Chapter 6
Quantum Models as Kernel Methods

This chapter is an adapted version of the preprint article "Quantum machine learning models are kernel methods" by Maria Schuld [1].

We expose the important link between kernel methods, and quantum circuits used for supervised learning. We show that a large class of supervised quantum models are kernel methods with a "quantum kernel" which is fully defined by the data-encoding strategy of the circuit. This has far-reaching consequences, for example that such quantum models can be trained by minimising a relatively simple cost function, and that their generalisation performance is determined by the data-encoding strategy.

In the previous chapter, we saw that deterministic quantum models can be trained and used similarly to neural networks by optimising classical control parameters with gradient descent-type algorithms. However, we also remarked that quantum models are mathematically quite distinct from neural networks. In this chapter, we will see that the mathematical framework of quantum computing is instead strikingly similar to kernel methods that we introduced in Sect. 2.5.4. Both describe how information is processed by mapping it to vectors that live in potentially inaccessibly large Hilbert spaces, without the need of ever computing an explicit numerical representation of these vectors (see Fig. 6.1) [2, 3]. This analogy has sparked a range of studies in the past years, for example, to construct kernelised quantum machine learning models [4, 5], or to reveal links between quantum machine learning and maximum mean embeddings [6] as well as metric learning [7].

It turns out that the connection between quantum computing and kernel methods has far-reaching consequences: most supervised, deterministic quantum models can fundamentally be formulated as a classical kernel method whose kernel is computed by a quantum computer. The insight is based on the observation that quantum models are linear models in a certain feature space [3], and holds both for variational algorithms presented in the previous chapter, as well as for more sophisticated fault-tolerant algorithms which we will encounter in Chap. 7. It has been a crucial tool to investigate the separation between the computational complexity of quantum and

M. Schuld and F. Petruccione, *Machine Learning with Quantum Computers*,
Quantum Science and Technology, https://doi.org/10.1007/978-3-030-83098-4_6

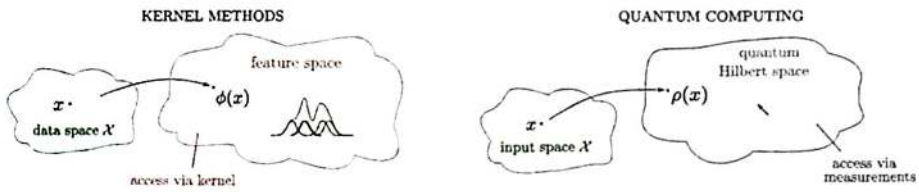

Fig. 6.1 Quantum computing and kernel methods are based on a similar principle. Both have mathematical frameworks in which information is mapped into and then processed in high-dimensional spaces to which we have only limited access. In kernel methods, the access to the feature space is facilitated through *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 through inner products of quantum states

classical machine learning [8, 9], and shows that the expressivity, optimisation and generalisation behaviour of quantum models is largely defined by the data-encoding strategy or embedding which fixes the kernel. Furthermore, this insight 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 strategy of training variational models, we do not have to worry about finding the right ansatz, or about how to avoid *barren plateaus* (see Sect. 5.3.3)—but pay the price of having to compute pairwise distances between embedded data points.

Since kernel theory is not very easy to understand, the next section will give an overview of the link between deterministic quantum models and kernel methods before jumping into more details. Wonderful textbooks on kernel methods are [10, 11], which serve as a basis for many of the following insights.

6.1 The Connection Between Quantum Models and Kernel Methods

First, a quick overview of the scope. We will consider deterministic quantum models as specified in Definition 5.1, and where the circuit consists of a data embedding and a variational part as in Eq. (5.3). However, it will be useful to consider the variational part and the measurement together as a variational measurement. Mathematically, this simply means we identify $\mathcal{M}'_\theta = W(\theta)^\dagger \mathcal{M} W(\theta)$ as the observable. Training a quantum model then means finding the measurement which minimises a cost function that depends on the training data. The important part of these assumptions is that the embedding is fixed and not trainable as, for example, proposed in [7, 12].

