applying unitary operations (which are known from the data, and their conjugate transpose can be easily calculated) we can translate our problem to the one of estimating $|\langle 0^n|C|0^n\rangle|^2$. If estimating $|\langle\psi(x)|C|\psi(y)\rangle|^2$ wouldn't be #P-hard, estimating $|\langle 0^n|C|0^n\rangle|^2$ would not be as well.

# Chapter 6

# Embeddings

Since embedding data to the quantum circuit together with random unitary gives rise to a kernel, we may introduce several ways to map classical data $\mathcal{D}(\mathcal{X}, \mathcal{Y})$ to the circuit. There exist several approaches to map such data, and there is still an open question about which mapping works well on the given data. In our experiment, these mappings will be tested against each other.

## 6.1 Basis embedding

One of the simplest embeddings is basis embedding [Sch21]. In this example, the data are binarized, and these binary strings are then mapped to the circuit. In our setup, this means that if the data is mapped to the binary string of length $N$, at least $2N$ qubits need to be present in the circuit. Basis embedding is by default meant such that every 0-bit gets mapped to the $|0\rangle$ and 1 to $|1\rangle$, but the states may be arbitrary, for example $|+\rangle$ and $|-\rangle$, $|l\rangle$ and $|r\rangle$, but as long as the mapped states are orthogonal, we can talk about basis embedding. The mapping is described by the following equation:

$$\psi(x) = \bigotimes_{i=1}^{N} m(x_i) \tag{6.1}$$

where $x_i$ is the $i$-th bit of the string, and $x$ is the mapping function as described above. Basis embedding generates kernel corresponding to Kronecker delta:

$$\kappa(x, x') = \delta_{x,x'} \tag{6.2}$$

## 6.2 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:

$$|\psi(x)\rangle = \frac{1}{|x|} \sum_{i=1}^{N} x_i \left|_i\right\rangle , \tag{6.3}$$

where $_i$ is from the 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 the linear kernel:

$$\kappa(x, x') = |x^\dagger x||^2 \tag{6.4}$$

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

$$R = \mathbb{I} - 2|v\rangle\langle v| \tag{6.5}$$

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

## 6.3 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:

$$|\psi(x)\rangle = \bigotimes_{i=1}^{N} \cos(x_i)\,|0\rangle + \sin(x_i)\,|1\rangle\,. \tag{6.6}$$

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

$$\kappa(x, x') = \prod_{i=1}^{n=|X|} |\cos(x_i - x_i')|^2 \tag{6.7}$$

## 6.4 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:

$$U_{\Phi(x)} = exp(i \sum_{S \in \mathcal{I}} \psi_S(x) \prod_{i \in S} P_i), \tag{6.8}$$

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

$$\psi(x) = \begin{cases} x_i & \text{if } S = \{i\} \\ \prod_{j \in S}(\pi - x_j) & \text{if } |S| > 1 \end{cases} \tag{6.9}$$

Some examples include using $P = Y$, or $P = Z$, in one qubit setting. These map interval $[-\pi, \pi]$ to the corresponding states visualised on bloch sphere: Corresponding colors of statevectors correspond to the gradient between green ($x = -\pi$) and blue ($x = \pi$) color.

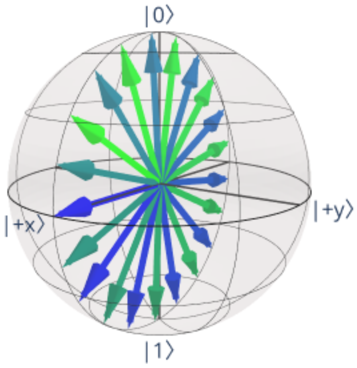


**Figure 6.1:** Pauli Y feature map

## 6.5 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 feature map family, and maps the data with the set of single qubit rotation

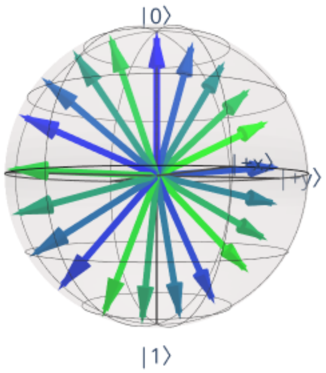


**Figure 6.2:** Pauli Z feature map

$P(\lambda)$ about the $Z$ axis depending on data and a set of $CNOT$ gates. $P(\lambda)$ is defined as:

$$P(\lambda) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\lambda}. \end{pmatrix} \tag{6.10}$$

