Kernels and Quantum Machine Learning

Bikram Khanal$^{[0000-0003-2292-520X]}$ and Pablo Rivas$^{[0000-0002-8690-0987]}$

School of Engineering and Computer Science
Department of Computer Science
Baylor University, Texas, USA
{bikram_khanal1,pablo_rivas}@baylor.edu

Abstract. Support Vector Machine (SVM) and Kernel Method (KM) are used widely for classification and regression in learning from data. Kernels are positive definite functions that map the data into higher (possibly infinite) dimensions. Generally, SVMs[1] implement kernel methods as a subroutine that maps the non-linear data to a higher dimensional where it becomes linearly separable. SVM draws a linear decision boundary between classes of data points in this feature space. This paper reviews kernels and kernel methods from a classical machine learning perspective and their possible implementation in quantum machine learning. We start with the basis of kernels, including Hilbert space and Reproducing Kernel Hilbert Space, Mercer's condition, and prove three widely used kernels validity satisfying Mercer's condition. We review two different approaches of quantum machine learning, parameterized quantum circuit and kernel-based training, and discuss the potential advantage of one over another. This paper can facilitate the readers' getting started with kernel theory and quantum machine learning.

Keywords: kernel methods · support vector machines · reproducing kernel Hilbert spaces · machine learning · quantum machine learning.

1 Introduction

In learning, SVMs and KMs are often used, but not limited, in numerical optimization, improvised generalization, working set selection, and parameter tuning and have shown promising result in science and engineering [34]. Given a dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i), \mathbf{x}_i \in \mathbb{R}^n, y \in \mathbb{R}, i = 1, \dots, N\}$[2], where n is the number of features in $\mathcal{D}$, often referred as dimension of D and N is the total number of data instances, SVM learning method computes an approximation function as

$$f(\mathbf{x}) = b + \sum_{j=1}^{m} y_j \alpha_j k(\mathbf{x}_j, \mathbf{x}) \tag{1}$$

[1] Except linear SVMs. Linear SVMs are linearly separable in input space and do not require feature mapping to a higher dimension.

[2] In many literature we find $y \in \{-1, +1\}$. We consider this setting for binary classification and try to generalize here for multi-class classification and Regression

with $y \approx sign(f(\mathbf{x}))$ for classification, and $y \approx f(\mathbf{x})$ for regression. For learning purpose, a Support Vector (SV) set $\{\mathbf{x}_j, j = 1, ..., m\} \subset \{\mathbf{x}_i, i = 1, \ldots, N\}$ are determined. A kernel function k is chosen based on the learning problem, and parameters α_j and b are computed. As a refresher, a kernel function maps the data from input space to higher dimensional feature space (sometimes infinite dimensions), in which data are linearly separable and hence, easier to analyse.

In recent years, kernel-based learning has been used widely in quantum-enhanced machine learning [30]. Huang et al. showed that for a specific dataset, the quantum kernel could learn with lower generalization bound error over optimal classical kernel method [20]. Effective spectral transformation technique can maintain quantum kernels superiority for a large dataset in noisy intermediate-scale quantum (NISQ) era [41]. Blank et al. exhibited to bypass the specific state preparation requirements and use the kernelized binary classifier to perform swap-test on data-encoding quantum states [6]. Recently, it was shown that supervised quantum machine learning fundamentally relates to quantum kernel methods [35]. Using quantum-embedding kernels, we can construct a learning problem that proves the separation between quantum machine learning and classical machine learning [25].
We consider these kernel behaviors as our motivation to explore kernel theory. This paper reviews the kernels, kernels tricks, and elementary introductory linear algebra. We briefly discuss two approaches for quantum machine learning and one's advantages and drawbacks over others. In the next section, we define, discuss and briefly describe kernel, some kernel functions, and their properties.

2 Kernel and Kernel Functions

Definition 1. ***(Kernel** [35]) Given two vectors $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X} \subseteq \mathcal{D}$, and $\phi : \mathcal{X} \rightarrow \mathcal{F}$ is a feature map from input space $\mathcal{X}$ to Hilbert space $\mathcal{F}$ such that $\boldsymbol{x} \rightarrow \boldsymbol{x} := \phi(\boldsymbol{x})$, k is a class of kernel that corresponds to the dot product in $\mathcal{F}$ is given by*

$$k(\mathbf{x}_1, \mathbf{x}_2) = \langle \phi(\mathbf{x}_1), \phi(\mathbf{x}_2) \rangle_{\mathcal{F}}. \tag{2}$$

Thus, for any two vectors $\mathbf{x}_1, \mathbf{x}_2$ that are mapped to a feature space $\mathcal{F}$ by some mapping function ϕ, their dot product $\langle \cdot, \cdot \rangle$ give rises to a kernel function k. Fig. (1) illustrates the mapping.