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 Fig. 6.2. Note that for this to hold, we need to define the data-encoding density matrices

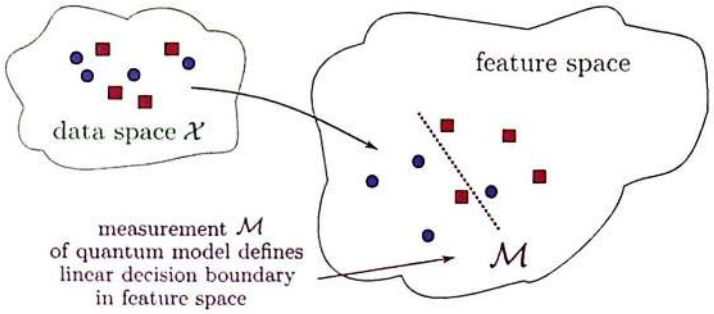

Fig. 6.2 Quantum models as linear models in a feature space. A quantum model can be understood as a model that maps data into a feature space in which the measurement defines a linear decision boundary. This feature space is not identical to the Hilbert space of the quantum system. Instead, we can define it as the space of complex matrices enriched with the Hilbert-Schmidt inner product

$\rho(x) = |\phi(x)\rangle\langle\phi(x)|$ as the feature "vectors"[1] instead of the Dirac vectors $|\phi(x)\rangle$ (which was the ϕ_2 version of the feature map introduced in Sect. 6.4.1). 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" $\rho(x)$. The (potentially trainable) measurement observable corresponds to the "weight vector".

As famously known from support vector machines [10], linear models in feature spaces can be efficiently evaluated and trained if we have access to inner products of feature vectors, which we saw in Sect. 2.5.4 is a function κ 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, $\kappa(x, x') = |\langle\phi(x')|\phi(x)\rangle|^2$ which we will call *quantum kernels*.

To understand what kernels can tell us about quantum machine learning, need another important concept from kernel theory: the *reproducing kernel Hilbe 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 $x \in \mathcal{X}$ from the input domain, as well as their linear combinations (for example, for the popular Gaussian kernel these linear combinations are sums of Gaussians centred 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

[1] 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 $\mathbb{R}^N$.

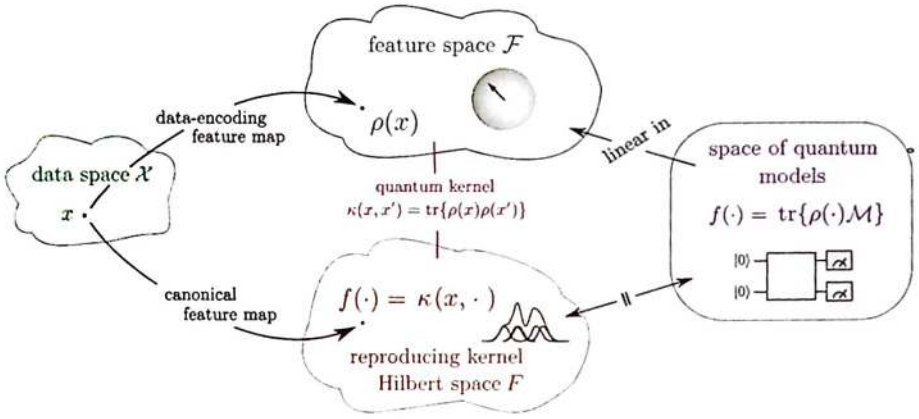

Fig. 6.3 Overview of the link between quantum models and kernel methods. The strategy with which data is encoded into quantum states is a feature map from the space of data to the feature space $\mathcal{F}$ of density matrices ρ. In this space, quantum models can be expressed as linear models whose 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 contain exactly the same functions (see Sect. 6.4.2). What we gain is an alternative representation of deterministic quantum models, one that only depends on the quantity $\mathrm{tr}\{\rho(x')\rho(x)\}$ (see Fig. 6.3).

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 the quantum kernel (see Sect. 6.5.1). 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

$$f_{\mathrm{opt}}(x) = \sum_{m=1}^{M} \alpha_m \mathrm{tr}\{\rho(x^m)\rho(x)\} = \mathrm{tr}\left\{\left(\sum_{m=1}^{M} \alpha_m \rho(x^m)\right)\rho(x)\right\}, \qquad (6.1)$$

where x^m, $m = 1, \ldots, M$ are the training data samples and $\alpha_m \in \mathbb{R}$ (see Sect. 6.5.2). 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 $M \ll 2^n$ degrees of freedom $\{\alpha_m\}$, where n is the number of qubits. 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 Sect. 6.5.3).