Circuit realising ZZ-feature map is then depicted in the figure below:

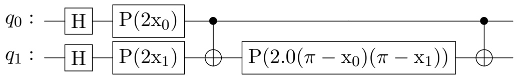


**Figure 6.3:** ZZ-feature map

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

## 6.6 Repeated embedding

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

$$\kappa(x, x') = (|x^\dagger x\|^2)^3 \tag{6.11}$$

# Chapter 7

# Experiments

A set of experiments were done in order to justify previously introduced concepts. At first, data are mapped to the circuit using an embedding. The kernel target alingment for kernel $\langle\psi(x)|\mathbb{I}|\psi(x')\rangle$ is calculated. This acts as a benchmark for the next circuits. New kernels are designed using random unitary sampling as:

$$\kappa(x, x') = \langle\psi(x)|U|\psi(x')\rangle \tag{7.1}$$

with $U$ being a randomly sampled unitary matrix. The idea then is to use rejection sampling to find the best possible choice for $U$ to improve the kernel. Then, using the previously introduced results, this kernel is #P-hard to simulate classically and thus can lead to a quantum advantage.

## 7.1 Random unitaries

For generating random unitaries QR decomposition approach is utilized [Mez07]. The procedure is as follows:

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

## 7.2 Algorithm

The algorithm for each experiment is designed so that kernel target alingment is evaluated for each random unitary $C$. This experiment is then run for 4 types of embedding:

- ZZ feature map
- XY feature map
- Amplitude embedding
- Rotation embedding

For each of these embeddings, 100 random unitaries are sampled to produce a kernel $\kappa(x, x') = \langle 0^n | C | 0^n \rangle$ which is #P hard to simulate classically. Kernel target alignment of these kernels given the task are then calculated and evaluated.

