# Supervised quantum machine learning models are kernel methods

Maria Schuld
*Xanadu, Toronto, ON, M5G 2C8, Canada*

With near-term quantum devices available and the race for fault-tolerant quantum computers in full swing, researchers became interested in the question of what happens if we replace a supervised machine learning model with a quantum circuit. While such "quantum models" are sometimes called "quantum neural networks", it has been repeatedly noted that their mathematical structure is actually much more closely related to kernel methods: they analyse data in high-dimensional Hilbert spaces to which we only have access through inner products revealed by measurements. This technical manuscript summarises and extends the idea of systematically rephrasing supervised quantum models as a kernel method. With this, a lot of near-term and fault-tolerant quantum models can be replaced by a general support vector machine whose kernel computes distances between data-encoding quantum states. Kernel-based training is then guaranteed to find better or equally good quantum models than variational circuit training. Overall, the kernel perspective of quantum machine learning tells us that the way that data is encoded into quantum states is the main ingredient that can potentially set quantum models apart from classical machine learning models.

## I. MOTIVATION

The mathematical frameworks of quantum computing and kernel methods are strikingly similar: both describe how information is processed by mapping it to vectors that live in potentially inaccessibly large spaces, without the need of ever computing an explicit numerical representation of these vectors (Figure 1). This similarity is particularly obvious – and as we will see, useful – in *quantum machine learning*, an emerging research field that investigates how quantum computers can learn from data [1–3]. If the data is "classical" as in standard machine learning problems, quantum machine learning algorithms have to encode it into the physical states of quantum systems. This process is formally equivalent to a *feature map* that assigns data to quantum states (see [4, 5] but also earlier notions in [6–8]). Inner products of such data-encoding quantum states then give rise to a kernel, a kind of similarity measure that forms the core concept of kernel theory.

The natural shape of this analogy sparked more research in the past years, for example on training generative quantum models [9], constructing kernelised machine learning models [10], understanding the separation between the computational complexity of quantum and classical machine learning [5, 11, 12] or revealing links between quantum machine learning and maximum mean embeddings [13] as well as metric learning [14]. But despite the growing amount of literature, a comprehensive review of the link between quantum computation and kernel theory, as well as its theoretical consequences, is still lacking. This technical manuscript aims at filling the gap by summarising, formalising and extending insights scattered across existing literature and "quantum community folklore". The central statement of this line of work is that quantum algorithms optimised with data can *fundamentally* be formulated as a classical kernel method whose kernel is computed by a quantum computer. This statement holds both for the popular class of classically trained variational near-term algorithms (e.g., [15]) as well as for more sophisticated fault-tolerant algorithms *trained* by a quantum computer (e.g., [6]). It will be apparent that once the right "spaces" for the analysis are defined (as first proposed in [5]), the theory falls into place itself. This is in stark contrast to the more popular, but much less natural, attempt to force quantum theory into the shape of neural

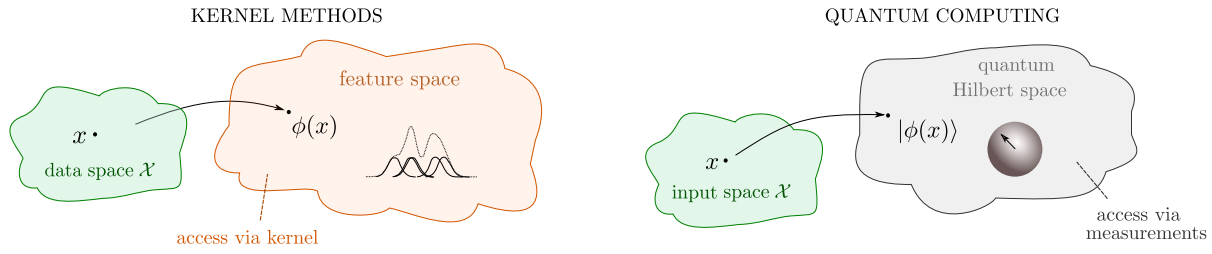


FIG. 1. **Quantum computing and kernel methods are based on a similar principle.** Both have mathematical frameworks in which information is mapped into and then processed in high-dimensional spaces to which we have only limited access. In kernel methods, the access to the feature space is facilitated through *kernels* or inner products of feature vectors. In quantum computing, access to the Hilbert space of quantum states is given by measurements, which can also be expressed by inner products of quantum states.

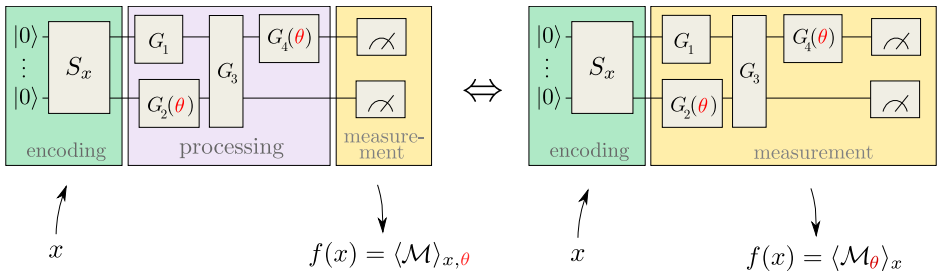


FIG. 2. **Interpreting a quantum circuit as a machine learning model.** After encoding the data with the routine $S_x$, a quantum circuit "processes" the embedded input, followed by a measurement (left). The processing circuit may depend on classically trainable parameters, as investigated in near-term quantum machine learning with *variational circuits*, or it may consist of standard quantum routines such as amplitude amplification or quantum Fourier transforms. The expected outcome of the measurement $\mathcal{M}$ is interpreted as the model's prediction, which is deterministic (generative models, which would consider the measurement *samples* as outputs, are not considered here). Since the processing circuit only changes the basis in which the measurement is taken, it can conceptually be understood as part of the measurement procedure (right). In this sense, quantum models consist of two parts, the data encoding/embedding and the measurement. Training a quantum model is the problem of finding the measurement that minimises a data-dependent cost function. Note that while the measurement could depend on trainable parameters I will not consider trainable embedding circuits here.

networks.[1]

A lot of the results presented here are of theoretical nature, but have important practical implications. Understanding quantum models as kernel methods means that the expressivity, optimisation and generalisation behaviour of quantum models is largely defined by the data-encoding strategy or *quantum embedding* which fixes the kernel. Furthermore, it means that while the kernel itself may explore high-dimensional state spaces of the quantum system, quantum models can be trained and operated in a low-dimensional subspace. In contrast to the popular strategy of variational models (where a quantum algorithm depends on a tractable number of classical parameters that are optimised), we do not have to worry about finding the right variational circuit ansatz, or about how to avoid *barren plateaus* problems [16, 17] – but pay the price of having to compute pairwise distances between data points.

For classical machine learning research, the kernel perspective can help to demystify quantum machine learning. A medium-term benefit may also derive from quantum computing's extensive tools that describe information in high-dimensional spaces, and possibly from interesting new kinds of kernels derived from physics. In the longer term, quantum computers promise access to fast linear algebra processing capabilities which are in principle able to deliver the polynomial speed-up that allows kernel methods to process big data without relying on approximations and heuristics.

The manuscript is aimed at readers coming from *either* a machine learning *or* quantum computing background, but assumes an advanced level of mathematical knowledge of Hilbert spaces and the like (and there will be a lot of Hilbert spaces). Instead of giving lengthy introductions to both fields at the beginning, I will try to explain relevant concepts such as quantum states, measurements, or kernels as they are needed. Since neither kernel methods nor quantum theory are easy to digest, the next section will summarise all the main insights from a high-level point of view to connect the dots right from the start.

A final side note may be useful: quantum computing researchers love to precede any concept with the word "quantum". In a young and explorative discipline like quantum machine learning (there we go!), this leads to very different ideas being labeled as "quantum kernels", "quantum support vector machines", "quantum classifiers" or even "quantum neural networks". To not add to this state of confusion I will – besides standard technical terms – only use the "quantum" prefix if a quantity is explicitly computed by a quantum algorithm (instead of being a mathematical construction in quantum theory). I will therefore speak of "quantum models" and "quantum kernels", but try to avoid constructions like "quantum feature maps" and "quantum reproducing kernel Hilbert space".

## II. SUMMARY OF RESULTS

First, a quick overview of the scope. Quantum algorithms have been proposed for many jobs in supervised machine learning, but the majority of them replace the *model*, such as a classifier or generator, with an algorithm that runs on

[1] In some sense, many near-term approaches to quantum machine learning can be understood as a kernel method with a special kind of kernel, where the model (and possibly even the kernel [14]) are trained like neural networks. This mix of both worlds makes quantum machine learning an interesting mathematical playground beyond the questions of asymptotic speedups that quantum computing researchers tend to ask by default.

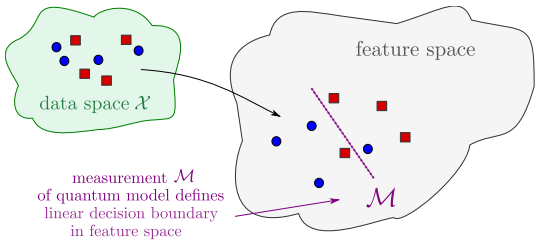


FIG. 3. **Quantum models as linear models in a feature space.** A quantum model can be understood as a model that maps data into a feature space in which the measurement defines a linear decision boundary. This feature space is not identical to the Hilbert space of the quantum system. Instead we can define it as the space of complex matrices enriched with the Hilbert-Schmidt inner product – which is the space where density matrices live in.

a quantum computer. These algorithms – I will call them *quantum models* – usually consist of two parts: the data encoding, which maps data inputs $x$ to quantum states $|\phi(x)\rangle$ (effectively *embedding* them into the space of quantum states), and a measurement $\mathcal{M}$. Statistical properties of the measurement are then interpreted as the output of the model. Training a quantum model means to find the measurement which minimises a cost function that depends on training data. This overall definition is fairly general, and it includes most near-term supervised quantum machine learning algorithms as well as many more complex, fault-tolerant quantum algorithms (see Figure 2). Throughout this manuscript I will interpret the expected measurement – or in practice, the average over measurement outcomes – as a prediction, but the results may carry over to other settings, such as generative quantum models (e.g., [18]). I will also consider the embedding fixed and not trainable as proposed in [14, 19].

The bridge between quantum machine learning and kernel methods is formed by the observation that quantum models map data into a high-dimensional feature space, in which the measurement defines a linear decision boundary as shown in Figure 3. Note that for this to hold we need to define the data-encoding density matrices $\rho(x) = |\phi(x)\rangle\langle\phi(x)|$ as the feature "vectors"[2] instead of the Dirac vectors $|\phi(x)\rangle$ (see Section V A). This was first proposed in Ref. [5]. Density matrices are alternative descriptions of quantum states as Hermitian operators which are handy because they can also express probability distributions over quantum states (in which case they are describing so-called *mixed* instead of *pure* states). We can therefore consider the space of complex matrices enriched with the Hilbert-Schmidt inner product as the feature space of a quantum model and state:

1. *Quantum models are linear models in the "feature vectors" $\rho(x)$.*

As famously known from support vector machines [20], linear models in feature spaces can be efficiently evaluated and trained if we have access to inner products of feature vectors, which is a function $\kappa$ in two data points $x, x'$ called the *kernel*. Kernel theory essentially uses linear algebra and functional analysis to derive statements about the expressivity, trainability and generalisation power of linear models in feature spaces directly from the kernel. For us this means that we can learn a lot about the properties of quantum models if we study inner products $\kappa(x, x') = \mathrm{tr}\,[\rho(x')\rho(x)]$, or, for pure states, $\kappa(x, x') = |\langle\phi(x')\,|\phi(x)\,\rangle|^2$ (see in particular Ref. [12]). I will call these functions "quantum kernels'.

To understand what kernels can tell us about quantum machine learning, we need another important concept from kernel theory: the *reproducing kernel Hilbert space* (RKHS). An RKHS is an alternative feature space of a kernel – and therefore reproduces all "observable" behaviour of the machine learning model. More precisely, it is a feature space of *functions* $x \to g_x(\cdot) = \kappa(x, \cdot)$, which are constructed from the kernel. The RKHS contains one such function for every input $x$, as well as their linear combinations (for example, for the popular Gaussian kernel these linear combinations are sums of Gaussians centered in the individual data points). In an interesting – and by no means trivial – twist, these functions happen to be identical to the linear models in feature space. For quantum machine learning this means that the space of quantum models and the RKHS of the quantum kernel contain exactly the same functions (see Section V B). What we gain is an alternative representation of quantum models, one that only depends on the quantity $\mathrm{tr}\,[\rho(x')\rho(x)]$ (see Figure 4).

This alternative representation can be very useful for all sorts of things. For example, it allows us to study the universality of quantum models as function approximators by investigating the universality of the RKHS, which in turn is a property of the quantum kernel. But probably the most important use is to study optimisation: minimising typical cost functions over the space of quantum models is equivalent to minimising the same cost over the RKHS of

[2] The term *feature vectors* derives from the fact that they are elements of a vector space, not that they are vectors in the sense of the space $\mathbb{C}^N$ or $\mathbb{R}^N$.