From (2) we see that we can compute the kernel function k by computing the inner product between two vectors and vice-versa. Since, inner product is symmetric by definition, the kernel function is also symmetric. i.e.,

$$k(\mathbf{x}_1, \mathbf{x}_2) = k(\mathbf{x}_2, \mathbf{x}_1), \tag{3}$$

that satisfies the Cauchy-Schwarz Inequality, given by (taken from [38]):

$$||k(\mathbf{x}_1, \mathbf{x}_2)||^2 \leq k(\mathbf{x}_1, \mathbf{x}_1) \cdot k(\mathbf{x}_2, \mathbf{x}_2). \tag{4}$$

From (3) one can ask, are all symmetric functions kernel? Or in particular, what makes a function f a kernel function? Mercer proved that any positive definite

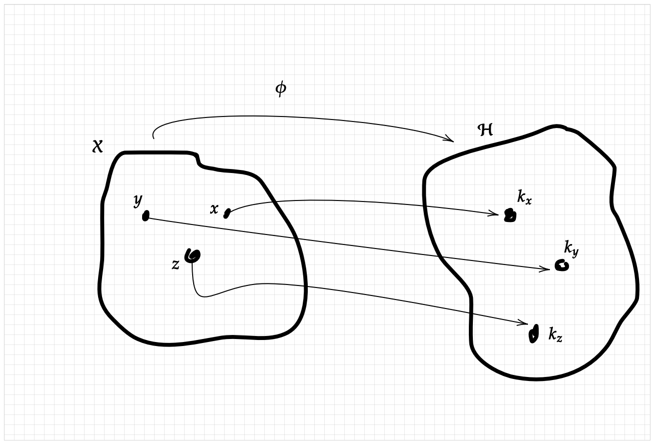

Fig. 1: Mapping of a input from input Space $\mathcal{X}$ to a feature space often called Hilbert Space $\mathcal{H}$ via a mapping function $\phi(\cdot)$. $k(\cdot)$ is a kernel function defined as $k : \mathcal{X} \rightarrow \mathbb{R}$.

symmetric function $k(\mathbf{x}_1, \mathbf{x}_2)$ is a kernel function [12,27]. Let, $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_l$ be any set of vectors, $l \in \mathbb{N}$ and $\alpha_1, \alpha_2, ..., \alpha_l$ be some real values. Then, from Mercer, k must satisfy

$$\sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0. \tag{5}$$

Previously, we digressed ourselves and introduced Mercer's Theorem properties informally to support our claim that any symmetric positive definite function is a kernel. However, we did not explicitly mention Mercer's Theorem's properties to facilitate the verification. Before we explain Mercer's theorem, let us define **Gram Matrix**, which will be handy later.

Definition 2. *(**Gram Matrix** [13]) $K \in \mathbb{R}^{n \times n}$ is a Gram matrix whose (i, j)-th element is*

$$K_{ij} := k(\mathbf{x}_i, \mathbf{x}_j), \forall i, j \in 1, \ldots, n. \tag{6}$$

We can prove that (6) satisfies Mercer's condition (we will shortly define Mercer's Theorem and Mercer's condition) and therefore is a valid kernel.

Proof. Let $r_i, r_j \in \mathbb{C}$ and $\{x_1, ..., x_M\} \subseteq \mathcal{X}$ with M ≥ 2, we can express that

$$\sum_{i,j=1}^{M} r_i r_2 * k(\mathbf{x}_i, \mathbf{x}_j) = \langle \sum_i r_i \phi(\mathbf{x}_i), \sum_j r_j \phi(\mathbf{x}_j) \rangle = || \sum_i r_i \phi(\mathbf{x}_i)||^2 \geq 0.$$

An inner product matrix must be a Gram Matrix for any valid kernel from Mercer's Theorem.

Mercer's Theorem
(*Simplified from [32,28,13]*) Let $\mathcal{X}$ be a compact subspace of $\mathbb{R}^n$. Suppose $k : [a, b] \times [a, b] \rightarrow \mathbb{R}$ is a continuous definite symmetric function satisfying (5). k is a positive definite kernel on $\mathcal{X}$. Let $T_k : L_2(\mathcal{X}) \rightarrow L_2(\mathcal{X})$ be an integral operator. If, k is a kernel of T_k then $\forall f \in L_2(\mathcal{X})$ it gives rise to T_k via

$$T_k f(x) = \int_{\mathcal{X}} k(x, y) f(y) dy, \tag{7}$$

then, we can expand $k(x, y)$ in a uniformly convergence series for set of orthonormal bases $\{\psi_i(\cdot)\}_{i=1}^{\infty}$ of L_2 that consist eigenfunctions of T_2 such that the corresponding eigenvalues sequences $\{\lambda_i\}_{i=1}^{\infty}$ are non-negative given by

$$k(x, y) = \sum_{j=1}^{\infty} \lambda_j \psi_j(x) \psi_j(y). \tag{8}$$

Thus, for any kernel function, $K(x, y) = \langle \phi(x), \phi(y) \rangle$, that computes the dot product between input x and y, transformed by a function,if it satisfies the Mercer's condition, i.e.,

$$\int \int K(x, y) g(x) g(y) dx dy \geq 0, \tag{9}$$

which is equivalent to (5), then it is guaranteed that there is an underlying mapping function ϕ. Mercer's condition is often referred as Positive-Definite Symmetric (PDS) condition.

In the next section we define and briefly describe some kernel functions and their properties.

2.1 Kernel Functions

Below we describe three kernel functions and their properties. Each of the kernel functions induces some feature space. However, since no explicit mapping to this feature space occurs, we can find the optimal linear separators for non-linear data in feature spaces with millions of dimensions [33]. Table 1 summarizes the selected kernel.

We describe the kernels mentioned in Table 1 in the following subsections. Without loss of generality, we omit the cost C for simplicity.

Table 1: Three Common kernels [39,21,19].