**Algorithm 1** Classifying data

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

                         Output:
        Unitary maximizing kernel-target alignment $U$
                 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 statevectors 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
        $t \leftarrow t + 1$
    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
        $\hat{y}_i \leftarrow \hat{y}_i$
    end for
end procedure
```

## 7.3 Ad hoc data experiment

For first experiment *ad hoc dataset* from qiskit machine learning library. Settings of data generation were : Example of generated data was: This experiment was done repeatedly while giving interesting results.

| | |
|---|---|
| Training set size for each class | 30 |
| Test set size for each class | 10 |
| Sample dimension | 2 |
| Gap | 0.5 |

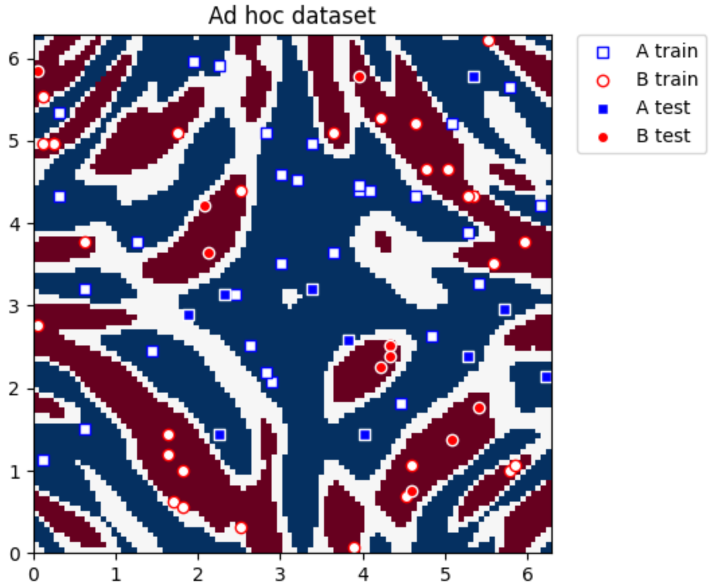
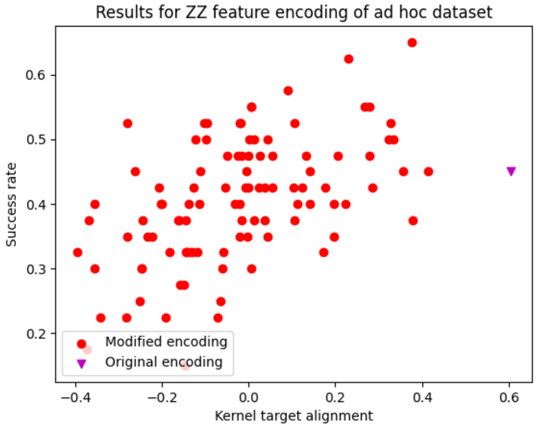


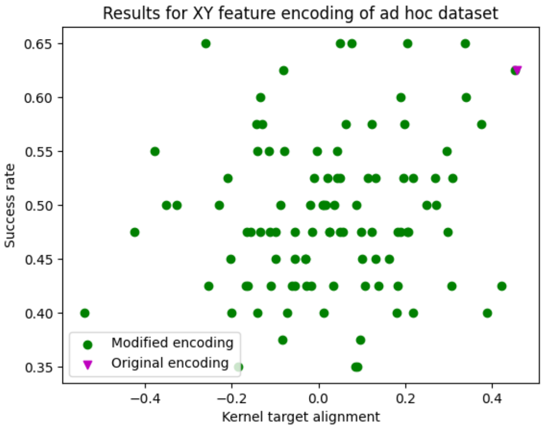
**Figure 7.1:** Example of data from ad hoc dataset

### 7.3.1 ZZ feature mapping

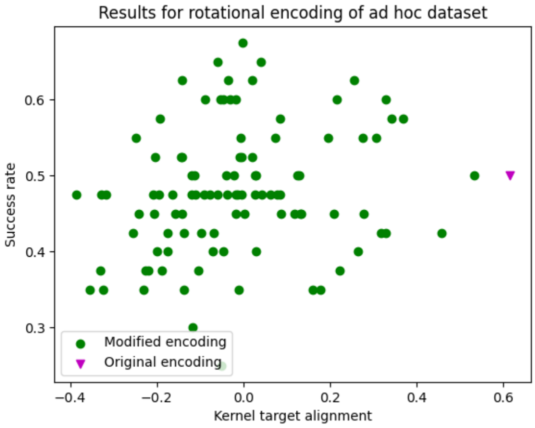


### 7.3.2 XY feature mapping

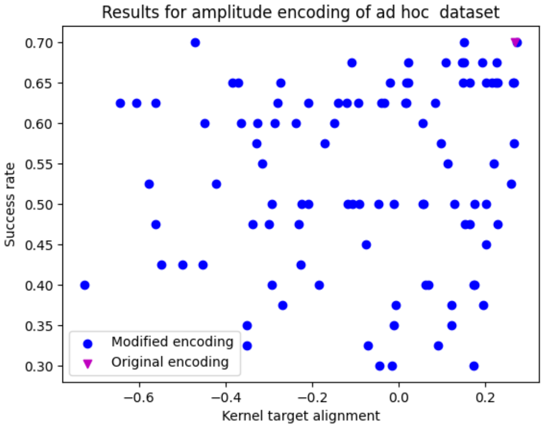
XY feature embedding achieved the greatest rating of tested kernels.



### 7.3.3 Rotational embedding



### 7.3.4 Amplitude embedding

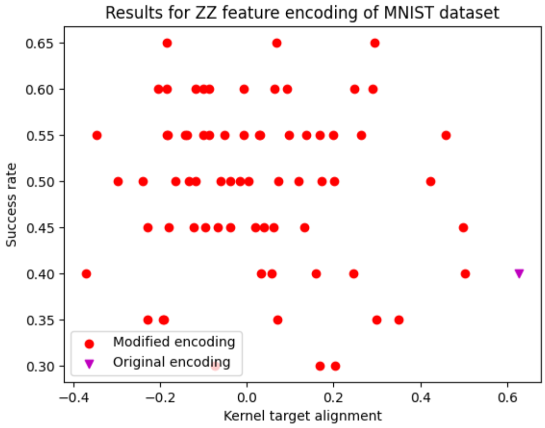


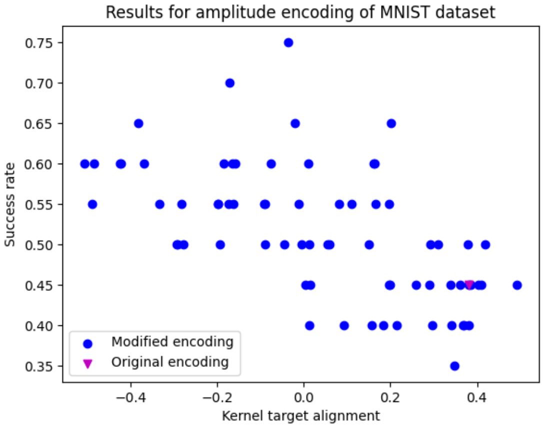
## 7.4 MNIST data experiment

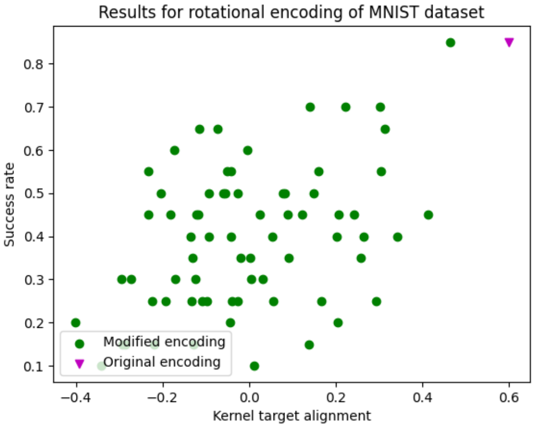
Another experiment was done using the notoriously known MNIST dataset. The modified National Institute of Standards and Technology database is a dataset of 60000 training and 10000 testing samples of handwritten letters. Each letter consists of (10,10) pixels. For binary classification, two letters $R$ and $M$ where selected. In order to encode this data into the quantum circuit, data needed to be preprocessed - for each letter were obtained two features:

- Difference between the sum of upper half pixels and lower half pixels
- Difference between the sum of left half pixels and right half pixels

These features were then downscaled to the interval of $[-\pi, \pi]$ so the rotational gates realising embedding can be used. The same algorithm was then used to find unitaries that increase kernel target alignment. The results are given below:




Results for amplitude encoding of MNIST dataset
Success rate
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
−0.4
−0.2
0.0
0.2