But how can we *find* the optimal model in Eq. (6.1)? We could simply train a variational ansatz, hoping that it learns the right measurement basis. But as illustrated in Fig. 6.4, 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 $M \ll 2^n$ degrees of freedom, *finding* the optimal measurement has the same favourable scaling (see Sect. 6.5.4) 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. (6.1) 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" [13] 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 [2, 3].

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

6.2 Quantum Computing, Feature Maps and Kernels

Let us start by laying the groundwork for the kernel perspective on quantum machine learning. First, we 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. We will then give some examples of data-encoding feature maps encountered in Chap. 4 and their quantum kernels, including a general description derived from the Fourier formalism in Sect. 5.2, which allows us to understand these kernels via trigonometric sums.

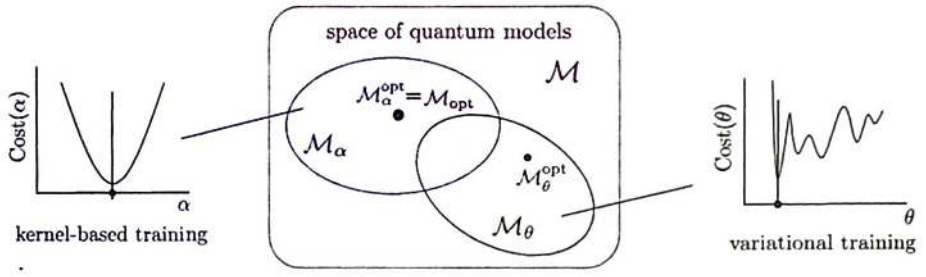

Fig. 6.4 Kernel-based training versus variational training. Training a quantum model as defined in this chapter tries to find the optimal measurement $\mathcal{M}_{\text{opt}}$ over all possible quantum measurements. Kernel theory guarantees that in most cases this optimal measurement will have a representation that is a linear combination in the training data with coefficients $\alpha = (\alpha_1, \ldots, \alpha_M)^T$. Kernel-based training, therefore, optimises over the parameters α 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 θ-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

6.2.1 *Data Encoding as a Feature Map*

Consider a data embedding circuit $S(x)$ that depends on data $x \in \mathcal{X}$ from some domain $\mathcal{X}$ to prepare a quantum state $S(x)|0\rangle = |\phi(x)\rangle$. While from a quantum physics perspective it seems natural—and has been done in Sect. 4.5—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, which means that the apparatus of kernel theory does not apply elegantly. Instead, we will solely work with $x \to \rho(x) = |\phi(x)\rangle\langle\phi(x)|$ as the "quantum feature map" and, to avoid confusion, call it the *data-encoding feature map*.

Definition 6.1 *(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 $\langle \rho, \sigma \rangle_{\mathcal{F}} = \text{tr}\{\rho^\dagger \sigma\}$ for $\rho, \sigma \in \mathcal{F}$. The data-encoding feature map is defined as the transformation

$$\phi : \mathcal{X} \to \mathcal{F}, \qquad \phi(x) = |\phi(x)\rangle\langle\phi(x)| = \rho(x), \tag{6.2}$$

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

This definition makes sure that the feature space is a Hilbert space, and it allows measurements to live in the same space [14], which we will need to define linear

models in $\mathcal{F}$. Section 6.2.3 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 [9]. Finally, note that while we limit our scope to pure quantum states here, the data-encoding feature map can easily be extended to mixed states.

6.2.2 *Quantum Kernels*

Remember that kernels were 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 $\mathbb{R}$ (see Sect. 2.5.4). 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 via the inner product in the corresponding feature space. 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 6.1 we can immediately write down the kernel induced by the data-encoding feature map, which we will call the *quantum kernel*.

Definition 6.2 *(Quantum kernel)* Let ϕ 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 $x, x' \in \mathcal{X}$,

$$\kappa(x, x') = \mathrm{tr}\{\rho(x')\rho(x)\} = |\langle \phi(x') | \phi(x) \rangle|^2. \tag{6.3}$$

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 $x^m \in \mathcal{X}$, $m = 1 \ldots M$, and for any $c_m \in \mathbb{C}$, we have

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

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

6.2.3 *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. We, therefore, want to 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

$$\phi_v : \mathcal{X} \to \mathcal{F}_v \subset \mathcal{H} \otimes \mathcal{H}^*, \tag{6.4}$$

$$\phi_v(x) = |\phi(x)\rangle \otimes |\phi^*(x)\rangle, \tag{6.5}$$

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 we 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

$$\langle\psi, \varphi\rangle_{\mathcal{F}_v} = |\langle\psi|\varphi\rangle|^2, \tag{6.6}$$

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 $\mathbb{C}^{4^n}$.