Kernel Function	Formula	Optimization Parameter
Linear	$k(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle$	-
Polynomial	$k(\mathbf{x}, \mathbf{x}') = (\langle \mathbf{x}, \mathbf{x}' \rangle + r)^d$	$r \in \mathbb{R}$ and $d \in \mathbb{N}$
RBF	$k(\mathbf{x}, \mathbf{x}') = exp(\frac{-\|\|\mathbf{x}, \mathbf{x}'\|\|^2}{2\sigma^2}) + C$	C and $\gamma = \frac{1}{2\sigma^2}$

Explanation: RBF is Radial Basis Function, C: Cost, r: coefficient, d: Polynomial Degree.

2.1.1 Linear Kernels A linear kernel is the basic type of kernel often considered the simplest kernel. It is defined as the dot product between the points that we are trying to classify:

$$k(\mathbf{x}, \mathbf{x}') := \langle \mathbf{x}, \mathbf{x}' \rangle. \tag{10}$$

Let us prove that linear kernel satisfies Mercer's Condition, by satisfying (5), and hence is a valid kernel.

Proof. Let us assume that we have m vectors, $\{\mathbf{x}_1, \dots, \mathbf{x}_m\}$. Then $\forall c \in \mathbb{R}^N$,

$$\begin{aligned}
\sum_{i,j} c_i c_j k(\mathbf{x}, \mathbf{x}') &= \sum_{i=1}^{m}\sum_{j=1}^{m} c_i c_j \sum_{a=1}^{N} \mathbf{x}_{i_a} \mathbf{x}'_{j_a} \\
&= \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{a=1}^{N} c_i \mathbf{x}_{i_a} c_j \mathbf{x}'_{j_a} \\
&= \sum_{a=1}^{N} (\sum_{i=1}^{m} c_i \mathbf{x}_{i_a})(\sum_{j=1}^{m} c_j \mathbf{x}_{j_a}) \\
&= \sum_{a=1}^{N} (\sum_{i=1}^{m} c_i \mathbf{x}_{i_a})^2 \geq 0.
\end{aligned}$$

Thus, from (4) and (5) we say that (10) is a positive definite kernel, satisfying the Mercer's condition. Note that we do not transfer the data to any other space in the linear kernel, but we use the inner products of points to obtain the feature space.

Below we briefly mention some properties of linear kernel informally.

1. The input vectors are not projected into higher dimensions and remain in the original feature space.
2. It can be estimated by calculating the inner product between input vectors.
3. It is often used in a dataset with many features.

The linear kernel performs the computation in the original space and does not induce higher dimensional feature space.

2.1.2 Polynomial Kernel Given $\mathbf{x}, \mathbf{x}'$ be the vectors in the input space, we define a polynomial kernel as

$$k(\mathbf{x}, \mathbf{x}') := \left(\langle \mathbf{x}, \mathbf{x}' \rangle + C\right)^d, \tag{11}$$

where d is a polynomial degree, C is a free training parameter, often referred as cost. Polynomial kernel with degree two is quite popular in Natural Language Processing (NLP). We can prove that (11) satisfies Mercer's condition and therefore is a valid kernel. Assume $C \geq 0$ and d a positive integer.

Take $C = 0$, for an arbitrary set of n vectors, $\mathbf{x}_n, \ldots, \mathbf{x}_n$, it immediately follows that $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ is positive-definite. If we take d Hadamard product of $k(\mathbf{x}_i, \mathbf{x}_j)$, this will yield Gram matrix, which is a positive definite by definition.

Take $C > 0$, $(\langle \mathbf{x}^T, \mathbf{x}' \rangle + C)$ is also positive definite because the sum of two positive-definite is also positive definite. If we take d Hadamard product of $(\langle \mathbf{x}^T, \mathbf{x}' \rangle + C)$, this will also yield Gram matrix. Thus, (11) satisfies a Mercer's condition and is a kernel function.

Below we briefly mention some properties of Polynomial kernel informally.

1. It accounts the original features of input samples and the combination of these features to estimate the similarity between two data-points.
2. It give rises to a polynomial decision boundary in input space of degree d.
3. The decision boundary in feature space is still a hyperplane.

It induces the kernel space of d dimension and maps the input data to this feature space.

2.1.3 Radial Basis Function kernel Given two vectors $\mathbf{x}, \mathbf{x}'$ in an input space, RBF kernel is defined as

$$K(\mathbf{x}, \mathbf{x}') = e^{-\frac{1}{2\sigma^2}||\mathbf{x}-\mathbf{x}'||^2}, \tag{12}$$

where σ is a variance.

Proof. To prove that (12) is a kernel function, let us define a feature map:

$$\phi(\mathbf{x}) = e^{\frac{-1}{2\sigma^2}||\mathbf{x}||_2^2}, \tag{13}$$

and a kernel function,

$$k_1(\mathbf{x}, \mathbf{x}') = e^{\frac{1}{2\sigma^2}\langle \mathbf{x}, \mathbf{x}' \rangle}, \tag{14}$$

where k_1 is a valid kernel because it is the exponential of positive scalar times linear kernel.