0.4
Kernel target alignment
● Modified encoding
▼ Original encoding


Results for rotational encoding of MNIST dataset
Success rate
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
−0.4
−0.2
0.0
0.2
0.4
0.6
Kernel target alignment
● Modified encoding
▼ Original encoding

# Chapter 8

# Discussion

From the experiments, several questions arise. The first of them is the absence of correlation between the kernel target alignment and model performance. Only amplitude embedding on ad hoc datasets exhibits the properties that were introduced - that kernel target alignment improves the model's performance. It is no surprise that for each dataset, each embedding has a different performance, but from the results, it can be seen that the best possible kernel target alignment for almost each of the embeddings is the initial one, with unitary being just the identity matrix.

# Chapter 9

# Conclusion

This work introduces the idea of creating kernels using random unitaries, making them #P hard to evaluate classically. Kernel target alignment is mentioned as the main method for defining the quality of the generated kernel. Concepts introduced in the first part of the thesis are then tested on data with rather unsatisfactory results, shattering the main idea of this work. Kernel target alignment was not improved in any of the experiments; what is more, even the performance of the generated models did not improve, compared to the basic kernel not being enhanced by the random unitary.

# Appendix A

## Bibliography

[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.

[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.

[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.

[KMS19] Jonas M Kübler, Krikamol Muandet, and Bernhard Schölkopf, *Quantum mean embedding of probability distributions*, Physical Review Research **1** (2019), no. 3, 033159.

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

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

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

[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.

# 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: **24.05.2024**

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

| | | |
|---|---|---|
| Mgr. Jakub Mareček, Ph.D.<br>Supervisor’s signature | prof. Ing. Michael Šebek, DrSc.<br>Head of department’s signature | prof. Mgr. Petr Páta, Ph.D.<br>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