Vectorised density matrices are common in the theory of open quantum systems [15], where they are written as $|\rho\rangle\!\rangle$ (see also the *Choi-Jamiolkowski isomor-*

phism). We will adopt this notation below to replace the Hilbert-Schmidt inner product $\mathrm{tr}\{\rho^\dagger\sigma\}$ with $\langle\!\langle \rho \,|\sigma \rangle\!\rangle$, which can be more illustrative at times. Note that the vectorised representation of the data-encoding feature map cannot capture mixed quantum states and is, therefore, less powerful.

6.3 Examples of Quantum Kernels

To fill the definition of the quantum kernel with life, let us revisit information encoding strategies and define the data-encoding feature map, as well as the kernels they give rise to (see also [2]; a summary is presented in Table 6.1). It has been shown that there are kernels that cannot be efficiently computed on classical computers [8], which we will explore in more detail in Sect. 9 on quantum advantages.

6.3.1 Quantum Kernels Derived from Data Encoding

The following strategies to encode data have a resemblance to kernels that can be found in 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 Fig. 6.5 using simulations of quantum computations implemented in the quantum machine learning software library PennyLane [16]. Note that, as usual, we switch to bold notation when the input space is $\mathbb{C}^N$ or $\mathbb{R}^N$.

Basis encoding. The data-encoding feature map of basis encoding maps a binary string to a computational basis state,

$$\phi : x \to |i_x\rangle\langle i_x|. \tag{6.7}$$

Table 6.1 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 $\mathcal{X} \subseteq \mathbb{R}^N$

Encoding	Kernel $\kappa(x, x')$
Basis encoding	$\delta_{x,x}$
Amplitude encoding	$\lvert\mathbf{x}^\dagger\mathbf{x}'\rvert^2$
Repeated amplitude encoding	$(\lvert\mathbf{x}^\dagger\mathbf{x}'\rvert^2)^r$
Angle 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 time-evolution encoding	$\sum_{s,t\in\Omega} e^{i\mathbf{s}\mathbf{x}} e^{i\mathbf{t}\mathbf{x}'} c_{\mathbf{s},\mathbf{t}}$

The quantum kernel is given by the Kronecker delta

$$\kappa(x, x') = |\langle i_{x'} | j_x \rangle|^2 = \delta_{x,x'}, \tag{6.8}$$

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.

Amplitude encoding. The data-encoding feature map of amplitude encoding associates each input with a quantum state whose amplitudes in the computational basis are the elements in the input vector

$$\phi : \mathbf{x} \to |\mathbf{x}\rangle\langle\mathbf{x}| = \sum_{i,j=1}^{N} x_i x_j^* |i\rangle\langle j|. \tag{6.9}$$

The quantum kernel is the absolute square of the linear kernel

$$\kappa(\mathbf{x}, \mathbf{x}') = |\langle \mathbf{x}' | \mathbf{x} \rangle|^2 = |\mathbf{x}^\dagger \mathbf{x}'|^2. \tag{6.10}$$

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.

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

$$\phi : \mathbf{x} \to |\mathbf{x}\rangle\langle\mathbf{x}| \otimes \cdots \otimes |\mathbf{x}\rangle\langle\mathbf{x}| \tag{6.11}$$

to get powers of the quantum kernel above,

$$\kappa(\mathbf{x}, \mathbf{x}') = (|\langle \mathbf{x}' | \mathbf{x} \rangle|^2)^r = (|(\mathbf{x}')^\dagger \mathbf{x}|^2)^r. \tag{6.12}$$

A constant non-homogenity can be added by extending the original input with constant dummy features.