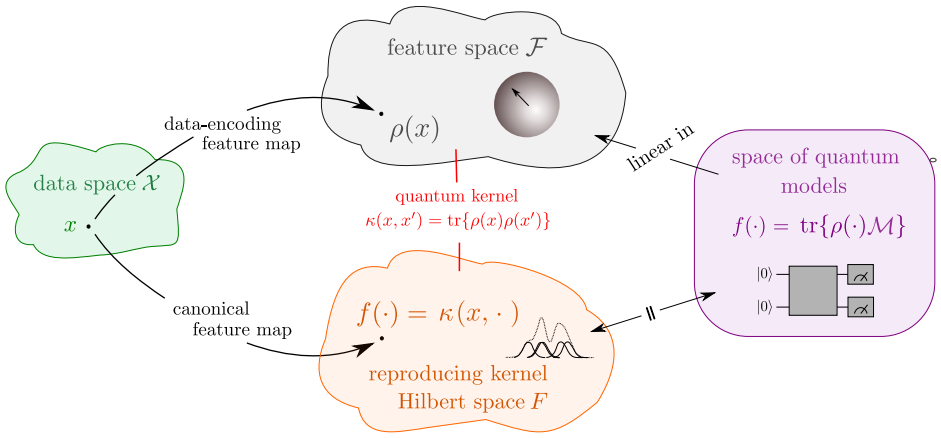


FIG. 4. **Overview of the link between quantum models and kernel methods.** The strategy with which data is encoded into quantum states is a feature map from the space of data to the feature space $\mathcal{F}$ "of density matrices" $\rho$. In this space, quantum models can be expressed as a linear model whose decision boundary is defined by the measurement. According to kernel theory, an alternative feature space with the same kernel is the RKHS $F$, whose vectors are functions arising from fixing one entry of the kernel (i.e., the inner product of data-encoding density matrices). The RKHS is equivalent to the space of quantum models, which are linear models in the data-encoding feature space. These connections can be used to study the properties of quantum models as learners, which turn out to be largely determined by the kernel, and therefore by the data-encoding strategy.

the quantum kernel (see Section VI A). The famous *representer theorem* uses this to show that "optimal models" (i.e., those that minimise the cost) can be written in terms of the quantum kernel as

$$f_{\mathrm{opt}}(x) = \sum_{m=1}^{M} \alpha_m \mathrm{tr}\left[\rho(x^m)\rho(x)\right] = \mathrm{tr}\left[\left(\sum_{m=1}^{M} \alpha_m \rho(x^m)\right)\rho(x)\right], \tag{1}$$

where $x^m, m = 1, \ldots, M$ is the training data and $\alpha_m \in \mathbb{R}$ (see Section VI B). Looking at the expression in the round brackets, this enables us to say something about optimal measurements for quantum models:

> *2. Quantum models that minimise typical machine learning cost functions have measurements that can be written as "kernel expansions in the data",* $\mathcal{M} = \sum_m \alpha_m \rho(x^m)$.

In other words, we are guaranteed that the best measurements for machine learning tasks only have $M$ degrees of freedom $\{\alpha_m\}$, rather than the $\mathcal{O}(2^{2n})$ degrees of freedom needed to express a general measurement on a standard $n$-qubit quantum computer. Even more, if we include a regularisation term into the cost function, the kernel defines entirely which models are actually penalised or preferred by regularisation. Since the kernel only depends on the way in which data is encoded into quantum states, one can conclude that data encoding fully defines the minima of a given cost function used to train quantum models (see Section VI C).

But how can we *find* the optimal model in Eq. (1)? We could use the near-term approach to quantum machine learning and simply train an ansatz, hoping that it learns the right measurement. But as illustrated in Figure 5, variational training typically only searches through a small subspace of all possible quantum models/measurements. This has a good reason: to train a circuit that can express any quantum model (and is hence guaranteed to find the optimal one) would require parameters for all $\mathcal{O}(2^{2n})$ degrees of freedom, which is intractable for all but toy models. However, also here kernel theory can help: not only is the optimal measurement defined by $M \ll 2^{2n}$ degrees of freedom, finding the optimal measurement has the same favourable scaling (see Section VI D) if we switch to a kernel-based training approach.

> *3. The problem of finding the optimal measurement for typical machine learning cost functions trained with $M$ data samples can be formulated as an $M$-dimensional optimisation problem.*

If the loss is convex, as is common in machine learning, the optimisation problem is guaranteed to be convex as well. Hence, under rather general assumptions, we are guaranteed that the "hard" problem of picking the best quantum model shown in Eq. (1) is tractable and of a simple structure, even without reverting to variational heuristics. In addition, convexity – the property that there is only one global minimum – may help with trainability problems like the notorious "barren plateaus" [16] in variational circuit training. If the loss function is the hinge loss, things reduce to a standard support vector machine with a quantum kernel, which is one of the algorithms proposed in [4] and [5].

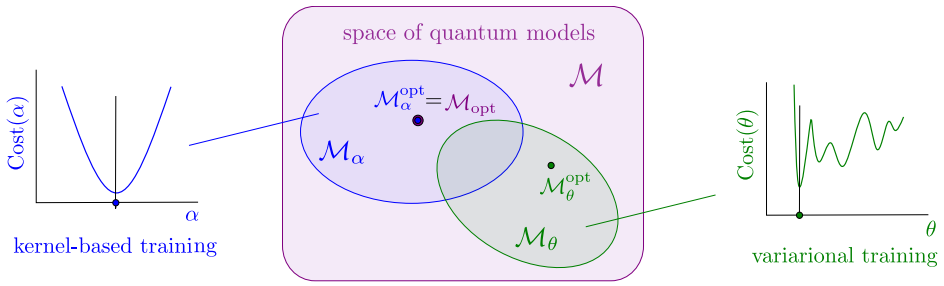


FIG. 5. **Kernel-based training vs. variational training.** Training a quantum model as defined here tries to find the optimal measurement $\mathcal{M}_{\text{opt}}$ over all possible quantum measurements. Kernel theory guarantees that in most cases this optimal measurement will have a representation that is a linear combination in the training data with coefficients $\alpha = (\alpha_1, \ldots, \alpha_M)$. Kernel-based training therefore optimises over the parameters $\alpha$ directly, effectively searching for the best model in an $M$-dimensional subspace spanned by the training data (blue). We are guaranteed that $\mathcal{M}_\alpha^{\text{opt}} = \mathcal{M}_{\text{opt}}$, and if the loss is convex this is the only minimum, which means that kernel-based training will find the best measurement out of all measurements. Variational training parametrises the measurement instead by a general ansatz that depends on $K$ parameters $\theta = (\theta_1, \ldots, \theta_K)$, and tries to find the optimal measurement $\mathcal{M}_\theta^{\text{opt}}$ in the subspace explored by the ansatz. This $\theta$-subspace is not guaranteed to contain the globally optimal measurement $\mathcal{M}_{\text{opt}}$, and optimisation is usually non-convex. We are therefore guaranteed that kernel-based training finds better or the same minima to variational training, but at the expense of having to compute pairwise distances of data points for training and classification.

Altogether, approaching quantum machine learning from a kernel perspective can have profound implications for the way we think about it. Firstly, most quantum models can be formulated as general support vector machines (in the sense of [20]) with a kernel evaluated on a quantum computer. As a corollary, we know that the measurements of optimal quantum models live in a low-dimensional subspace spanned by the training data, and that we can train in that space. Kernel-based training is guaranteed to find better minima – or as phrased here, measurements – than variational circuit training, at the expense of having to evaluate pair-wise distances of data points in feature space. (In the conclusion I will discuss how larger fault-tolerant quantum computers could potentially help with this as well!). Secondly, if the kernel defines the model, and the data encoding defines the kernel, we have to be very aware of the data encoding strategy we use in quantum machine learning – a step that has often taken the backseat over other parts of quantum models. Thirdly, since quantum models can always be rewritten as a classical model plus quantum kernel, the separation between classical and quantum machine learning lies only in the ability of quantum computers to implement classically hard kernels. The first steps into investigating such separations have been made in papers like [11, 12], but it is still unclear whether any *useful* applications turn out to be enabled solely by quantum computers.

The remainder of the paper will essentially follow the structure of this synopsis to discuss every statement in more mathematical detail.

## III. QUANTUM COMPUTING, FEATURE MAPS AND KERNELS

Let us start by laying the ground work for the kernel perspective on quantum machine learning. First I review the link between the process of encoding data into quantum states and feature maps, and construct the "quantum kernel" that we will use throughout the manuscript. I will then give some examples of data-encoding feature maps and quantum kernels, including a general description that allows us to understand these kernels via Fourier series.

### A. Encoding data into quantum states is a feature map

First, a few important concepts from quantum computing, which can be safely skipped by readers with a background in the field. Those who deem the explanations to be too casual shall be referred to the wonderful script by Michael Wolf [21].

*Quantum state.* According to quantum theory, the state of a quantum system is fully described by a length-1 vector $|\psi\rangle$ (or, more precisely, a ray represented by this vector) in a complex Hilbert space $\mathcal{H}$. The notation $|\cdot\rangle$ can be intimidating, but simply reminds of the fact that the Hilbert space has an inner product $\langle\cdot,\cdot\rangle$, which for Hilbert spaces describing quantum systems is denoted as $\langle\cdot|\cdot\rangle$, and that its vectors constitute "the right side" of the inner product. Quantum theory textbooks then introduce the

left side of the inner product as a functional $\langle\varphi|$ from a dual space $\mathcal{H}^*$ acting on elements of the original Hilbert space. Mainstream quantum computing considers rather simple quantum systems of $n$ binary subsystems called "qubits", whose Hilbert space is the $\mathbb{C}^{2^n}$. The dual space $\mathcal{H}^*$ can then be thought of as the space of complex $2^n$-dimensional "row vectors". A joint description of two quantum systems $|\psi\rangle$ and $|\varphi\rangle$ is expressed by the tensor product $|\psi\rangle \otimes |\phi\rangle$.

*Density matrix.* There is an alternative representation of a quantum state as a Hermitian operator called a *density matrix*. The density matrix corresponding to a state vector $|\psi\rangle$ reads

$$\rho = |\psi\rangle\langle\psi| . \tag{2}$$

If we represent quantum states as vectors in $\mathbb{C}^{2^n}$, then the corresponding density matrix is given by the outer product of a vector with itself – resulting in a matrix (and hence the name). The density matrix contains all observable information of $|\psi\rangle$, but is useful to model probability distributions $\{p_k\}$ over multiple quantum states $\{|\psi_k\rangle\langle\psi_k|\}$ as so-called *mixed states*

$$\rho = \sum_k p_k |\psi_k\rangle\langle\psi_k| , \tag{3}$$

without changing the equations of quantum theory. For simplicity I will assume that we are dealing with pure states in the following, but as far as I know everything should hold for mixed states as well.

*Quantum computations.* A quantum computation applies physical operations to quantum states, which – in analogy to classical circuits – are known as "quantum gates". The gates are applied to a small amount of qubits at a time. A collection of quantum gates (possibly followed by a measurement, which will be explained below) is called a *quantum circuit*. Any physical operation acting on the quantum system maps from a density matrix $\rho$ to another density matrix $\rho'$. In the most basic setting, such a transformation is described by a unitary operator $U$, with $\rho' = U^\dagger \rho U$, or $|\psi'\rangle = U|\psi\rangle$.[3] Unitary operations are length-preserving linear transformations, which is why we often say that a unitary "rotates" the quantum state. In the finite-dimensional case, a unitary operator can conveniently be represented by a unitary matrix, and the evolution of a quantum state becomes a matrix multiplication.

Consider a physical operation or quantum circuit $U(x)$ that depends on data $x \in \mathcal{X}$ from some data domain $\mathcal{X}$. For example, if the domain is the set of all bit strings of length $n$, the quantum circuit may apply specific operations only if bits are 1 and do nothing if they are 0. After the operation, the quantum state $|\phi(x)\rangle = U(x)|\psi\rangle$ depends on $x$. In other words, the data-dependent operation "encodes" or "embeds" $x$ into a vector $|\phi(x)\rangle$ from a Hilbert space (and I will use both terms interchangeably). This is a common definition of a feature map in machine learning, and we can say that *any data-dependent quantum computation implements a feature map.*

While from a quantum physics perspective it seems natural – and has been done predominantly in the early literature – to think of $x \to |\phi(x)\rangle$ as the feature map that links quantum computing to kernel methods, we will see below that quantum models are *not* linear in the Hilbert space of the quantum system [5], which means that the apparatus of kernel theory does not apply elegantly. Instead, I will define $x \to \rho(x)$ as the feature map and call it the *data-encoding feature map*. Note that consistent with the proposed naming scheme, the term "quantum feature map" would be misleading, since the result of the feature map is a state, which without measurement is just a mathematical concept.

**Definition 1** (Data-encoding feature map)**.** *Given a $n$-qubit quantum system with states $|\psi\rangle$, and let $\mathcal{F}$ be the space of complex-valued $2^n \times 2^n$-dimensional matrices equipped with the Hilbert-Schmidt inner product $\langle\rho,\sigma\rangle_\mathcal{F} = \mathrm{tr}\{\rho^\dagger\sigma\}$ for $\rho,\sigma \in \mathcal{F}$. The data-encoding feature map is defined as the transformation*

$$\phi : \mathcal{X} \to \mathcal{F}, \tag{4}$$

$$\phi(x) = |\phi(x)\rangle\langle\phi(x)| = \rho(x), \tag{5}$$

*and can be implemented by a data-encoding quantum circuit $U(x)$.*

While density matrices of qubit systems live in a subspace of $\mathcal{F}$ (i.e., the space of positive semi-definite trace-class operators), it will be useful to formally define the data-encoding feature space as above. Firstly, it makes sure that the feature space is a Hilbert space, and secondly, it allows measurements to live in the same space [21], which we will need to define linear models in $\mathcal{F}$. Section III C will discuss that this definition of the feature space is equivalent to the tensor product space of complex vectors $|\psi\rangle \otimes |\psi^*\rangle$ used in [12].

[3] The unitary operator is the quantum equivalent of a stochastic matrix which acts on vectors that represent discrete probability distributions.

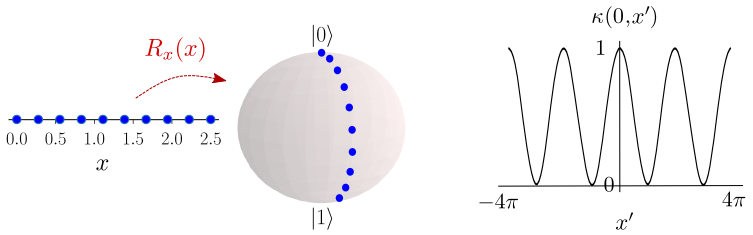


FIG. 6. **Example of a data-encoding feature map and quantum kernel.** A scalar input is encoded into a single-qubit quantum state, which is represented as a point on a Bloch sphere. The embedding uses a feature map facilitated by a Pauli-X rotation. As can be seen when plotting the quantum states encoding equidistant points on an interval, the embedding preserves the structure of the data rather well, but is periodic. The embedding gives rise to a quantum kernel $\kappa$. When we fix the first input at zero, we can visualise the distance measure, which is a squared cosine function.

### B. The data-encoding feature map gives rise to a kernel

Let us turn to kernels.