Let us define a valid kernel function, satisfying Definition 1:

$$k_2(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle = e^{\frac{-1}{2\sigma^2}[||\mathbf{x}||_2^2 + ||\mathbf{x}'||_2^2]}, \tag{15}$$

We can rewrite equation (12) as

$$\begin{aligned} K(\mathbf{x}, \mathbf{x}') =& e^{\frac{-1}{2\sigma^2}(\mathbf{x}-\mathbf{x}')^T(\mathbf{x}-\mathbf{x})} \\ =& e^{\frac{-1}{2\sigma^2}||\mathbf{x}||_2^2 + ||\mathbf{x}'||_2^2 - 2\langle \mathbf{x}, \mathbf{x}' \rangle} \\ =& e^{\frac{-1}{2\sigma^2}||\mathbf{x}||_2^2 + ||\mathbf{x}'||_2^2} e^{\frac{1}{\sigma^2}\langle \mathbf{x}, \mathbf{x}' \rangle} \\ =& k_2(\mathbf{x}, \mathbf{x}') k_1(\mathbf{x}, \mathbf{x}'). \end{aligned} \tag{16}$$

Eq. (16) proves that the RBF kernel could be defined as a product of two valid kernels. The product of two positive definite kernels is a positive definite kernel. Thus, (16) satisfies Mercer's condition and hence is a valid kernel. Further, from the Taylor series, we know that e^x gives a polynomial equation of infinite power; thus RBF kernel works as a projection into infinite dimension. It is one of the most preferred and widely used kernels in SVM [10].

Below we briefly mention some properties of RBF kernel informally.

1. It places RBF centered at each point then performs linear manipulation that maps the data to a higher dimension, (possibly infinite) space where the data-points are easier to separate [43].
2. It is a stationary kernel that is, for some vector-valued c of dimension to match inputs, a stationary kernel will yield same $k(\mathbf{x}, \mathbf{y})$ for $k(\mathbf{x}+c, \mathbf{y}+c)$ [16].
3. It can be applied without any prior knowledge about data.

Using the Taylor series, we can expand e^x to a polynomial equation of infinite power. Thus, the RBF kernel can induce a feature space of infinite dimension in theory.

2.2 Reproducing Kernel Hilbert Space (RKHS)

Before we begin defining the RKHS and its properties, we define some related terminology.

Definition 3. *(**Hilbert Space** [44]) Hilbert space is a complete inner product space with respect to norm defined as*

$$||x|| = \sqrt{\langle x, x \rangle}. \tag{17}$$

Definition 4. *(**Reproducing Kernel** [2]) For a Hilbert Space $\mathcal{H}$ $k(\cdot, \cdot)$ is a reproducing kernel if $\forall f \in \mathcal{H}, f(x) = \langle k(x, \cdot), f(\cdot) \rangle$.*

Definition 5. *(**Reproducing Kernel Hilbert Space** [4]) It is a Hilbert Space $\mathcal{H}$ with a reproducing kernel k whose span is dense in $\mathcal{H}$.*

In simpler terms, RKHS is a vector Space, $\mathcal{H}$ with a scalar product $\langle \cdot, \cdot \rangle$ and a mapping function $\phi : \mathcal{X} \mapsto \mathcal{H}$ such that, $\forall x, y \in X$ and k a kernel function

$$k(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}}. \tag{18}$$

The space we define here is a space of functions. So, all the elements in this space are functions. From previous definition (equation (18)) we define the functions in $\mathcal{H}$ as

$$x \in \mathcal{X} \mapsto \phi(x) \in \mathcal{H} := k_x := k(x, \cdot).$$

Consider x as a parameter of function k_x and a as an argument. So, when we pass a to k_x, it maps to $k(x, a)$. That is, $x \in \mathcal{X}$ is mapped to a function $K_x : \mathcal{X} \to \mathbb{R}, k_x(y) = k(x, y)$. In other words, under this definition we can write $\phi(x) = k(\cdot, x)$ and $\phi(y) = k(\cdot, y)$ without ambiguity. From these examples we can illustrates two important features of RKHS [15] mentioned below.

– The feature map of every point is in the feature space.

$$\forall x \in \mathcal{X}, k(\cdot, x) \in H.$$

– k has a **reproducing property**, i.e., $\forall x \in \mathcal{X}, \forall f \in \mathcal{H}, \langle f, k(., x)\rangle_{\mathcal{H}} = f(x)$ In particular, for any $x, y \in \mathcal{X}$

$$k(x, y) = \langle k(., x), k(., y)\rangle_{\mathcal{H}}.$$

Moore-Aronszajn Theorem [1] states that for every positive definite function $k(\cdot, \cdot)$ there exists a unique RKHS and vice-versa.

2.3 Quantum Kernels

Before we turn ourselves towards quantum kernels, lets recall some subtle descriptions of RKHS. A RKHS is an alternative feature space $\mathcal{F}$ for a kernel κ, that is a feature space of functions $x \to f_x(\cdot) = \kappa(x, \cdot)$ [36,35]. In simple terms, $\mathcal{F}$ is a Hilbert Space with a reproducing kernel $k : \mathcal{X} \times X \to \mathcal{R}$ that satisfies two properties.

– For $x \in \mathcal{X}$, $\kappa(x, \cdot) \in \mathcal{F}$.
– $\langle f, \kappa(x, \cdot)\rangle = f(x), \forall f \in \mathcal{F}$.

We provide an example to check if a kernel is a reproducing kernel with reproducing properties.
Consider a kernel definition $K(x, x_1) = \sum_{i=1}^{P} \phi_i(x)\phi_i(x_1)$. Lets fix a point x and remove x_1, we get

$$K(x, .) = \sum_{i=1}^{P} \phi_i(x)\phi_i(.) = \sum_{i=1}^{P} \phi_i(x)\phi_i \in \mathcal{F}.$$

It is a linear combination of functions, so it does belong to $\mathcal{F}$. This satisfies the first property.