> *Kernels.* Unsurprisingly, the central concept of kernel theory are kernels, which in the context of machine learning are defined as real or complex-valued positive definite functions in two data points, $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{K}$, where $\mathbb{K}$ can be $\mathbb{C}$ or $\mathbb{R}$. For every such function we are guaranteed that there exist at least one feature map such that inner products of feature vectors $\phi(x)$ from the feature Hilbert space $\mathcal{F}$ form the kernel, $\kappa(x, x') = \langle \phi(x'), \phi(x) \rangle_{\mathcal{F}}$. Vice versa, every feature map gives rise to a kernel. The importance of kernels for machine learning is that they are a means of "computing" in feature space without ever accessing or numerically processing the vectors $\phi(x)$: everything we need to do in machine learning can be expressed by inner products of feature vectors, instead of the feature vectors themselves. In the cases that are practically useful, these inner products can be computed by a comparably simple function. This makes the computations in intractably large spaces tractable.

With the Hilbert-Schmidt inner product from Definition 1 we can immediately write down the kernel induced by the data-encoding feature map, which we will call the "quantum kernel" (since it is a function computed by a quantum computer):

**Definition 2** (Quantum kernel)**.** *Let $\phi$ be a data-encoding feature map over domain $\mathcal{X}$. A quantum kernel is the inner product between two data-encoding feature vectors $\rho(x), \rho(x')$ with $x, x' \in \mathcal{X}$,*

$$\kappa(x, x') = \mathrm{tr}\left[\rho(x')\rho(x)\right] = |\langle \phi(x') | \phi(x) \rangle|^2. \tag{6}$$

To justify the term "kernel" we need to show that the quantum kernel is indeed a positive definite function. The quantum kernel is a product of the complex-valued kernel $\kappa_c(x, x') = \langle \phi(x') | \phi(x) \rangle$ and its complex conjugate $\kappa_c(x, x')^* = \langle \phi(x) | \phi(x') \rangle = \langle \phi(x') | \phi(x) \rangle^*$. Since products of two kernels are known to be kernels themselves, we only have to show that the complex conjugate of a kernel is also a kernel. For any $x^m \in \mathcal{X}, m = 1 \dots M$, and for any $c_m \in \mathbb{C}$, we have

$$\begin{aligned}
\sum_{m,m'} c_m c_{m'}^* \left(\kappa_c(x^m, x^{m'})\right)^* &= \sum_{m,m'} c_m c_{m'}^* \left\langle \phi(x^m) \middle| \phi(x^{m'}) \right\rangle \\
&= \left(\sum_m c_m \langle \phi(x^m)|\right)\left(\sum_m c_m^* |\phi(x^m)\rangle\right) \\
&= \| \sum_m c_m^* |\phi(x^m)\rangle \|^2 \\
&\geq 0,
\end{aligned}$$

which means that the complex conjugate of a kernel is also positive definite.

**Example III.1.** *Consider an embedding that encodes a scalar input $x \in \mathbb{R}$ into the quantum state of a single qubit. The embedding is implemented by the Pauli-X rotation gate $R_X(x) = e^{-i\frac{x}{2}\sigma_x}$, where $\sigma_x$ is the Pauli-X operator. The data-encoding feature map is then given by $\phi : x \to \rho(x)$ with*

$$\rho(x) = \cos^2\left(\frac{x}{2}\right)|0\rangle\langle 0| + i\cos\left(\frac{x}{2}\right)\sin\left(\frac{x}{2}\right)|0\rangle\langle 1| - i\cos\left(\frac{x}{2}\right)\sin\left(\frac{x}{2}\right)|1\rangle\langle 0| + \sin^2\left(\frac{x}{2}\right)|1\rangle\langle 1|, \tag{7}$$

*and the quantum kernel becomes*

$$\kappa(x,x') = \left|\cos\left(\frac{x}{2}\right)\cos\left(\frac{x'}{2}\right) + \sin\left(\frac{x}{2}\right)\sin\left(\frac{x'}{2}\right)\right|^2 = \cos\left(\frac{x-x'}{2}\right)^2, \tag{8}$$

*which is a translation invariant squared cosine kernel. We will stick with this simple example throughout the following sections. It is illustrated in Figure 6.*

### C. Making sense of matrix-valued feature vectors

For readers that struggle to think of density matrices as feature vectors the data-encoding feature map (and further below, linear models) may be hard to visualise. I want to therefore insert a brief comment on an alternative version of the data-encoding feature map.

For all matters and purposes, the data-encoding feature map can be replaced by an alternative formulation

$$\phi_v : \mathcal{X} \to \mathcal{F}_v \subset \mathcal{H}\otimes\mathcal{H}^*, \tag{9}$$

$$\phi_v = |\phi(x)\rangle\otimes|\phi^*(x)\rangle\,, \tag{10}$$

where $|\phi^*(x)\rangle$ denotes the quantum state created from applying the complex conjugated (but not transposed) unitary $|\phi^*(x)\rangle = U^*(x)\,|0\rangle$ instead of $|\phi(x)\rangle = U(x)\,|0\rangle$, and $\mathcal{F}_v$ is the space of tensor products of a data-encoding Dirac vector with its complex conjugate. Note that since the complex conjugate of a unitary is a unitary, the unusual notation $|\phi^*(x)\rangle$ describes a valid quantum state which can be prepared by a physical circuit. The alternative feature space $\mathcal{F}_v$ is a subspace of the Hilbert space $\mathcal{H}\otimes\mathcal{H}^*$ with the property that inner products are real. One can show (but I won't do it here) that $\mathcal{F}_v$ is indeed a Hilbert space.

The inner product in this alternative feature space $\mathcal{F}_v$ is the absolute square of the inner product in the Hilbert space $\mathcal{H}$ of quantum states,

$$\langle\psi|\varphi\rangle_{\mathcal{F}_v} = |\,\langle\psi\,|\varphi\,\rangle_{\mathcal{H}}\,|^2, \tag{11}$$

and is therefore equivalent to the inner product in $\mathcal{F}$. This guarantees that it leads to the same quantum kernel.

The subscript $v$ refers to the fact that $|\phi(x)\rangle\otimes|\phi^*(x)\rangle$ is a *vectorisation* of $\rho(x)$, which reorders the $2^n$ matrix elements as a vector in $\mathbb{C}^{4n}$. To see this, let us revisit Example III.1 from above.

**Example III.2.** *Consider the embedding from Example III.1. The vectorised version of the data-encoding feature map is given by*

$$\phi_v : x \to |\phi(x)\rangle\otimes|\phi^*(x)\rangle = \left(\cos\left(\frac{x}{2}\right)|0\rangle - i\sin\left(\frac{x}{2}\right)|1\rangle\right)\otimes\left(\cos\left(\frac{x}{2}\right)|0\rangle + i\sin\left(\frac{x}{2}\right)|1\rangle\right) \tag{12}$$

$$= \begin{pmatrix} \cos^2\left(\frac{x}{2}\right) \\ i\cos\left(\frac{x}{2}\right)\sin\left(\frac{x}{2}\right) \\ -i\cos\left(\frac{x}{2}\right)\sin\left(\frac{x}{2}\right) \\ \sin^2\left(\frac{x}{2}\right) \end{pmatrix}, \tag{13}$$

*and one can verify easily that the inner product of two such vectors leads to the same kernel.*

Vectorised density matrices are common in the theory of open quantum systems [22], where they are written as $|\rho\rangle\!\rangle$ (see also the *Choi-Jamiolkowski isomorphism*). I will adopt this notation in Section VI B below to replace the Hilbert-Schmidt inner product $\mathrm{tr}\left[\rho^\dagger\sigma\right]$ with $\langle\!\langle\rho\,|\sigma\,\rangle\!\rangle$, which can be more illustrative at times. Note that the vectorised feature map, as opposed to Definition 1, cannot capture mixed quantum states and is therefore less powerful.

## IV. EXAMPLES OF QUANTUM KERNELS

To fill the definition of the quantum kernel with life, let us have a look at typical information encoding strategies or data embeddings in quantum machine learning, and the kernels they give rise to (following [4], and see Table I). Note that it has been shown that there are kernels that cannot be efficiently computed on classical computers [11].[4] As important as such results are, the question of quantum kernels that are actual useful for every-day problems is still wide open.

[4] The argument basically defines a feature map based on a computation that is conjectured by quantum computing research to be classically hard.

| **encoding** | **kernel** $\kappa(x, x')$ |
|---|---|
| basis encoding | $\delta_{x,x}$ |
| amplitude encoding | $\lvert\mathbf{x}^\dagger\mathbf{x}'\rvert^2$ |
| repeated amplitude encoding | $(\lvert\mathbf{x}^\dagger\mathbf{x}'\rvert^2)^r$ |
| rotation encoding | $\prod_{k=1}^N \lvert\cos(x'_k - x_k)\rvert^2$ |
| coherent state encoding | $e^{-\lvert\mathbf{x}-\mathbf{x}'\rvert^2}$ |
| general near-term encoding | $\sum_{s,t\in\Omega} e^{i\mathbf{s}\mathbf{x}} e^{i\mathbf{t}\mathbf{x}'} c_{\mathbf{s},\mathbf{t}}$ |

TABLE I. **Overview of data encoding strategies used in the literature and their quantum kernels.** If bold notation is used, the input domain is assumed to be the $\mathcal{X} \subseteq \mathbb{R}^N$.

### A. Data encoding that relates to classical kernels

The following strategies to encode data all have resemblance to kernels from the classical machine learning literature. This means that, sometimes up to an absolute square value, we can identify them with standard kernels such as the polynomial or Gaussian kernel. These kernels are plotted in Figure 7 using simulations of quantum computations implemented in the quantum machine learning software library PennyLane [23]. Note that I switch to bold notation when the input space is $\mathbb{C}^N$ or $\mathbb{R}^N$

**Basis encoding.** Basis encoding is possibly the most common information encoding strategy in qubit-based quantum computing. Inputs $x \in \mathcal{X}$ are assumed to be binary strings of length $n$, and $\mathcal{X} = \{0,1\}^{\otimes n}$. Every binary string has a unique integer representation $i_x = \sum_{k=0}^{n-1} 2^k x_k$. The data-encoding feature map maps the binary string to a computational basis state,

$$\phi : x \to |i_x\rangle\langle i_x| . \tag{14}$$

The quantum kernel is given by the Kronecker delta

$$\kappa(x, x') = |\langle i_{x'} | j_x \rangle|^2 = \delta_{x,x'}, \tag{15}$$

which is of course a very strict similarity measure on input space, and arguably not the best choice of data encoding for quantum machine learning tasks. Basis encoding requires $\mathcal{O}(n)$ qubits.

**Amplitude encoding.** Amplitude encoding assumes that $\mathcal{X} = \mathbb{C}^{2^n}$, and that the inputs are normalised as $\|\mathbf{x}\|^2 = \sum_i |x_i|^2 = 1$. The data-encoding feature map associates each input with a quantum state whose amplitudes in the computational basis are the elements in the input vector,

$$\phi : \mathbf{x} \to |\mathbf{x}\rangle\langle\mathbf{x}| = \sum_{i,j=1}^{N} x_i x_j^* |i\rangle\langle j| . \tag{16}$$

This data-encoding strategy leads to an identity feature map, which can be implemented by a non-trivial quantum circuit (for obvious reasons also known as "arbitrary state preparation"), which takes time $\mathcal{O}(2^n)$ [24]. The quantum kernel is the absolute square of the linear kernel

$$\kappa(\mathbf{x}, \mathbf{x}') = |\langle \mathbf{x}' | \mathbf{x} \rangle|^2 = |\mathbf{x}^\dagger \mathbf{x}'|^2. \tag{17}$$

It is obvious that this quantum kernel does not add much power to a linear model in the original feature space, and it is more of interest for theoretical investigations that want to eliminate the effect of the feature map. Amplitude encoding requires $\mathcal{O}(n)$ qubits.

**Repeated amplitude encoding.** Amplitude encoding can be repeated $r$ times,

$$\phi : \mathbf{x} \to |\mathbf{x}\rangle\langle\mathbf{x}| \otimes \cdots \otimes |\mathbf{x}\rangle\langle\mathbf{x}| \tag{18}$$

to get powers of the quantum kernel in amplitude encoding

$$\kappa(\mathbf{x}, \mathbf{x}') = (|\langle \mathbf{x}' | \mathbf{x} \rangle|^2)^r = (|(\mathbf{x}')^\dagger \mathbf{x}|^2)^r. \tag{19}$$

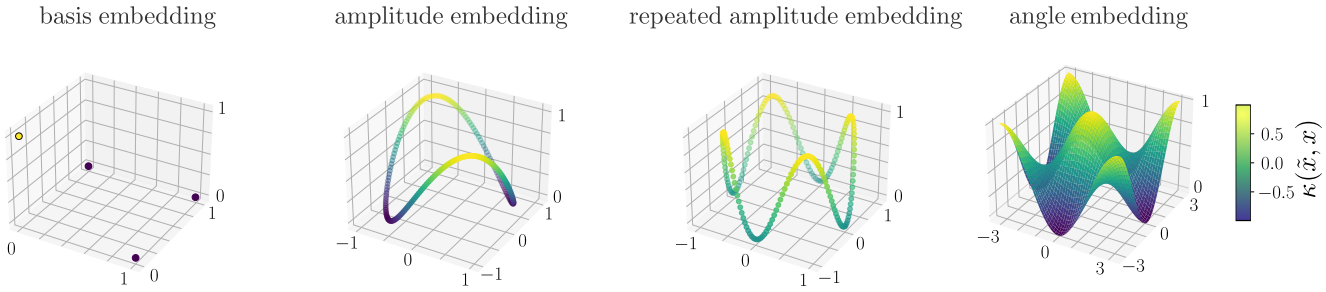


FIG. 7. **Quantum kernels of different data embeddings.** Plots of some of the functions $\kappa(\tilde{x}, x)$ for the kernels introduced above, using $\mathbf{x} = (x_1, x_2) \in \mathbb{R}^2$ for illustration purposes. The first entry $\tilde{\mathbf{x}}$ is fixed at $\tilde{\mathbf{x}} = (0, 0)$ for basis and rotation embedding, and at $\tilde{\mathbf{x}} = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$ for the variations of amplitude embedding. The second value is depicted as the x-y plane.

A constant non-homogenity can be added by extending the original input with constant dummy features. Repeated amplitude encoding requires $\mathcal{O}(rn)$ qubits.

**Rotation encoding.** Rotation encoding is a qubit-based embedding that assumes $\mathcal{X} = \mathbb{R}^n$ (where $n$ is again the number of qubits) without any normalisation condition. Since it is $2\pi$-periodic one may want to limit $\mathbb{R}^n$ to the hypercube $[0, 2\pi]^{\otimes n}$. The $i$th feature $x_i$ is encoded into the $i$th qubit via a Pauli rotation. For example, a Pauli-Y rotation puts the qubit into state $|q_i(x_i)\rangle = \cos(x_i)\,|0\rangle + \sin(x_i)\,|1\rangle$. The data-encoding feature map is therefore given by

$$\phi : \mathbf{x} \to |\phi(\mathbf{x})\rangle\langle\phi(\mathbf{x})| \;\text{ with }\; |\phi(\mathbf{x})\rangle = \sum_{q_1,\ldots,q_n=0}^{1} \prod_{k=1}^{n} \cos(x_k)^{q_k} \sin(x_k)^{1-q_k} \,|q_1, \ldots, q_n\rangle\,, \tag{20}$$

and the corresponding quantum kernel is related to the cosine kernel:

$$\kappa(\mathbf{x}, \mathbf{x}') = \prod_{k=1}^{n} |\sin x_k \sin x_k' + \cos x_k \cos x_k'|^2 = \prod_{k=1}^{n} |\cos(x_k - x_k')|^2. \tag{21}$$

Rotation encoding requires $\mathcal{O}(n)$ qubits.

**Coherent state encoding.** Coherent states are known in the field of quantum optics as a description of light modes. Formally, they are superpositions of so called *Fock states*, which are basis states from an infinite-dimensional discrete basis $\{|0\rangle, |1\rangle, |2\rangle, ...\}$, instead of the binary basis of qubits. A coherent state has the form

$$|\alpha\rangle = e^{-\frac{|\alpha|^2}{2}} \sum_{k=0}^{\infty} \frac{\alpha^k}{\sqrt{k!}} \,|k\rangle\,, \tag{22}$$

for $\alpha \in \mathbb{C}$. Encoding a real scalar input $x_i \in \mathbb{R}$ into a coherent state $|\alpha_{x_i}\rangle$, corresponds to a data-encoding feature map with an infinite-dimensional feature space,

$$\phi : x_i \to |\alpha_{x_i}\rangle\langle\alpha_{x_i}|\,,\;\text{ with }\; |\alpha_{x_i}\rangle = e^{-\frac{|x_i|^2}{2}} \sum_{k=0}^{\infty} \frac{x_i^k}{\sqrt{k!}} \,|k\rangle\,. \tag{23}$$

We can encode a real vector $\mathbf{x} = (x_1, ..., x_n)$ into $n$ joint coherent states,

$$|\alpha_{\mathbf{x}}\rangle\langle\alpha_{\mathbf{x}}| = |\alpha_{x_1}\rangle\langle\alpha_{x_1}| \otimes \cdots \otimes |\alpha_{x_n}\rangle\langle\alpha_{x_n}|\,. \tag{24}$$

The quantum kernel is a Gaussian kernel [7]:

$$\kappa(\mathbf{x}, \mathbf{x}') = \left| e^{-\left(\frac{|\mathbf{x}|^2}{2} + \frac{|\mathbf{x}'|^2}{2} - \mathbf{x}^T\mathbf{x}'\right)} \right|^2 = e^{-|\mathbf{x}-\mathbf{x}'|^2} \tag{25}$$

Preparing coherent states can be done with displacement operations in quantum photonics.

### B. Fourier representation of the quantum kernel

It is suspicious that all embeddings plotted in Figure 7 have a periodic, trigonometric structure. This is a fundamental characteristic of how physical parameters enter quantum states. To see this we will define a general class of embeddings (also called "time-evolution encoding") that is used a lot in near-term quantum machine learning, and which includes all examples above if we allow for classical pre-processing of the features. This strategy assumes that $\mathcal{X} = \mathbb{R}^N$ for some arbitrary $N$ (whose relation to the number of qubits $n$ depends on the embedding), which means that I will stick to the bold notation. The embedding of $x_i$ is executed by gates of the form $e^{-ix_iG_i}$ where $G_i$ is $d_i \leq 2^n$-dimensional Hermitian operator called the *generating Hamiltonian*. For the popular choice of Pauli rotations, $G_i = \frac{1}{2}\sigma$ with the Pauli operator $\sigma \in \{\sigma_z, \sigma_y, \sigma_z\}$. The gates can be applied to different qubits as in rotation encoding, or to the same qubits, and to be general we allow for arbitrary quantum computations between each encoding gate.

Refs. [25] and [26] showed that the Dirac vectors $|\phi(\mathbf{x})\rangle$ can be represented in terms of periodic functions of the form $e^{ix_i\omega}$, where $\omega \in \mathbb{R}$ can be interpreted as a frequency. The frequencies involved in the construction of the data-encoding feature vectors are solely determined by the generating Hamiltonians $\{G_i\}$ of the gates that encode the data. For popular choices of Hamiltonians, the frequencies $\omega$ are integer-valued, which means that the feature space is constructed from *Fourier basis functions* $e^{ix_in}, n \in \mathbb{Z}$. This allows us to describe and analyse the quantum kernel with the tools of Fourier analysis.

Let me state the result for the simplified case that each input $x_i$ is only encoded once, and that all the encoding Hamiltonians are the same ($G_1 = \cdots = G_N = G$). The proof is deferred to Appendix A, which also shows how our example of Pauli-X encoding can be cast as a Fourier series.

**Theorem 1** (Fourier representation of the quantum kernel)**.** *Let $\mathcal{X} = \mathbb{R}^N$ and $S(\mathbf{x})$ be a quantum circuit that encodes the data inputs $\mathbf{x} = (x_1, \ldots, x_N) \in \mathcal{X}$ into a $n$-qubit quantum state $S(\mathbf{x})|0\rangle = |\phi(\mathbf{x})\rangle$ via gates of the form $e^{-ix_iG}$ for $i = 1, \ldots, N$. Without loss of generality $G$ is assumed to be a $d \leq 2^n$-dimensional diagonal operator with spectrum $\lambda_1, \ldots, \lambda_d$. Between such data-encoding gates, and before and after the entire encoding circuit, arbitrary unitary evolutions $W^{(1)}, \ldots, W^{(N+1)}$ can be applied, so that*

$$S(\mathbf{x}) = W^{(N+1)}e^{-ix_NG}W^{(N)} \ldots W^{(2)}e^{-ix_1G}W^{(1)}. \tag{26}$$

*The quantum kernel $\kappa(\mathbf{x}, \mathbf{x}')$ can be written as*

$$\kappa(\mathbf{x}, \mathbf{x}') = \sum_{\mathbf{s},\mathbf{t}\in\Omega} e^{i\mathbf{s}\mathbf{x}}e^{i\mathbf{t}\mathbf{x}'}c_{\mathbf{st}}, \tag{27}$$

*where $\Omega \subseteq \mathbb{R}^N$, and $c_{\mathbf{st}} \in \mathbb{C}$. For every $\mathbf{s}, \mathbf{t} \in \Omega$ we have $-\mathbf{s}, -\mathbf{t} \in \Omega$ and $c_{\mathbf{st}} = c^*_{-\mathbf{s}-\mathbf{t}}$, which guarantees that the quantum kernel is real-valued.*

While the conditions of this theorem may sound restrictive at first, it includes a fairly general class of quantum models. The standard way to control a quantum system is to apply an evolution of Hamiltonian $G$ for time $t$, which is exactly described by the form $e^{-itG}$. The time $t$ is associated with the input to the quantum computer (which may be the original input $x \in \mathcal{X}$ or the result of some pre-processing, in which case we can just redefine the dataset to be the pre-processed one). In short, most quantum kernels will be of the form shown in Eq. (27).

Importantly, for the class of Pauli generators, the kernel becomes a Fourier series:

**Corollary 1.1** (Fourier series representation of the quantum kernel)**.** *For the setting described in Theorem 1, if the eigenvalue spectrum of $G$ is such that any difference $\lambda_i - \lambda_j$ for $i, j = 1, \ldots, d$ is in $\mathbb{Z}$, then $\Omega$ becomes the set of $N$-dimensional integer-valued vectors $\mathbf{n} = (n_1, \ldots, n_N)$, $n_1, \ldots n_N \in \mathbb{Z}$. In this case the quantum kernel is a multi-dimensional Fourier series,*

$$\kappa(\mathbf{x}, \mathbf{x}') = \sum_{\mathbf{n},\mathbf{n}'\in\Omega} e^{i\mathbf{n}\mathbf{x}}e^{i\mathbf{n}'\mathbf{x}'}c_{\mathbf{n},\mathbf{n}'}, \tag{28}$$

Expressions (27) and (28) reveal a lot about the structure of quantum kernels, for example that they are not necessarily translation invariant, $\kappa(\mathbf{x}, \mathbf{x}') \neq g(\mathbf{x} - \mathbf{x}')$, unless the data-encoding strategy leads to $c_{\mathbf{st}} = \tilde{c}_{\mathbf{st}}\delta_{\mathbf{st}} = c_{\mathbf{s}}$ and

$$\kappa(\mathbf{x}, \mathbf{x}') = \sum_{\mathbf{s}\in\Omega} e^{i\mathbf{s}(\mathbf{x}-\mathbf{x}')}\tilde{c}_{\mathbf{s}}. \tag{29}$$

Since $e^{-ix_iG}e^{ix'_iG} = e^{-i(x_i-x'_i)G}$, this is true for all data embeddings that encode each original input into a separate physical subsystem, like rotation encoding introduced above.

It is an interesting question if this link between data embedding and Fourier basis functions given to us by physics can help design particularly suitable kernels for applications, or be used to control smoothness properties of the kernel in a useful manner.

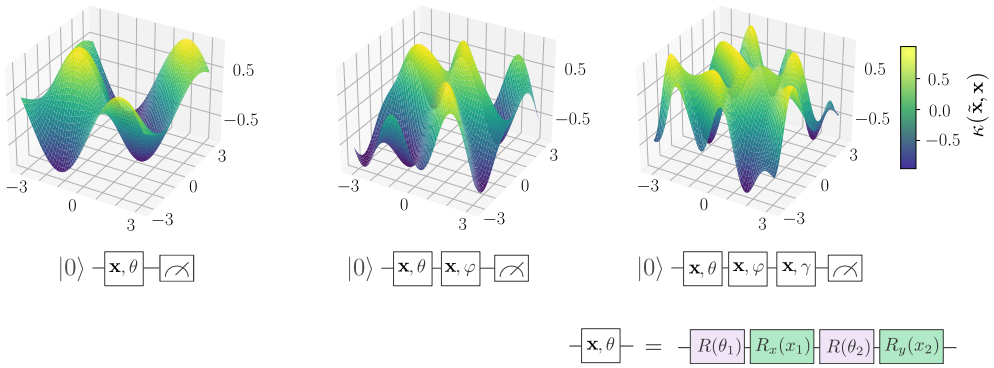


FIG. 8. **Kernels generated by rotation embeddings.** Plots of the quantum kernel $\kappa(\tilde{\mathbf{x}}, \mathbf{x})$ with $\tilde{\mathbf{x}} = (0, 0)$ using a very general data encoding strategy that repeats the input encoding into a single qubit one, two and three times. It is obvious that the repetition decreases the smoothness of the kernel by increasing the Fourier basis functions from which the kernel is inherently constructed.

## V. QUANTUM MODELS AND REPRODUCING KERNEL HILBERT SPACES

I will now discuss the observation that quantum models are linear models in the feature space $\mathcal{F}$ of the data-encoding feature map. This automatically allows us to apply the results of kernel methods to quantum machine learning. A beautiful summary of these results can be found in [20] and [27], which serve as a basis for many of the following insights.

### A. Quantum models are linear models in feature space

First, let us define a quantum model. For this we need measurements.

*Measurements.* In quantum computing, a measurement produces the observable result of a quantum circuit, and can therefore be seen as the final step of a quantum algorithm[5]. Mathematically speaking, a measurement corresponds to a Hermitian operator $\mathcal{M}$ acting on vectors in the Hilbert space of the quantum system $\mathcal{H}$. Just like density matrices, measurement operators can be represented as elements of the space of $2^n \times 2^n$-dimensional complex matrices [21], and therefore live in a subspace of the data-encoding feature space $\mathcal{F}$. This will become quite crucial below.

A Hermitian operator can always be diagonalised and written as

$$\mathcal{M} = \sum_i \mu_i \, |\mu_i\rangle\langle\mu_i| \, , \tag{30}$$

where $\mu_i$ are the eigenvalues of $\mathcal{M}$ and $\{|\mu_i\rangle\}$ is an orthonormal basis in the Hilbert space $\mathcal{H}$ of the quantum system. Note that $|\mu_i\rangle\langle\mu_i|$ is an outer product, and can be thought of as a (density) matrix.