Next, let us take a function in $\mathcal{F}$ defined as $f = \sum_j w^j \phi_j$. Then compute

$$\langle f, k(x, .)\rangle_{\mathcal{F}} = \sum_{j=1}^{p} w_j \phi_j(x) = f(x).$$

This means that k is a reproducing kernel. A reproducing kernel is often used as input encoding in quantum machine learning. Let $x \in \mathcal{X}$ be a set of input data. Let $|\phi(x)\rangle$ be an encoded quantum state that lives in Hilbert space $\mathcal{F}$. In other words, this data encoding is similar to the feature map $\phi : \mathcal{X} \to \mathcal{F}$, often referred to as quantum feature map. Thus, by the definition of kernel function we get a kernel κ from ϕ via (2). From the second property of reproducing kernel, κ is a reproducing kernel of an RKHS R_k where $|\phi(.)\rangle$ lives. Hence, we conclude that any kernel function that satisfies the reproducing properties is a quantum kernel without getting into the mathematics behind the claim.

Definition 6. ***Quantum Kernel*** *[36] If $\mathcal{X}$ is an input domain and ϕ data encoding feature map, a quantum kernel is an inner product between two data-encoding feature vectors, $\rho(x), \rho(y)$ with $x, y \in \mathcal{X}$,*

$$\kappa(x, y) = tr\{\rho(y)\rho(x)\} = ||\,\langle\phi(y)|\phi(x)\rangle\,||^2. \tag{19}$$

Unlike classical kernel, quantum kernel is a product of complex-valued kernel $\kappa_c(x, y) = \langle\phi(y)|\phi(x)\rangle$ and its complex conjugate, $\kappa_c(x, y)^* = \langle\phi(y)|\phi(x)\rangle^* = \langle\phi(x)|\phi(y)\rangle$. We will prove that (19) is a valid kernel, i.e. symmetric and positive definite. $\kappa_c(x, y)$ is a valid kernel from (2). We have to show that $\kappa_c(x, y)^*$ is also a valid kernel.

Proof. (Simplified from [35] Def. 2) For any $c_m \in \mathbb{C}$ and for any $x_m \in \mathcal{X}, m = 1, \ldots, M$

$$\sum_{m,m'} c_m c^*_{m'} (\kappa_c(x_m, x_{m'}))^* = \sum_{m,m'} c_m c^*_{m'} \langle\phi(x_m)|\phi(x_{m'})\rangle$$

$$= \left(\sum_m c_m \langle\phi(x_m)|\right)\left(\sum_m c^*_m |\phi(x_m)\rangle\right)$$

$$\left|\left|\sum_m c^*_m |\phi(x_m)\rangle\right|\right|^2 \geq 0. \tag{20}$$

We proved that the complex conjugate of a kernel is positive definite function therefore a valid kernel. By the property of kernel, the product of two kernels is also a kernel. Thus, (19) is a valid quantum kernel.

3 Quantum Machine Learning

Classical machine learning algorithms generally require several operations polynomial to dataset size N in training examples. Due to the constant growth in globally stored data, $\mathcal{O}(N^2)$ might be two demanding even after efficiency optimization without altering the statistical performance [9]. Quantum computation could potentially improve the efficiency by running costly machine learning algorithms or their subroutines efficiently [37,9]. In quantum computing, a state $|\psi\rangle$ with n qubits is a vector in 2^n dimensional complex space. A wide variety of QML algorithms build a matrix transformation by performing quantum logic operations/measurements on qubits for which it multiplies the state vector by $2^n \times 2^n$ matrices [5]. Leveraging the matrix transformation, quantum computers have improved the efficiency exponentially over the classical counterparts by solving common linear algebraic operations in the time that is polynomial in n [17,5]. These efficiency improvements have shown some promising results in machine learning algorithms. For example, for a dataset with N dimensional feature space, and M training vectors, SVM can formulate a quadratic problem [40], that can be solved proportional to $\mathcal{O}(\log(\epsilon^{-1}) poly(N, M))$ with ϵ accuracy. However, Rebentrost et al. has shown that for both training and classification, quantum SVM can be implemented with $\mathcal{O}(\log NM)$ run time complexity [31]. Rivas et al. showed that it is possible to determine qubit input sequence for yielding 1 as an output of the circuit [22] in $O(\sqrt{N})$ complexity using Grover Search. There are two approaches for QML widely exploited in literature, Quantum Neural Network (QNN) or parameterized quantum circuits and using quantum kernel as the inner product of two data-encoded quantum states. We briefly describe these two approaches in the following sections.

3.1 Parameterized Quantum Circuits

In parameter Quantum Circuits (PQCs), often called QNN in machine learning, the learning problems are formulated as variational optimization problems. The hybrid system[3] is used to find the approximate solution for these problems. The general approach for learning in a hybrid system follows to pre-process the data in the classical computer. It helps to determine the set of parameters θ for PQC. In the hybrid system, a quantum device prepares the quantum states and performs measurements. The measurement outcomes are post-processed by classical computers. A learning algorithm in a classical computer updates the model parameters. The entire algorithm is executed in a close loop of the hybrid system.