The apparatus of quantum theory allows us to compute expected outcomes or *expectations* of measurement results. Such expectations derive from expressing the quantum state in the eigenbasis of the measurement operator, $|\psi\rangle = \sum_i \langle \mu_i \, | \psi \rangle \, |\mu_i\rangle$, and using the fact that $\mathcal{M} \, |\mu_i\rangle = \mu_i \, |\mu_i\rangle$ and $\langle \mu_i \, | \mu_i \rangle = 1$:

$$\mathrm{tr}\left[\rho \mathcal{M}\right] = \langle\psi|\mathcal{M}|\psi\rangle = \sum_{i,j} \langle \psi \, | \mu_j \rangle \, \langle \mu_i \, | \psi \rangle \, \langle \mu_j | \, \mathcal{M} \, | \mu_i \rangle = \sum_i | \, \langle \psi \, | \mu_i \rangle \, |^2 \mu_i = \sum_i p(\mu_i) \mu_i . \tag{31}$$

The above used the "Born rule", which states that the probability of measuring outcome $\mu_i$ is given by

$$p(\mu_i) = |\langle \mu_i | \psi \rangle|^2 . \tag{32}$$

It is clear that the right hand side of Eq. (31) is an expectation of a random variable in the classical sense of probability theory, but the probabilities themselves are computed by an unusual mathematical

[5] An important exception is when the outcome of a measurement is used to influence the quantum circuit itself, but I do not consider those complications here.

framework. Finally, it is good to know that the expectation of a measurement $\mathcal{M}_\varphi = |\varphi\rangle\langle\varphi|$ (where $|\varphi\rangle$ is an arbitrary quantum state) gives us the *overlap* of $|\varphi\rangle$ and $|\psi\rangle$,

$$\mathrm{tr}\left[\rho\mathcal{M}_\varphi\right] = \langle\psi|\mathcal{M}_\varphi|\psi\rangle = |\langle\varphi|\psi\rangle|^2. \tag{33}$$

Note that only because we can write down a measurement mathematically, we cannot necessarily implement it efficiently on a quantum computer. However, for measurements of type $\mathcal{M}_\varphi$ there is a very efficient routine called the SWAP test to do so, if we can prepare the corresponding state efficiently. In practice, more complicated measurements are implemented by applying a circuit $W$ to the final quantum state, followed by a simple measurement (such as the well-known Pauli-Z measurement $\sigma_z$ that probes the state of qubits, which effectively implements $\mathcal{M} = W^\dagger\sigma_z W$).

Of course, actual quantum computers can only ever produce an estimate of the above statistical properties, namely by repeating the entire computation $K$ times and computing the empirical probability/frequency or the empirical expectation $\frac{1}{K}\sum_{i=1}^{K}\mu_i$. However, repeating a fixed computation tens of thousands of times can be done in a fraction of a second on most hardware platforms, and only leads to a small constant overhead.

We can define a quantum model as a measurement performed on a data-encoding state:

**Definition 3** (Quantum model)**.** *Let $\rho(x)$ be a quantum state that encodes classical data $x \in \mathcal{X}$ and $\mathcal{M}$ a Hermitian operator representing a quantum measurement. A quantum model is the expectation of the quantum measurement as a function of the data input,*

$$f(x) = \mathrm{tr}\left[\rho(x)\mathcal{M}\right]. \tag{34}$$

*The space of all quantum models contains functions $f : \mathcal{X} \to \mathbb{R}$. For pure-state embeddings with $\rho(x) = |\phi(x)\rangle\langle\phi(x)|$, this simplifies to*

$$f(x) = \langle\phi(x)|\,\mathcal{M}\,|\phi(x)\rangle. \tag{35}$$

As mentioned above, this definition is very general, but does not consider the important class of generative quantum models.

**Example V.1.** *Getting back to the standard example of the Pauli-X rotation encoding, we can upgrade it to a full quantum model with parametrised measurement by applying an additional arbitrary rotation $R(\theta_1,\theta_2,\theta_3)$, which is parametrised by three trainable angles and is expressive enough to represent any single-qubit computation. After this, we measure in the Pauli-Z basis, yielding the overall quantum model:*

$$f(x) = \mathrm{tr}\left[\rho(x)\mathcal{M}(\theta_1,\theta_2,\theta_3)\right] = \langle\phi(x)|\,\mathcal{M}(\theta_1,\theta_2,\theta_3)\,|\phi(x)\rangle, \tag{36}$$

*with measurement $\mathcal{M}(\theta_1,\theta_2,\theta_3) = R^\dagger(\theta_1,\theta_2,\theta_3)\sigma_z R(\theta_1,\theta_2,\theta_3)$,*

$$R(\theta_1,\theta_2,\theta_3) = \begin{pmatrix} e^{i(-\frac{\theta_1}{2}-\frac{\theta_3}{2})}\cos(\frac{\theta_2}{2}) & -e^{i(-\frac{\theta_1}{2}+\frac{\theta_3}{2})}\sin(\frac{\theta_2}{2}) \\ e^{i(\frac{\theta_1}{2}-\frac{\theta_3}{2})}\sin(\frac{\theta_2}{2}) & e^{i(\frac{\theta_1}{2}+\frac{\theta_3}{2})}\cos(\frac{\theta_2}{2}) \end{pmatrix} \tag{37}$$

*and $|\phi(x)\rangle = R_x(x)\,|0\rangle$. One can use a computer-algebra system (or, for the patient among us, lengthy calculations) to verify that the quantum model is equivalent to the function*

$$f(x) = \cos(\theta_2)\cos(x) - \sin(\theta_1)\sin(\theta_2)\sin(x), \tag{38}$$

*and hence independent of the third parameter.*

Next, let us define what a linear (machine learning) model in feature space is:

**Definition 4** (Linear model)**.** *Let $\mathcal{X}$ be a data domain and $\phi : \mathcal{X} \to \mathcal{F}$ a feature map. We call any function*

$$f(x) = \langle\phi(x), w\rangle_{\mathcal{F}}, \tag{39}$$

*with $w \in \mathcal{F}$ a linear model in $\mathcal{F}$.*

From these two definitions we immediately see that:

**Theorem 2** (Quantum models are linear models in data-encoding feature space)**.** *Let $f(x) = \mathrm{tr}\,[\rho\mathcal{M}]$ be a quantum model with feature map $\phi : x \in \mathcal{X} \to \rho(x) \in \mathcal{F}$ and data domain $\mathcal{X}$. The quantum model $f$ is a linear model in $\mathcal{F}$.*

It is interesting to note that the measurement $\mathcal{M}$ can always be expressed as a linear combination $\sum_k \gamma_k \rho(x^k)$ of data-encoding states $\rho(x^k)$ where $x^k \in \mathcal{X}$.

**Theorem 3** (Quantum measurements are linear combinations of data-encoding states)**.** *Let $f_{\mathcal{M}}(x) = \mathrm{tr}\,[\rho\mathcal{M}]$ be a quantum model. There exists a measurement $\mathcal{M}_{\mathrm{exp}} \in \mathcal{F}$ of the form*

$$\mathcal{M}_{\mathrm{exp}} = \sum_k \gamma_k \rho(x^k) \tag{40}$$

*with $x^k \in \mathcal{X}$, such that $f_{\mathcal{M}}(x) = f_{\mathcal{M}_{\mathrm{exp}}}(x)$ for all $x \in \mathcal{X}$.*

*Proof.* We can divide $\mathcal{M}$ into the part that lies in the image of $\mathcal{X}$ and the remainder $R$,

$$\mathcal{M} = \mathcal{M}_{\mathrm{exp}} + R. \tag{41}$$

Since the trace is linear, we have:

$$\mathrm{tr}\,[\rho(x)\mathcal{M}] = \mathrm{tr}\,[\rho(x)\mathcal{M}_{\mathrm{exp}}] + \mathrm{tr}\,[\rho(x)R]\,. \tag{42}$$

The data-encoding state $\rho(x)$ only has contributions in $\mathcal{F}$, which means that the inner product $\mathrm{tr}\,[\rho(x)R]$ is always zero. □

Below we will see that optimal measurements with respect to typical machine learning cost functions can be expanded in the training data only.

Note that the fact that a quantum model can be expressed as a linear model in the feature space does *not* mean that it is linear in the Hilbert space of the Dirac vectors $|\phi(x)\rangle$, nor is it linear in the data input $x$. As mentioned before, in the context of variational circuits the measurement usually depends on trainable parameters, which is realised by applying a parametrised quantum operation or circuit that "rotates" the basis of a fixed measurement. Variational quantum models are also not necessarily linear in their actual trainable parameters.

As a last comment for readers that prefer the vectorised version of the data-encoding feature map, by writing the measurement operator $\mathcal{M} = \sum_i \mu_i |\mu_i\rangle\langle\mu_i|$ in its eigenbasis, we can likewise write a quantum model as the inner product of a vectorised feature vector $|\phi(x)\rangle \otimes |\phi^*(x)\rangle \in \mathcal{F}_v$ with some other vector $\sum_i \mu_i |\mu_i\rangle \otimes |\mu_i\rangle \in \mathcal{F}_v$.

$$\begin{aligned} f(x) &= \langle\phi(x)|\,\mathcal{M}\,|\phi(x)\rangle && (43)\\ &= \sum_i \mu_i |\,\langle\mu_i\,|\phi(x)\,\rangle\,|^2 && (44)\\ &= \Big(\,\langle\phi(x)| \otimes \langle\phi^*(x)|\,\Big)\Big(\sum_i \mu_i\,|\mu_i\rangle \otimes |\mu_i^*\rangle\,\Big), && (45)\end{aligned}$$

or using the vectorised density matrix notation introduced above,

$$f(x) = \langle\!\langle \rho(x)\,|w\,\rangle\!\rangle\,, \tag{46}$$

with $w = \sum_i \mu_i\,|\rho_i\rangle\!\rangle$.

### B. The RKHS of the quantum kernel and the space of quantum models are equivalent

So far we were dealing with two different kinds of Hilbert spaces: The Hilbert space $\mathcal{H}$ of the quantum system, and the feature space $\mathcal{F}$ that contains the embedded data. I will now construct yet another feature space for the quantum kernel, but one derived directly from the kernel and with no further notion of a quantum model. This time the feature space is a Hilbert space $F$ of *functions*, and due to its special construction it is called the *reproducing kernel Hilbert space* (RKHS). The relevance of this feature space is that the functions it contains turn out to be exactly the quantum model functions $f$ (which is a bit surprising at first: this feature space contains linear models defined in an equivalent feature space!).

The RKHS $F$ of the quantum kernel can be defined as follows (as per Moore- Aronsajn's construction[6]):

---

[6] See also `http://www.stats.ox.ac.uk/~sejdinov/teaching/atml14/Theory_2014.pdf` for a great overview.

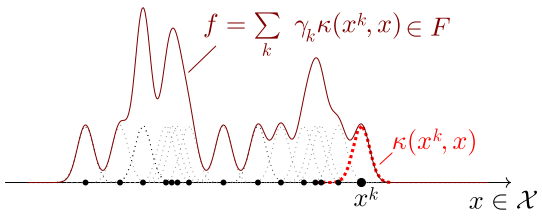


FIG. 9. **Intuition for the functions living in the reproducing kernel Hilbert space (RKHS).** The RKHS $F$ contains functions that are linear combinations of kernel functions where one "slot" is fixed in a possible data sample $x^k \in \mathcal{X}$. This illustration of one such function $f \in F$, using a Gaussian kernel, shows how the kernel regulates the "smoothness" of the functions in $F$, as a wider kernel will simplify $f$. Since the RKHS is equivalent to the space of linear models that it has been derived from, the kernel fundamentally defines the class of functions that the linear model can express.

**Definition 5** (Reproducing kernel Hilbert space)**.** *Let $\mathcal{X} \neq \varnothing$. The reproducing kernel Hilbert space of a kernel $\kappa$ over $\mathcal{X}$ is the Hilbert space $F$ created by completing the span of functions $f : \mathcal{X} \to \mathbb{R}$, $f(\cdot) = \kappa(x, \cdot)$, $x \in \mathcal{X}$ (i.e., including the limits of Cauchy series). For two functions $f(\cdot) = \sum_i \alpha_i \kappa(x^i, \cdot)$, $g(\cdot) = \sum_j \beta_j \kappa(x^j, \cdot) \in F$, the inner product is defined as*

$$\langle f, g \rangle_F = \sum_{ij} \alpha_i \beta_j \kappa(x^i, x^j), \tag{47}$$

*with $\alpha_i, \beta_j \in \mathbb{R}$.*

Note that according to Theorem 1 the "size" of the space of common quantum models, and likewise the RKHS of the quantum kernel, are fundamentally limited by the generators of the data-encoding gates. If we consider $\kappa$ as the quantum kernel, the definition of the inner product reveals with

$$\langle \kappa(x, \cdot), \kappa(x', \cdot) \rangle_F = \kappa(x, x'), \tag{48}$$

that $x \to \kappa(x, \cdot)$ is a feature map of this kernel (but one mapping data to *functions* instead of matrices, which feels a bit odd at first). In this sense, $F$ can be regarded as an alternative feature space to $\mathcal{F}$. The name of this unique feature space comes from the *reproducing property*

$$\langle f, \kappa(x, \cdot) \rangle_F = f(x) \text{ for all } f \in F, \tag{49}$$

which also shows that the kernel is the evaluation functional $\delta_x$ which assigns $f$ to $f(x)$. An alternative definition of the RKHS is the space in which the evaluation functional is bounded, which gives the space a lot of favourable properties from a mathematical perspective.

To most of us, the definition of an RKHS is terribly opaque when first encountered, so a few words of explanation may help (see also Figure 9). One can think of the RKHS as a space whose elementary functions $\kappa(x, \cdot)$ assign a distance measure to every data point. Functions of this form were also plotted in Figure 7 and 8. By feeding another data point $x'$ into this "similarity measure", we get the distance between the two points. As a vector space, $F$ also contains linear combinations of these building blocks. The functions living in $F$ are therefore linear combinations of data similarities, just like for example kernel density estimation constructs a smooth function by adding Gaussians centered in the data. The kernel then regulates the "resolution" of the distance measure, for example by changing the variance of the Gaussian.

Once one gets used to this definition, it is immediately apparent that the functions living in the RKHS of the quantum kernel are what we defined as quantum models:

**Theorem 4.** *Functions in the RKHS $F$ of the quantum kernel are linear models in the data-encoding feature space $\mathcal{F}$ and vice versa.*

*Proof.* The functions in the RKHS of the quantum kernel are of the form $f(\cdot) = \sum_k \gamma_k \kappa(x^k, \cdot)$, with $x^k \in \mathcal{X}$. We get

$$\begin{aligned} f(x) &= \sum_k \gamma_k \kappa(x^k, x) && (50)\\ &= \sum_k \gamma_k \operatorname{tr}\left[\rho(x^k)\rho(x)\right] && (51)\\ &= \operatorname{tr}\left[\sum_k \gamma_k \rho(x^k)\rho(x)\right] && (52)\\ &= \operatorname{tr}\left[\mathcal{M}\rho(x)\right]. && (53) \end{aligned}$$

Using Theorem 3 we know that all quantum models can be expressed by measurements $\sum_k \gamma_k \rho(x^k)$, and hence by functions in the RKHS. □

In fact, the above observation applies to *any* linear model in a feature space that gives rise to the quantum kernel (see Theorem 4.21 in [20]).