For a input domain $\mathcal{X}$ and output domain $\mathcal{Y}$, a classical machine learning model can be defined as a function $f_\theta : \mathcal{X} \to \mathcal{Y}$ for deterministic models and $p_\theta : \mathcal{X}[\otimes\mathcal{Y}] \to [0,1]$ for probabilistic models [36]. We will see that PQCs can be interpreted as machine learning models with a mere conceptual change. Given a close quantum system with n qubits we apply a quantum circuit $U(\theta, x)$ $x \in \mathcal{X}, \theta \in \mathbb{R}^k$ to the initial state $|0\rangle^{\otimes n}$. The quantum circuit depends on inputs x and parameters θ. One can interpret the expected value of measurements $\mathcal{M}$ as an output of a quantum system in hybrid system. There can be any internal structure for $U(x,\theta)$; however, (21) provides the most popular choice:

$$U(x,\theta) = W(\theta)S(x), \tag{21}$$

where $S(x)$ is the data embedding block and $W(\theta)$ is the parametrised block. Provided with a quantum circuit and input, a domain deterministic quantum model can be defined as follows.

Definition 7. ***Deterministic Quantum model*** *(simplified from [35,3,8]). If $\mathcal{X}$ be a input domain and $U(x,\theta)$ be a unitary operator, and $\mathcal{M} = \sum_i \lambda_i p_i$ be a Hermitian operator of interest with ith eigenvalue λ_i and p_i the ith projector on corresponding eigenspace, $|\psi\rangle(x,\theta)$ is the state prepared by $U|0\rangle^{\otimes n}$, deterministic variational quantum model is defined as*

$$f_\theta(x) = \langle\psi(x,\theta)|\, \mathcal{M}\, |\psi(x,\theta)\rangle\,. \tag{22}$$

Similar to classical machine learning algorithms, PQC models can be trained by minimizing the loss function $L(\theta)$. The parameter vector θ on (21) can be optimized via gradient descent and gradient-free way. For the scope of this paper, we discuss the gradient-based optimization where the parameters are updated in an iterative method towards the direction of the local minimum provided by the update rule:

$$\theta \leftarrow \theta - \eta\nabla_\theta L, \tag{23}$$

[3] A widely accepted term for a system with classical computers and quantum computer where pre/post-processing of data and learning algorithm implementation occurs in classical computer and state preparation and measurement occurs in quantum computer.

where η is the learning rate and $\nabla_\theta L$ is the gradient vector. We can approximate the partial derivatives of (22) that depends on a parameter $\mu \in \theta$ using finite-difference method,

$$\frac{\partial f_\theta}{\partial \mu} \approx \frac{f_\theta - f_{\theta+\triangle\theta}}{||\triangle\theta||}, \tag{24}$$

where $\triangle\theta$ is the Cartesian unit vector where μ was exchanged by $\mu + \triangle\mu$, $\triangle\mu$ is a tiny shift.

With gradient-descent as an optimization technique, one immediately thinks of vanishing gradient and exploding gradient problems. Performing unitary operations on Recurrent Neural Networks (RNN) solves the exploding gradient problem [42]. By the definition of quantum gates, all quantum gates except measurements and reset operations are unitary. Thus, any quantum circuit implementing quantum gates avoids exploding gradient problems. While the issue of vanishing gradients on the random quantum circuit with reasonable depth remains [26] we could alleviate the problem by using the highly structured quantum circuits [14].

The following subsection describes a different approach, kernel-based learning, for supervised quantum machine learning.

3.2 Quantum Machine Learning and Kernel Methods

In QML (we focus on supervised learning), rather quantum computing, the data $x \in \mathcal{X}$ are mapped into a quantum state $\phi : x \to |\phi(x)\rangle$ by a mapping function $\phi : \mathcal{X} \to \mathcal{F}$ that lives in a Hilbert space $\mathcal{F}$ and described as $|\phi(x)\rangle$. This mapping is similar to state preparation circuit $U_\phi(x)$. More importantly, encoding inputs to a higher dimension is surprisingly similar to kernel methods. In kernel methods, we can access the higher dimensional feature space via kernel or inner products of features vectors given by (2). Similar to kernel methods, we can access the Hilbert space $\mathcal{F}$ of quantum state via measurement, which can also be expressed by inner products between quantum states [35]. Unlike PQC, the Quantum Kernel Estimator (QKE) does not use a variational circuit for data processing [3]. So, one can easily avoid the barren plateau problem [18,26] using kernel methods for learning, but the cost of pair-wise distance estimation remains.

The first step in QML is to encode the data that can be implemented by a quantum circuit $U_\phi(x)$. Various data-encoding feature map techniques give rise to a kernel. It is important to note that there are kernels that the classical computer can not compute efficiently [25]. Basis encoding, amplitude encoding, repeated amplitude encoding, rotation encoding, and coherent state encoding are some data-encoding strategies that give rise to quantum kernels [36]. We can obtain these kernels by calculating the inner product between states, that is, provided by overlap defined by (19). Once we have the kernel, we can implement it as a subroutine in (1). Note that in quantum machine learning, although the kernel is computed in a quantum computer in (1), the hyperplane is constructed in a classical computer. We can train the quantum-kernel featured model via least square approximation [31,30] or solve standard regularized empirical risk minimization over the RKHS of the quantum kernel [36].