As a first taste of how the connection of quantum models and kernel theory can be exploited for quantum machine learning, consider the question whether quantum models are universal function approximators. If quantum models are universal, the RKHS of the quantum kernel must be universal (or dense in the space of functions we are interested in). This leads to the definition of a universal kernel (see [20] Definition 4.52):

**Definition 6** (Universal kernel)**.** *A continuous kernel $\kappa$ on a compact metric space $\mathcal{X}$ is called universal if the RKHS $F$ of $\kappa$ is dense in $C(\mathcal{X})$, i.e., for every function $g$ in the set of functions $C(\mathcal{X})$ mapping from elements in $\mathcal{X}$ to a scalar value, and for all $\epsilon > 0$ there exists an $f \in F$ such that*

$$\|f - g\|_\infty \leq \epsilon. \tag{54}$$

The reason why this is useful is that there are a handful of known necessary conditions for a kernel to be universal, for example if its feature map is injective (see [20] for more details). This immediately excludes quantum models defined on the data domain $\mathcal{X} = \mathbb{R}$ which use single-qubit Pauli rotation gates of the form $e^{ix\sigma}$ (with $\sigma$ a Pauli matrix) to encode data: since such rotations are $2\pi$-periodic, two different $x, x' \in \mathcal{X}$ get mapped to the same data-encoding state $\rho(x)$. In other words, and to some extent trivially so, on a data domain that extends beyond the periodicity of a quantum model we never have a chance for universal function approximation. Another example for universal kernels are kernels of the form $\kappa(x, x') = \sum_{k=1}^{\infty} c_k \langle x', x\rangle^k$ (see [20] Corollary 4.57). Vice versa, the universality proof for a type of quantum model in [26] suggests that some quantum kernels of the form (1) are universal in the asymptotic limit of exponentially large circuits.

I want to finish with a final note about the relation between "wavefunctions" and functions in the RKHS of quantum systems (see also the appendix of [4]). Quantum states are sometimes called "wavefunctions", since an alternative definition of the Hilbert space of a quantum system is the space of functions $f(\cdot) = \psi(\cdot)$ which map a measurement outcome $i$ corresponding to basis state $|i\rangle$ to an "amplitude" $\psi(i) = \langle i|\psi\rangle$. (The dual basis vector $\langle i|$ can here be understood as the evaluating functional $\delta_i$ which returns this amplitude.) Hence, the Hilbert space of a quantum system can be written as a space of functions mapping from $\{i\} \to \mathbb{C}$. But the functions that we are interested in for machine learning are functions *in the data*, not in the possible measurement outcomes. This means that the Hilbert space of the quantum system is only equivalent to the RKHS of a quantum machine learning model if we associate data with the measurement outcomes. This is true for many proposals of generative quantum machine learning models [18, 28], and it would be interesting to transfer the results to this setting.

## VI. TRAINING QUANTUM MODELS

While the question of universality addresses the expressivity of quantum models, the remaining sections will look at questions of trainability and optimisation, for which the kernel perspective has the most important results to offer. Notably, we will see that the optimal measurements of quantum models for typical machine learning cost functions only have relatively few degrees of freedom. Similarly, the process of finding these optimal models (i.e., training over the space of all possible quantum models) can be formulated as a low-dimensional optimisation problem. Most of the results are based on the fact that for kernel methods, the task of training a model is equivalent to optimising over the model's corresponding RKHS.

### A. Optimising quantum models is equivalent to optimising over the RKHS

In machine learning we want to find optimal models, or those that minimise the cost functions derived from learning problems. This process is called *training*. From a learning theory perspective, training can be phrased as *regularised empirical risk minimisation*, and the problem of training quantum models can be cast as follows:

**Definition 7** (Regularised empirical risk minimisation of quantum models)**.** *Let $\mathcal{X}, \mathcal{Y}$ be data input and output domains, $p$ a probability distribution on $\mathcal{X}$ from which data is drawn, and $L : \mathcal{X} \times \mathcal{Y} \times \mathbb{R} \to [0, \infty)$ a loss function that quantifies the quality of the prediction of a quantum model $f(x) = \mathrm{tr}\,[\rho(x)\mathcal{M}]$. Let*

$$\mathcal{R}_L(f) = \int_{\mathcal{X}\times\mathcal{Y}} L(x, y, f(x))\, \mathrm{d}p(x, y) \tag{55}$$

*be the expected loss (or "risk") of $f$ under $L$, where $L$ may depend explicitly on $x$. Since $p$ is unknown, we approximate the risk by the empirical risk*

$$\hat{\mathcal{R}}_L(f) = \frac{1}{M}\sum_{m=1}^{M} L(x^m, y, f(x^m)). \tag{56}$$

*Regularised empirical risk minimisation of quantum models is the problem of minimising the empirical risk over all possible quantum models while also minimising the norm of the measurement $\mathcal{M}$,*

$$\inf_{\mathcal{M}\in\mathcal{F}} \lambda\|\mathcal{M}\|_{\mathcal{F}}^2 + \hat{\mathcal{R}}_L(\mathrm{tr}\left[\rho(x)\mathcal{M}\right]), \tag{57}$$

*where $\lambda \in \mathbb{R}^+$ is a positive hyperparameter that controls the strength of the regularisation term.*

We saw in Section V that quantum models are equivalent to functions in the RKHS of the quantum kernel, which allows us to replace the term $\hat{\mathcal{R}}_L(\mathrm{tr}\left[\rho(x)\mathcal{M}\right])$ in the empirical risk by $\hat{\mathcal{R}}_L(f)$, $f \in F$.

But what about the regularisation term? Since with Theorem 3 we can write

$$\begin{aligned}
\|\mathcal{M}\|_{\mathcal{F}}^2 &= \mathrm{tr}\left[\mathcal{M}^2\right] && (58)\\
&= \sum_{ij}\gamma_i\gamma_j \mathrm{tr}\left[\rho(x^i)\rho(x^j)\right] && (59)\\
&= \sum_{ij}\gamma_i\gamma_j \kappa(x^i, x^j) && (60)\\
&= \langle \sum_i \gamma_i \kappa(x^i,\cdot), \sum_i \gamma_i \kappa(x^i,\cdot)\rangle_F && (61)\\
&= \langle f, f\rangle_F, && (62)
\end{aligned}$$

the norm of $\mathcal{M} \in \mathcal{F}$ is equivalent to the norm of a corresponding $f \in F$. Hence, the regularised empirical risk minimisation problem in Eq. (57) is equivalent to

$$\inf_{f\in F} \gamma\|f\|_F^2 + \hat{\mathcal{R}}_L(f), \tag{63}$$

which minimises the regularised risk over the RKHS of the quantum kernel. We will see in the remaining sections that this allows us to characterise the problem of training and its solutions to a surprising degree.

### B. The measurements of optimal quantum models are expansions in the training data

The *representer theorem*, one of the main achievements of classical kernel theory, prescribes that the function $f$ from the RKHS which minimises the regularised empirical risk can always be expressed as a weighted sum of the kernel between $x$ and the training data. Together with the connection between quantum models and the RKHS of the quantum kernel, this fact will allow us to write optimal quantum machine learning models in terms of the quantum kernel.

More precisely, the representer theorem can be stated as follows (for a more general version, see [27], Theorem 5.1):

**Theorem 5** (Representer theorem)**.** *Let $\mathcal{X}, \mathcal{Y}$ be an input and output domain, $\kappa : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$ a kernel with a corresponding reproducing kernel Hilbert space $F$, and given training data $\mathcal{D} = \{(x^1, y^1), \ldots, (x^M, y^M) \in \mathcal{X}\times\mathcal{Y}\}$. Consider a strictly monotonic increasing regularisation function $g\colon [0,\infty) \to \mathbb{R}$, and an arbitrary loss $L\colon \mathcal{X}\times\mathcal{Y}\times\mathbb{R} \to \mathbb{R}\cup\{\infty\}$. Any minimiser of the regularised empirical risk*

$$f_{\mathrm{opt}} = \operatorname*{argmin}_{f\in F}\left\{\hat{\mathcal{R}}_L(f) + g\left(\|f\|_F\right)\right\}, \tag{64}$$

*admits a representation of the form:*

$$f_{\mathrm{opt}}(x) = \sum_{m=1}^{M} \alpha_m\, \kappa(x^m, x), \tag{65}$$

*where $\alpha_m \in \mathbb{R}$ for all $1 \le m \le M$.*

Note that the crucial difference to the form in Theorem (3) is that $m$ does not sum over arbitrary data from $\mathcal{X}$, but over a finite training data set. For us this means that the optimal quantum model can be written as

$$f_{\text{opt}}(x) = \sum_{m=1}^{M} \alpha_m \operatorname{tr}\left[\rho(x)\rho(x^m)\right] = \sum_{m=1}^{M} \alpha_m \left|\langle\phi(x)|\phi(x^m)\rangle\right|^2. \tag{66}$$

This in turn defines the measurements $\mathcal{M}$ of optimal quantum models.

**Theorem 6** (Optimal measurements)**.** *For the settings described in Theorem 5, the measurement that minimises the regularised empirical risk can be written as an expansion in the training data $x^m$, $m = 1 \dots M$,*

$$\mathcal{M}_{\text{opt}} = \sum_m \alpha_m \rho(x^m), \tag{67}$$

*with $\alpha_m \in \mathbb{R}$.*

*Proof.* This follows directly by noting that

$$\begin{aligned} f_{\text{opt}}(x) &= \sum_{m=1}^{M} \alpha_m \operatorname{tr}\left[\rho(x)\rho(x^m)\right] && (68)\\ &= \operatorname{tr}\left[\rho(x) \sum_{m=1}^{M} \alpha_m \rho(x^m)\right] && (69)\\ &= \operatorname{tr}\left[\rho(x)\mathcal{M}_{\text{opt}}\right] && (70) \end{aligned}$$

□

As mentioned in the summary and Figure 5, in variational circuits we typically only optimise over a subspace of the RKHS since the measurements $\mathcal{M}$ are constrained by a particular circuit ansatz. We can therefore not guarantee that the optimal measurement can be expressed by the variational ansatz. However, the above guarantees that there will always be a measurement of the form of Eq. (67) for which the quantum model has a lower regularised empirical risk than the best solution of the variational training.

As an example, we can use the apparatus of linear regression to show that the optimal measurement for a quantum model under least-squares loss can indeed be written as claimed in Eq. (67). For this I will assume once more that $\mathcal{X} = \mathbb{R}^N$ where $N = 2^n$ and $n$ is the number of qubits, and switch to bold notation. I will also use the (here much more intuitive) vectorised notation in which the quantum model $f(x) = \operatorname{tr}\left[\rho(x)\mathcal{M}\right]$ becomes $f(x) = \langle\!\langle \mathcal{M} \,|\rho(x)\,\rangle\!\rangle$, with the vectorised measurement $|\mathcal{M}\rangle\!\rangle = \sum_k \gamma_k \left|\rho(x^k)\right\rangle\!\rangle$.

A well-known result from linear regression states that the vector $\mathbf{w}$ that minimises the least-squares loss of a linear model $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ is given by

$$\mathbf{w} = (\mathbf{X}^\dagger\mathbf{X})^{-1}\mathbf{X}^\dagger\mathbf{y}, \tag{71}$$

if the inverse of $\mathbf{X}^\dagger\mathbf{X}$ exist. Here, $\mathbf{X}$ is the matrix that contains the data vectors as rows,

$$\mathbf{X} = \begin{pmatrix} x_1^1 & \cdots & x_N^1 \\ \vdots & \ddots & \vdots \\ x_1^M & \cdots & x_N^M \end{pmatrix}, \tag{72}$$

and $\mathbf{y}$ is an $M$-dimensional vector containing the target labels. A little trick exposes that $\mathbf{w}$ can be written as a linear combination of training inputs,

$$\mathbf{w} = \mathbf{X}^\dagger\left(\mathbf{X}(\mathbf{X}^\dagger\mathbf{X})^{-2}\mathbf{X}^\dagger\mathbf{y}\right) = \mathbf{X}^\dagger\boldsymbol{\alpha} = \sum_m \alpha_m \mathbf{x}^m, \tag{73}$$

where $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_M)$.

Since a quantum model is a linear model in feature space, we can associate the vectors in linear regression with the vectorised measurement and density matrix, and immediately derive

$$|\mathcal{M}\rangle\!\rangle = \sum_m y^m \left(\sum_{m'} \left|\rho(\mathbf{x}^{m'})\right\rangle\!\rangle\langle\!\langle\rho(\mathbf{x}^{m'})\right|\right)^{-1} |\rho(\mathbf{x}^m)\rangle\!\rangle\,, \tag{74}$$

by making use of the fact that in our notation

$$\mathbf{X}^\dagger\mathbf{X} \Longleftrightarrow \sum_m |\rho(\mathbf{x}^m)\rangle\!\rangle\langle\!\langle\rho(\mathbf{x}^m)|\,, \tag{75}$$

and

$$\mathbf{X}^\dagger\mathbf{y} \Longleftrightarrow \sum_m y^m\,|\rho(\mathbf{x}^m)\rangle\!\rangle\,. \tag{76}$$

Note that although this looks like an expansion in the feature states, the "coefficient" of $|\rho(\mathbf{x}^m)\rangle\!\rangle$ still contains an operator. However, with Eq. (73) and writing $\sum_m |\rho(\mathbf{x}^m)\rangle\!\rangle\langle\!\langle\rho(\mathbf{x}^m)|$ in its diagonal form,

$$\sum_m |\rho(\mathbf{x}^m)\rangle\!\rangle\langle\!\langle\rho(\mathbf{x}^m)| = \sum_k h_k\,|h_k\rangle\!\rangle\langle\!\langle h_k|\,, \tag{77}$$

we have

$$|\mathcal{M}\rangle\!\rangle = \sum_m \alpha_m\,|\rho(\mathbf{x}^m)\rangle\!\rangle\,, \tag{78}$$

with

$$\alpha_m = \sum_k h_k^{-2}\langle\!\langle h_k\,|\rho(\mathbf{x}^m)\,\rangle\!\rangle \sum_{m'} y^{m'}\left\langle\!\!\left\langle h_k\,\middle|\rho(\mathbf{x}^{m'})\,\right\rangle\!\!\right\rangle. \tag{79}$$

The optimal measurement in "matrix form" reads

$$\mathcal{M} = \sum_m \alpha_m\rho(\mathbf{x}^m) = \sum_m \alpha_m\,|\phi(\mathbf{x}^m)\rangle\langle\phi(\mathbf{x}^m)|\,, \tag{80}$$

as claimed by the representer theorem.