Quantum kernel learning methods finds best coefficient $\alpha_m, m = 1, ..., M$ to optimize the observable measurements in a subspace spanned by trainable parameters M given by

$$\alpha_{opt} = \max_{\alpha} \sum_m \alpha_m - \frac{1}{2} \sum_{m,m'} \alpha_m \alpha_{m'} y_m y_{m'} \kappa(x_m, x_{m'}), x_m \in \mathcal{X}, y_m \in \mathcal{Y}. \quad (25)$$

In the previous two sections, we describe learning in quantum machine learning via two different approaches. Recently, there have been numerous works in these areas. An excellent review with experiments in this field can be found in [11,23,35,24]. In the next section, we review the advantages and disadvantages of these two approaches over.

3.3 Is Kernel-based learning better than Variational Training?

This optimization problem is convex in kernel-based learning, and the subspace contains globally optimal measurements that provide some advantage for kernel models over variational training models. Using available tools, it is guaranteed to find the optimal minimum [7]. On the other hand parameterized ansatz, or “templates”, defines the optimizing subspace for variational training [3]. It is uncertain if this subspace overlaps with the training data sub-space. Thus, the optimization might not be convex, and hence we do not always have access to the optimal minimum. Therefore, kernel-based learning guarantees better or at least the same minima to variational training by finding the global optimal measurements for all possible quantum models. One thing to note, although in principle quantum computers can train the kernel methods learning in $\mathcal{O}(N)$ complexity in the number of training samples N, the pair-wise distance calculation is at least $\mathcal{O}(N^2)$ complexity in classical machine learning. So, does training variational quantum circuit have an advantage over kernel-based training for the larger dataset, or in general, overall? Well, it depends. Parameter-shift rules [29] for gradient-based learning in variational circuits scales linearly in a number of parameters $|\theta|$ and circuit measurement M for training the model. So, the growth function is $\mathcal{O}(|\theta|M)$. From this growth function, one can immediately follow that if the number of parameters grows slow enough with the increase in data size, then the variational-based training does have an advantage over kernel-based learning. However, if similar to a neural network, the number of parameters in ansatz grows linearly with the data size, the variational quantum model will have the same scaling as kernel-based learning regarding M. Regardless, suppose one selects variational training over kernel-based training. In that case, there is no guarantee to obtain an optimal final measurement, and we will be adding extra work to select a better variational ansatz. Thus, kernel-based learning can be a better choice for quantum machine learning in theory.

4 Conclusion

In this review, we surveyed the applications of kernel and kernel methods in SVM (briefly) and quantum machine learning. We covered various topics, including Mercer's theorem, RKHS, three kernel functions, quantum kernels, and quantum machine learning. While doing so, we showed that any symmetric function that satisfies Mercer's condition is a valid kernel. We further proved that the kernel functions satisfying the reproducing property of RKHS could be considered quantum kernels. We explored two approaches for quantum machine learning, parameterized quantum circuit or variational quantum model training and kernel-based training. We believe with access to fault-tolerant quantum computers, kernel-based quantum machine learning enables computations that are exponentially hard classically.

Acknowledgements

The authors would like to thank Dr. Erich Baker and Dr. Greg Speegle for constructive criticism of the manuscript. The work presented here was supported in part by the `BAYLOR.AI` lab.

References

1. Aronszajn, N.: Theory of reproducing kernels. Transactions of the American mathematical society **68**(3), 337–404 (1950)
2. Bartlett, P., Gu, S.: Reproducing kernel hilbert space (2008)
3. Benedetti, M., Lloyd, E., Sack, S., Fiorentini, M.: Parameterized quantum circuits as machine learning models. Quantum Science and Technology **4**(4), 043001 (2019)
4. Berlinet, A., Thomas-Agnan, C.: Reproducing kernel Hilbert spaces in probability and statistics. Springer Science & Business Media (2011)
5. Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., Lloyd, S.: Quantum machine learning. Nature **549**(7671), 195–202 (2017)
6. Blank, C., Park, D.K., Rhee, J.K.K., Petruccione, F.: Quantum classifier with tailored quantum kernel. npj Quantum Information **6**(1), 1–7 (2020)
7. Boyd, S., Boyd, S.P., Vandenberghe, L.: Convex optimization. Cambridge university press (2004)
8. Cerezo, M., Arrasmith, A., Babbush, R., Benjamin, S.C., Endo, S., Fujii, K., McClean, J.R., Mitarai, K., Yuan, X., Cincio, L., et al.: Variational quantum algorithms. Nature Reviews Physics **3**(9), 625–644 (2021)
9. Ciliberto, C., Herbster, M., Ialongo, A.D., Pontil, M., Rocchetto, A., Severini, S., Wossnig, L.: Quantum machine learning: a classical perspective. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences **474**(2209), 20170551 (2018)
10. Deris, A.M., Zain, A.M., Sallehuddin, R.: Overview of support vector machine in modeling machining performances. Procedia Engineering **24**, 308–312 (2011)
11. Farhi, E., Neven, H.: Classification with quantum neural networks on near term processors. arXiv preprint arXiv:1802.06002 (2018)
12. Genton, M.G.: Classes of kernels for machine learning: a statistics perspective. Journal of machine learning research **2**(Dec), 299–312 (2001)
13. Ghojogh, B., Ghodsi, A., Karray, F., Crowley, M.: Reproducing kernel hilbert space, mercer's theorem, eigenfunctions, nystr\" om method, and use of kernels in machine learning: Tutorial and survey. arXiv preprint arXiv:2106.08443 (2021)