Of course, it may require a large routine to implement this measurement fully quantumly, since it involves inverting operators acting on the feature space. Alternatively one can compute the desired $\{\alpha_m\}$ classically and use the quantum computer to just measure the kernel. In the last section we will see ideas of how to use quantum algorithms to do the inversion, but these quantum training algorithms are complex enough to require fault-tolerant quantum computers which we do not have available today.

### C. The kernel defines which models are punished by regularisation

In statistical learning theory, the role of the regulariser in the regularised empirical risk minimisation problem is to "punish" some functions and favour others. Above, we specifically looked at regularisers of the form $\|f\|_F^2$, $f \in F$, which was shown to be equivalent to minimising the norm of the measurement (or the length of the vectorised measurement) in feature space. But what is it exactly that we are penalising here? It turns out that the kernel does not only fix the space of quantum models themselves, it also defines which functions are penalised in regularised empirical risk minimisation problems. This is beautifully described in [27] Section 4.3, and I will only give a quick overview here.

To understand regularisation, we need to have a closer look at the regularising term $\|f\|_F^2 = \langle f, f\rangle_F$. But with the construction of the RKHS it actually remains very opaque what this inner product actually computes. It turns out that for every RKHS $F$ there is a transformation $\Upsilon : F \to L_2(\mathcal{X})$ that maps functions in the RKHS to square integrable functions on $\mathcal{X}$. What we gain is a more intuitive inner product formed by an integral,

$$\langle f, f\rangle_F = \langle \Upsilon f, \Upsilon f\rangle_{L_2} = \int_{\mathcal{X}} (\Upsilon f(x))^2 dx. \tag{81}$$

The operator $\Upsilon$ can be understood as extracting the information from the model $f$ which gets integrated over in the usual $L_2$ norm, and hence penalised during optimisation. For example, for some kernels this can be shown to be the derivative of functions, and regularisation therefore provably penalise models with "large" higher-order derivatives – which means it favours smooth functions.

The important point is that every kernel defines a unique transformation $\Upsilon$, and therefore a unique kind of regularisation. This is summarised in Theorem 4.9 in [27], which I will reprint here without proof:

**Theorem 7** (RKHS and Regularization Operators)**.** *For every RKHS with reproducing kernel $\kappa$ there exists a corresponding regularization operator $\Upsilon : F \to D$ (where $D$ is an inner product space) such that for all $f \in F$,*

$$\langle \Upsilon\kappa(x,\cdot), \Upsilon f(\cdot)\rangle_D = f(x), \tag{82}$$

*and in particular*

$$\langle \Upsilon\kappa(x,\cdot), \Upsilon\kappa(x',\cdot)\rangle_D = \kappa(x,x'). \tag{83}$$

*Likewise, for every regularization operator $\Upsilon : F \to D$, where $F$ is some function space equipped with a dot product, there exists a corresponding RKHS $F$ with reproducing kernel $\kappa$ such that these two equations are satisfied.*

In short, the quantum kernel or data-encoding strategy does not only tell us about universality and optimal measurements, it also fixes the regularisation properties in empirical risk minimisation. Which data encoding actually leads to which regularisation property is still an interesting open question for research.

### D. Picking the best quantum model is a low-dimensional (convex) optimisation problem

Besides the representer theorem, a second main achievement of kernel theory is to recognise that optimising the empirical risk of convex loss functions over functions in an RKHS can be formulated as a finite-dimensional convex optimisation problem (or in less cryptic language, optimising over extremely large spaces is surprisingly easy when we use training data, something noted in [12] before).

The fact that the optimisation problem is finite-dimensional – and we will see the dimension is equal to the number of training data – is important, since the feature spaces in which the model classifies the data are usually very high-dimensional, and possibly even infinite-dimensional. This is obviously true for the data-encoding feature space of quantum computations as well – which is precisely why variational quantum machine learning parametrise circuits with a small number of trainable parameters instead of optimising over all unitaries/measurements. But even if we optimise over all quantum models, the results of this section guarantee that the dimensionality of the problem is limited by the size of the training data set.

The fact that optimisation is convex means that there is only one global minimum, and that we have a lot of tools to find it [29] - in particular, more tools than mere gradient descent. Convex optimisation problems can be roughly solved in time $\mathcal{O}(M^2)$ in the number of training data. Although prohibitive for large datasets, it makes the optimisation guaranteed to be tractable (and below we will see that quantum computers could in principle help to train with a runtime of $\mathcal{O}(M)$).

Let me make the statement more precise. Again, it follows from the fact that optimising over the RKHS of the quantum kernel is equivalent to optimising over the space of quantum models.

**Theorem 8** (Training quantum models can be formulated as a finite-dimensional convex program)**.** *Let $\mathcal{X}$ be a data domain and $\mathcal{Y}$ an output domain, $L : \mathcal{X} \times \mathcal{Y} \times \mathbb{R} \to [0,\infty)$ be a loss function, $F$ the RKHS of the quantum kernel over a non-empty convex set $\mathcal{X}$ with the reproducing kernel $\kappa$. Furthermore, let $\lambda \geq 0$ be a regularisation parameter and $D = \{(x^m, y^m), m = 1,\ldots,M\} \subset \mathcal{X} \times \mathcal{Y}$ a training data set. The regularised empirical risk minimisation problem is finite-dimensional, and if the loss is convex, it is also convex.*

*Proof.* Recall that according to the Representer Theorem 5, the solution to the regularised empirical risk minimisation problem

$$f_{\text{opt}} = \inf_{f\in F} \lambda \|f\|_F^2 + \hat{\mathcal{R}}_L(f) \tag{84}$$

has a representation of the form

$$f_{\text{opt}}(x) = \sum_m \alpha_m \text{tr}\left[\rho(x^m)\rho(x)\right]. \tag{85}$$

We can therefore write

$$\hat{\mathcal{R}}_L(f) = \frac{1}{M}\sum_m L(x^m, y^m, \sum_{m'} \alpha_{m'}\kappa(x^m, x^{m'})). \tag{86}$$

If the loss $L$ is convex, then this term is also convex, and it is $M$-dimensional since it only involves the $M$ degrees of freedom $\alpha_m$.

Now let us turn to the regularisation term and try to show the same. Consider

$$\|f\|_F^2 = \sum_{m,m'} \alpha_m \alpha_{m'} \mathrm{tr}\left[\rho(x^m)\rho(x^{m'})\right] = \sum_{m,m'} \alpha_m \alpha_{m'} \kappa(x^m, x^{m'}) = \boldsymbol{\alpha}^T \mathbf{K} \boldsymbol{\alpha}, \tag{87}$$

where $\mathbf{K} \in \mathbb{R}^{M\times M}$ is the kernel matrix or *Gram matrix* with entries $K_{m,m'} = \kappa(x^m, x^{m'})$, and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_M)$ is the vector of coefficients $\alpha_m$. Since $\mathbf{K}$ is by definition of the kernel positive definite, this term is also convex. Both $\boldsymbol{\alpha}$ and $\mathbf{K}$ are furthermore finite-dimensional.

Together, training a quantum model to find the optimal solution from Eq. (66) can be done by solving the optimisation problem

$$\inf_{\boldsymbol{\alpha}\in\mathbb{R}^M} \frac{1}{M} \sum_m L(x^m, y^m, \sum_{m'} \alpha_{m'} \kappa(x^m, x^{m'})) + \lambda \boldsymbol{\alpha}^T \mathbf{K} \boldsymbol{\alpha}, \tag{88}$$

which optimises over $M$ trainable parameters, and is convex for convex loss functions. □

A *support vector machine* is a special case of kernel-based training which uses a special convex loss function, namely the hinge loss, for $L$:

$$L(f(x), y) = \max(0, 1 - f(x)y), \tag{89}$$

where one assumes that $y \in \{-1, 1\}$. As derived in countless textbooks, the resulting optimisation problem can be constructed from geometric arguments as maximising the "soft" margin of the closest vectors to a decision boundary. Under this loss, Eq. (88) reduces to

$$\alpha_{\mathrm{opt}} = \max_\alpha \sum_m \alpha_m - \frac{1}{2} \sum_{m,m'} \alpha_m \alpha_{m'} y^m y^{m'} \kappa(x^m, x^{m'}). \tag{90}$$

Training a support vector machine with hinge loss and a quantum kernel $\kappa$ is equivalent to finding the general quantum model that minimises the hinge loss. The "quantum support vector machine" in [4, 5] is therefore not one of many ideas to build a hybrid classifier, it is a generic blueprint of how to train quantum models in a kernel-based manner.

## VII. SHOULD WE SWITCH TO KERNEL-BASED QUANTUM MACHINE LEARNING?

The fact that quantum models can be formulated as kernel methods with a quantum kernel raises an important question for current quantum machine learning research: how do kernel-based models, i.e., solutions to the problem in Eq. (88), compare to models whose measurements are trained variationally? Let us revisit Figure 5 in light of the results of the previous section.

We saw in Section VI D how kernel-based training optimises the measurement over a subspace spanned by $M$ encoded training inputs by finding the best coefficients $\alpha_m$, $m = 1 \ldots M$. We also saw in Section VI B that this subspace contains the globally optimal measurement. Variational training instead optimises over a subspace defined by the parametrised ansatz, which may or may not overlap with the training-data subspace, and could therefore not have access to the global optimum. The advantages of kernel-based training are therefore that we are guaranteed to find the globally optimal measurement over all possible quantum models. If the loss is convex, the optimisation problem is furthermore of a favourable structure that comes with a lot of guarantees about the performance and convergence of optimisation algorithms. But besides these great properties, in classical machine learning with big data, kernel methods were superseded by neural networks or approximate kernel methods [30] because of their poor scaling. Training involves computing the pair-wise distances between all training data in the Gram matrix of Eq. (88), which has at least a runtime of $\mathcal{O}(M^2)$ in the number of training samples $M$.[7] In contrast, training neural networks takes time $\mathcal{O}(M)$ that only depends linearly on the number of training samples. Can the training of variational quantum circuits offer a similar advantage over kernel-based training?

The answer is that it depends. So far, training variational circuits with gradient-based methods on hardware is based on so-called parameter-shift rules [31, 32] instead of backpropagation. This strategy introduces a linear scaling with the number of parameters $|\theta|$, and the number of circuits that need to be evaluated to train a variational

[7] Note that this is also true when using the trained model for predictions, where we need to compute the distance between a new input to any training input in feature space as shown in Eq. (66). However, in maximum margin classifiers, or support vector machines in the stricter sense, most $\alpha_m$ coefficients are zero, and only the distances to a few "support vectors" are needed.

quantum model therefore grows with $\mathcal{O}(|\theta|M)$. If the number of parameters in an application grows sufficiently slowly with the dataset size, variational circuits will almost be able to match the good scaling behaviour of neural networks, which is an important advantage over kernel-based training. But if, like in neural networks, the number of parameters in a variational ansatz grows linearly with the number of data, variational quantum models end up having the same quadratic scaling as the kernel-based approach regarding the number of circuits to evaluate. Practical experiments with $10-20$ parameters and about 100 data samples show that the constant overhead of gradient calculations on hardware make kernel-based training in fact much faster for small-scale applications.[8] In addition, there is no guarantee that the final measurement is optimal, we have high-dimensional non-convex training landscapes, and the additional burden of choosing a good variational ansatz. In conclusion, the kernel perspective is not only a powerful and theoretically appealing alternative to think about quantum machine learning, but may also speed up current quantum machine learning methods significantly.

As a beautiful example of the mutually beneficial relation of quantum computing and kernel methods, the story does not end here. While all of the above is based on models evaluated on a quantum computer but trained classically, convex optimisation problems happen to be exactly the kind of thing quantum computers are good at [33]. We can therefore ask whether quantum models could not in principle be *trained* by quantum algorithms. "In principle" alludes to the fact that such algorithms would likely be well beyond the reach of near-term devices, since training is a more complex affair that requires fully error-corrected quantum computers which we do not have yet.

The reasons why quantum training could help to lower this scaling are hidden in results from the early days of quantum machine learning, when quantum-based training was actively studied in the hope of finding exponential speedups for classical machine learning [6, 34, 35]. While these speedups only hold up under very strict assumptions of data loading oracles, they imply quadratic speedups for rather general settings (see also Appendix B). They can be summarised as follows: *given a feature map implemented by a fault-tolerant quantum computer, we can train kernel methods in time that grows linearly in the data.* If a kernel can be implemented as a quantum computation (like the Gaussian kernel [7]), this speedup would also hold for "classical models" – which are then merely run on a quantum computer.

Of course, fault-tolerant quantum computers may still take many years to develop and are likely to have a large constant overhead due to the expensive nature of quantum error correction. But in the longer term, this shows that the use of quantum computing is not only to implement interesting kernels. Quantum computers have the potential to become a game changer for kernel-based machine learning in a similar way to how GPU-accelerated hardware enabled deep learning.

## ACKNOWLEDGEMENTS

I want to thank Johannes Jakob Meyer, Nathan Killoran, Olivia di Matteo and Filippo Miatto, Nicolas Quesada and Ilya Sinayskiy for their time and helpful comments.

[1] P. Wittek, *Quantum machine learning: what quantum computing means to data mining* (Academic Press, 2014).
[2] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Quantum machine learning, Nature **549**, 195 (2017).
[3] M. Schuld and F. Petruccione, *Supervised learning with quantum computers* (Springer, 2018).
[4] M. Schuld and N. Killoran, Quantum machine learning in feature hilbert spaces, Physical Review Letters **122**, 040504 (2019).
[5] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, Supervised learning with quantum-enhanced feature spaces, Nature **567**, 209 (2019).
[6] P. Rebentrost, M. Mohseni, and S. Lloyd, Quantum support vector machine for big data classification, Physical Review Letters **113**, 130503 (2014).
[7] R. Chatterjee and T. Yu, Generalized coherent states, reproducing kernels, and quantum support vector machines, arXiv preprint arXiv:1612.03713 (2016).
[8] M. Schuld, A. Bocharov, K. Svore, and N. Wiebe, Circuit-centric quantum classifiers, arXiv preprint arXiv:1804.00633 (2018).
[9] J.-G. Liu and L. Wang, Differentiable learning of quantum circuit born machines, Physical Review A **98**, 062324 (2018).
[10] C. Blank, D. K. Park, J.-K. K. Rhee, and F. Petruccione, Quantum classifier with tailored quantum kernel, npj Quantum Information **6**, 1 (2020).