14. Grant, E., Wossnig, L., Ostaszewski, M., Benedetti, M.: An initialization strategy for addressing barren plateaus in parametrized quantum circuits. Quantum **3**, 214 (2019)
15. Gretton, A.: Introduction to rkhs, and some simple kernel algorithms. Adv. Top. Mach. Learn. Lecture Conducted from University College London **16**, 5–3 (2013)
16. Gupta, S.: Why is rbf kernel used in svm? (Aug 1963), `https://stats.stackexchange.com/questions/172554/why-is-rbf-kernel-used-in-svm`
17. Harrow, A.W., Hassidim, A., Lloyd, S.: Quantum algorithm for linear systems of equations. Physical review letters **103**(15), 150502 (2009)
18. Holmes, Z., Sharma, K., Cerezo, M., Coles, P.J.: Connecting ansatz expressibility to gradient magnitudes and barren plateaus. PRX Quantum **3**(1), 010313 (2022)
19. Howley, T., Madden, M.G.: The genetic kernel support vector machine: Description and evaluation. Artificial intelligence review **24**(3), 379–395 (2005)
20. Huang, B., Mohseni, B., Boixo, N., McClean: Power of data in quantum machine learning. Nature Communications **12**(2631), 2041–1723 (2021)
21. Karatzoglou, A., Meyer, D., Hornik, K.: Support vector machines in r. Journal of statistical software **15**, 1–28 (2006)
22. Khanal, B., Rivas, P., Orduz, J.: Quantum machine learning: A case study of grover's algorithm. In: Proceedings of the 19th International Conference on Scientific Computing (CSC 2021), Las Vegas, NV, USA. pp. 15–17 (2021)
23. Killoran, N., Bromley, T.R., Arrazola, J.M., Schuld, M., Quesada, N., Lloyd, S.: Continuous-variable quantum neural networks. Physical Review Research **1**(3), 033063 (2019)
24. Kübler, J., Buchholz, S., Schölkopf, B.: The inductive bias of quantum kernels. Advances in Neural Information Processing Systems **34** (2021)
25. Liu, Y., Arunachalam, S., Temme, K.: A rigorous and robust quantum speed-up in supervised machine learning. Nature Physics **17**(9), 1013–1017 (2021)
26. McClean, J.R., Boixo, S., Smelyanskiy, V.N., Babbush, R., Neven, H.: Barren plateaus in quantum neural network training landscapes. Nature communications **9**(1), 1–6 (2018)
27. Mercer, J.: Functions ofpositive and negativetypeand theircommection with the theory ofintegral equations. Philos. Trinsdictions Rogyal Soc **209**, 4–415 (1909)
28. Minh, H.Q., Niyogi, P., Yao, Y.: Mercer's theorem, feature maps, and smoothing. In: International Conference on Computational Learning Theory. pp. 154–168. Springer (2006)
29. Mitarai, K., Negoro, M., Kitagawa, M., Fujii, K.: Quantum circuit learning. Physical Review A **98**(3), 032309 (2018)
30. Park, D.K., Blank, C., Petruccione, F.: The theory of the quantum kernel-based binary classifier. Physics Letters A **384**(21), 126422 (2020)
31. Rebentrost, P., Mohseni, M., Lloyd, S.: Quantum support vector machine for big data classification. Physical review letters **113**(13), 130503 (2014)
32. Rivas-Perea, P.: Algorithms for training large-scale linear programming support vector regression and classification. Ph.D. thesis, The University of Texas at El Paso (5 2011)
33. Russell, S., Norvig, P.: Artificial intelligence: a modern approach (2002)
34. Sánchez A, V.D.: Advanced support vector machines and kernel methods. Neurocomputing **55**(1-2), 5–20 (2003)
35. Schuld, M.: Supervised quantum machine learning models are kernel methods. arXiv preprint arXiv:2101.11020 (2021)
36. Schuld, M., Petruccione, F.: Machine Learning with Quantum Computers. Springer (2021)

37. Schuld, M., Sinayskiy, I., Petruccione, F.: An introduction to quantum machine learning. Contemporary Physics **56**(2), 172–185 (2015)
38. Smola, A.J., Schölkopf, B.: Learning with kernels, vol. 4. Citeseer (1998)
39. Souza, C.R.: Kernel functions for machine learning applications. Creative commons attribution-noncommercial-share alike **3**(29), 1–1 (2010)
40. Tsang, I.W., Kwok, J.T., Cheung, P.M., Cristianini, N.: Core vector machines: Fast svm training on very large data sets. Journal of Machine Learning Research **6**(4) (2005)
41. Wang, X., Du, Y., Luo, Y., Tao, D.: Towards understanding the power of quantum kernels in the nisq era. Quantum **5**, 531 (2021)
42. Wisdom, S., Powers, T., Hershey, J., Le Roux, J., Atlas, L.: Full-capacity unitary recurrent neural networks. Advances in neural information processing systems **29** (2016)
43. Ye, A.: Radial basis functions, rbf kernels, &amp; rbf networks explained simply (Jan 2022), `https://medium.com/dataseries/radial-basis-functions-rbf-kernels-rbf-networks-explained-simply-35b246c4b76c`
44. Young, N.: An introduction to Hilbert space. Cambridge university press (1988)