[8] See `https://pennylane.ai/qml/demos/tutorial_kernel_based_training.html`.

[11] Y. Liu, S. Arunachalam, and K. Temme, A rigorous and robust quantum speed-up in supervised machine learning, arXiv preprint arXiv:2010.02174 (2020).
[12] H.-Y. Huang, M. Broughton, M. Mohseni, R. Babbush, S. Boixo, H. Neven, and J. R. McClean, Power of data in quantum machine learning, arXiv preprint arXiv:2011.01938 (2020).
[13] J. M. Kübler, K. Muandet, and B. Schölkopf, Quantum mean embedding of probability distributions, Physical Review Research **1**, 033159 (2019).
[14] S. Lloyd, M. Schuld, A. Ijaz, J. Izaac, and N. Killoran, Quantum embeddings for machine learning, arXiv preprint arXiv:2001.03622 (2020).
[15] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, Parameterized quantum circuits as machine learning models, Quantum Science and Technology **4**, 043001 (2019).
[16] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, Barren plateaus in quantum neural network training landscapes, Nature communications **9**, 1 (2018).
[17] Z. Holmes, K. Sharma, M. Cerezo, and P. J. Coles, Connecting ansatz expressibility to gradient magnitudes and barren plateaus, arXiv preprint arXiv:2101.02138 (2021).
[18] M. Benedetti, D. Garcia-Pintos, O. Perdomo, V. Leyton-Ortega, Y. Nam, and A. Perdomo-Ortiz, A generative modeling approach for benchmarking and training shallow quantum circuits, npj Quantum Information **5**, 1 (2019).
[19] A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster, and J. I. Latorre, Data re-uploading for a universal quantum classifier, Quantum **4**, 226 (2020).
[20] I. Steinwart and A. Christmann, *Support vector machines* (Springer Science & Business Media, 2008).
[21] M. Wolf, Quantum channels and operations: Guided tour (2012).
[22] V. Jagadish and F. Petruccione, An invitation to quantum channels, arXiv preprint arXiv:1902.00909 (2019).
[23] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, M. S. Alam, S. Ahmed, J. M. Arrazola, C. Blank, A. Delgado, S. Jahangiri, *et al.*, Pennylane: Automatic differentiation of hybrid quantum-classical computations, arXiv preprint arXiv:1811.04968 (2018).
[24] R. Iten, R. Colbeck, I. Kukuljan, J. Home, and M. Christandl, Quantum circuits for isometries, Physical Review A **93**, 032318 (2016).
[25] J. G. Vidal and D. O. Theis, Input redundancy for parameterized quantum circuits, arXiv preprint arXiv:1901.11434 (2019).
[26] M. Schuld, R. Sweke, and J. J. Meyer, The effect of data encoding on the expressive power of variational quantum machine learning models, arXiv preprint arXiv:2008.08605 (2020).
[27] B. Schölkopf, A. J. Smola, F. Bach, *et al.*, *Learning with kernels: support vector machines, regularization, optimization, and beyond* (MIT press, 2002).
[28] S. Cheng, J. Chen, and L. Wang, Information perspective to probabilistic modeling: Boltzmann machines versus born machines, Entropy **20**, 583 (2018).
[29] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization* (Cambridge university press, 2004).
[30] A. Rahimi and B. Recht, Random features for large-scale kernel machines, Advances in neural information processing systems **20**, 1177 (2007).
[31] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, Quantum circuit learning, Physical Review A **98**, 032309 (2018).
[32] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, Evaluating analytic gradients on quantum hardware, Physical Review A **99**, 032331 (2019).
[33] A. W. Harrow, A. Hassidim, and S. Lloyd, Quantum algorithm for linear systems of equations, Physical Review Letters **103**, 150502 (2009).
[34] N. Wiebe, D. Braun, and S. Lloyd, Quantum algorithm for data fitting, Physical Review Letters **109**, 050505 (2012).
[35] S. Lloyd, M. Mohseni, and P. Rebentrost, Quantum principal component analysis, Nature Physics **10**, 631 (2014).

## Appendix A: Proof of Theorem 1

First, note that we are able to assume without loss of generality that the encoding generator $G$ is diagonal because one can diagonalise Hermitian operators as $G = Ve^{-ix_i\Sigma}V^\dagger$ with

$$e^{-ix_i\Sigma} = \begin{pmatrix} e^{-ix_i\lambda_1} & \vdots & 0 \\ 0 & \ddots & \\ 0 & \vdots & e^{-ix_i\lambda_d} \end{pmatrix} \tag{A1}$$

where $\{\lambda_1, \ldots, \lambda_d\}$ are the eigenvalues of $G$. Formally one can "absorb" $V, V^\dagger$ into the arbitrary circuits $W$ before and after the encoding gate. The remainder is just a matter of writing the matrix multiplications that represent the quantum circuit as a sum in the computational basis, and trying to introduce notation that hides irrelevant complexity:

$$\kappa(\mathbf{x},\mathbf{x}') = |\langle\phi(\mathbf{x}')|\phi(\mathbf{x})\rangle|^2 \tag{A2}$$

$$= \left|\langle 0|(W^{(1)})^\dagger(e^{-ix_1'\Sigma})^\dagger\cdots(e^{-ix_N'\Sigma})^\dagger \underbrace{(W^{(N+1)})^\dagger W^{(N+1)}}_{\mathbb{1}} e^{-ix_N\Sigma}\cdots e^{-ix_1\Sigma}W^{(1)}|0\rangle\right|^2 \tag{A3}$$

$$= \left|\langle 0|(W^{(1)})^\dagger(e^{-ix_1'\Sigma})^\dagger\cdots(e^{-ix_N'\Sigma})^\dagger e^{-ix_N\Sigma}\cdots e^{-ix_1\Sigma}W^{(1)}|0\rangle\right|^2 \tag{A4}$$

$$= \left|\sum_{j_1,\ldots,j_N=1}^{d}\sum_{k_1,\ldots,k_N=1}^{d} e^{-i(\lambda_{j_1}x_1-\lambda_{k_1}x_1'+\cdots+\lambda_{j_N}x_N-\lambda_{k_N}x_N')}\left(W^{(1)}_{1k_1}\ldots W^{(N)}_{k_{N-1}k_N}\right)^* W^{(N)}_{j_Nj_{N-1}}\ldots W^{(1)}_{j_11}\right|^2 \tag{A5}$$

$$= \left|\sum_{\mathbf{j}}\sum_{\mathbf{k}} e^{-i(\Lambda_\mathbf{j}\mathbf{x}-\Lambda_\mathbf{k}\mathbf{x}')}(w_\mathbf{k})^* w_\mathbf{j}\right|^2 \tag{A6}$$

$$= \sum_{\mathbf{j}}\sum_{\mathbf{k}}\sum_{\mathbf{h}}\sum_{\mathbf{l}} e^{-i(\Lambda_\mathbf{j}-\Lambda_\mathbf{l})\mathbf{x}}e^{i(\Lambda_\mathbf{k}-\Lambda_\mathbf{h})\mathbf{x}'}(w_\mathbf{k}w_\mathbf{h})^* w_\mathbf{j}w_\mathbf{l} \tag{A7}$$

Here, the scalars $W^{(i)}_{ab}$, $i=1,\ldots,N$, refer to the element $\langle a|W^{(i)}|b\rangle$ of the unitary operator $W^{(i)}$, the bold multi-index $\mathbf{j}$ summarises the set $(j_1,\ldots,j_N)$ where $j_i\in\{1,\ldots,d\}$ and $\Lambda_\mathbf{j}$ is a vector containing the eigenvalues selected by the multi-index (and similarly for $\mathbf{k},\mathbf{h},\mathbf{l}$).

We can now summarise all terms where $\Lambda_\mathbf{j}-\Lambda_\mathbf{l}=\mathbf{s}$ and $\Lambda_\mathbf{k}-\Lambda_\mathbf{h}=\mathbf{t}$, in other words where the differences of eigenvalues amount to the same vectors $\mathbf{s},\mathbf{t}$. Then

$$\kappa(\mathbf{x},\mathbf{x}') = \sum_{\mathbf{s},\mathbf{t}\in\Omega} e^{-i\mathbf{s}\mathbf{x}}e^{i\mathbf{t}\mathbf{x}'}\sum_{\mathbf{j},\mathbf{l}|\Lambda_\mathbf{j}-\Lambda_\mathbf{l}=\mathbf{s}}\ \sum_{\mathbf{k},\mathbf{h}|\Lambda_\mathbf{k}-\Lambda_\mathbf{h}=\mathbf{t}} w_\mathbf{j}w_\mathbf{l}(w_\mathbf{k}w_\mathbf{h})^* \tag{A8}$$

$$= \sum_{\mathbf{s},\mathbf{t}\in\Omega} e^{-i\mathbf{s}\mathbf{x}}e^{i\mathbf{t}\mathbf{x}'}c_\mathbf{st}. \tag{A9}$$

The frequency set $\Omega$ contains all vectors $\{\Lambda_\mathbf{j}-\Lambda_\mathbf{k}\}$ with $\Lambda_\mathbf{j}=(\lambda_{j_1},\ldots,\lambda_{j_N})$, $j_1,\ldots,j_N\in[1,\ldots,d]$. Let me illustrate this rather unwieldy notation with our standard example of encoding a real scalar input $x$ via a Pauli-X rotation.

**Example A.1.** *Consider the embedding from Example III.2. We have $W^{(1)}=W^{(2)}=\mathbb{1}$. With a singular value decomposition one can write the rotation operator as*

$$R_x(x) = e^{-ix\frac{1}{2}\sigma_x} = V^\dagger e^{-ix\frac{1}{2}\Sigma}V, \tag{A10}$$

*with*

$$V = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ -1 & 1\end{pmatrix}. \tag{A11}$$

*The unitary operators $V,V^\dagger$ can be absorbed into the general unitaries applied before and after the encoding, which sets $W^{(1)}=V^\dagger$ and $W^{(2)}=V$. The remaining $\frac{1}{2}\Sigma$ is a diagonal operator with eigenvalues $\{\lambda_1=-\frac{1}{2},\lambda_2=\frac{1}{2}\}$. We get*

$$\kappa(x,x') = \left|\sum_{j=1}^{2}\sum_{k=1}^{2}\sum_{i=1}^{2} e^{-i(\lambda_j x-\lambda_k x')}\,(V_{1k})^*(V_{ki})^*V_{ij}V_{j1}\right|^2. \tag{A12}$$

*Due to unitarity, inner products of different rows/columns of $V$, $V^\dagger$ are zero, and so $\sum_{i=1}^{2}(V_{ki})^*V_{ij}=\delta_{kj}$, leading to*

$$\kappa(x,x') = \left|\sum_{j=1}^{2} e^{-i\lambda_j(x-x')}\,(V_{1j})^*V_{j1}\right|^2 \tag{A13}$$

$$= \left|e^{-i\lambda_1(x-x')}\,(V_{11})^*V_{11}+e^{-i\lambda_2(x-x')}\,(V_{12})^*V_{21}\right|^2 \tag{A14}$$

$$= \left|\frac{1}{2}e^{i\frac{x-x'}{2}}+\frac{1}{2}e^{-i\frac{x-x'}{2}}\right|^2 \tag{A15}$$

$$= |\cos\left(\frac{x-x'}{2}\right)|^2 \tag{A16}$$

$$= \cos^2\left(\frac{x-x'}{2}\right). \tag{A17}$$

*This is the same result as in the "straight" computation from Eq. (38).*

### Appendix B: Convex optimisation with quantum computers

The family of quantum algorithms for convex optimisation in machine learning consists of many variations, but is altogether based on results that establish fast linear algebra processing routines for quantum computers. They are very technical in design, which is why they may not be easily accessible to many machine learning researchers (or in fact, for anyone who does not spend years of her life studying quantum computational complexity). This is why I will only summarise the results from a high-level perspective here.

- Given access to a quantum algorithm that encodes data into quantum states, we can prepare a mixed quantum state $\rho$ representing a $M \times M$ kernel Gram matrix in time $\mathcal{O}(MN)$, where $N$ is the size of the inputs $\mathbf{x} \in \mathbb{R}^N$ (see [6] or [3] Section 6.2.5),
- We can prepare a quantum state $|\mathbf{y}\rangle$ representing $M$ binary labels as amplitudes in time $\mathcal{O}(M)$ (see for example [24], or [3] Section 5.2.1).
- Given $|\mathbf{y}\rangle$, as well as $k \in \mathcal{O}(\epsilon^{-1})$ "copies" of $\rho(\mathbf{x})$ (meaning that we have to repeat the first step $k$ times), we can prepare $|\boldsymbol{\alpha}\rangle = \rho^{-1}(\mathbf{x})\,|\mathbf{y}\rangle$, a state whose amplitudes correspond to the coefficients $\boldsymbol{\alpha}$ in Theorem (8), to precision $\epsilon$ in time $\mathcal{O}(k \log d)$, where $d$ is the rank of $\rho$ (see [35], where this quantum algorithm was called "quantum principal component analysis", or [3] Section 5.4.3).
- We can estimate the amplitudes of $|\boldsymbol{\alpha}\rangle$ in time $\mathcal{O}(S/\tilde{\epsilon}^2)$ to precision $\tilde{\epsilon}$, where $S \leq M$ is the number of nonzero amplitudes (following from standard probability theory applied to quantum measurements, or [3] Section 5.1.3).

Overall, this is a recipe to compute the $S$ coefficients of the support vectors in time that is linear in the number of data points, a feat that is unlikely to be possible with a classical computer, at least not without imposing more structure on the problem, or allowing for heuristic results.