Rotation encoding. The data-encoding feature map of this time-evolution encoding executed by Pauli rotations is given by

$$\phi : \mathbf{x} \to |\phi(\mathbf{x})\rangle\langle\phi(\mathbf{x})| \tag{6.13}$$

with, if we use Pauli-Y rotations,

$$|\phi(\mathbf{x})\rangle = \sum_{q_1,\ldots,q_n=0}^{1} \prod_{k=1}^{n} \cos(x_k)^{q_k} \sin(x_k)^{1-q_k} |q_1, \ldots, q_n\rangle, \tag{6.14}$$

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

$$\kappa(\mathbf{x}, \mathbf{x}') = \prod_{k=1}^{n} |\sin x_k \sin x_k' + \cos x_k \cos x_k'|^2 = \prod_{k=1}^{n} |\cos(x_k - x_k')|^2. \quad (6.15)$$

Coherent state encoding. Finally, we want to add a kernel from a slightly different computational model, which implements the ubiquitous Gaussian kernel. 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

$$|\alpha\rangle = e^{-\frac{|\alpha|^2}{2}} \sum_{k=0}^{\infty} \frac{\alpha^k}{\sqrt{k!}} |k\rangle, \quad (6.16)$$

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

$$\phi : x_i \to |\alpha_{x_i}\rangle\langle\alpha_{x_i}|, \text{ with } |\alpha_{x_i}\rangle = e^{-\frac{|x_i|^2}{2}} \sum_{k=0}^{\infty} \frac{x_i^k}{\sqrt{k!}} |k\rangle. \quad (6.17)$$

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

$$|\alpha_{\mathbf{x}}\rangle\langle\alpha_{\mathbf{x}}| = |\alpha_{x_1}\rangle\langle\alpha_{x_1}| \otimes \cdots \otimes |\alpha_{x_n}\rangle\langle\alpha_{x_n}|. \quad (6.18$$

The quantum kernel is a Gaussian kernel [17]:

$$\kappa(\mathbf{x}, \mathbf{x}') = \left| e^{-\left(\frac{|\mathbf{x}|^2}{2} + \frac{|\mathbf{x}'|^2}{2} - \mathbf{x}^T\mathbf{x}'\right)} \right|^2 = e^{-|\mathbf{x}-\mathbf{x}'|^2} \quad (6.19)$$

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

6.3.2 *Fourier Representation of Quantum Kernels*

The reason that all embeddings plotted in Fig. 6.5 have a periodic, trigonometric structure lies in an effect we have seen before in Sect. 5.2, where we showed that a quantum model can be written as a Fourier-type sum, which is a linear combination of trigonometric functions. Here we will use the same fundamental calculation to define a general class of embeddings that is used a lot in near-term quantum machine learn-

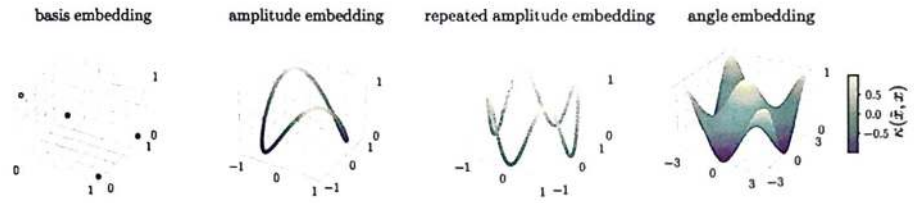

Fig. 6.5 Quantum kernels of different data embeddings. Plots of some of the functions $\kappa(\tilde{\mathbf{x}}, \mathbf{x})$ for the kernels introduced above, using $\mathbf{x} = (x_1, x_2) \in \mathbb{R}^2$ for illustration purposes. The first entry $\tilde{\mathbf{x}}$ is fixed at $\tilde{\mathbf{x}} = (0, 0)$ for basis and angle 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

ing, 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 we will stick to the bold notation. As before, the embedding of x_i is executed by time-evolution encoding, which uses 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 $\sigma \in \{\sigma_z, \sigma_y, \sigma_z\}$ (see also Fig. 6.6). The gates can be applied to different qubits as in angle encoding, or to the same qubits, and to be general we allow for arbitrary quantum computations between each encoding gate. For simplicity, we will assume 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 works pretty much as sketched in Sect. 5.2.

Theorem 6.1 (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 $\mathbf{x} = (x_1, \ldots, x_N)^T \in \mathcal{X}$ into an 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*

$$S(\mathbf{x}) = W^{(N+1)}e^{-ix_NG}W^{(N)} \ldots W^{(2)}e^{-ix_1G}W^{(1)}. \tag{6.20}$$

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

$$\kappa(\mathbf{x}, \mathbf{x}') = \sum_{\mathbf{s},\mathbf{t}\in\Omega} e^{i\mathbf{s}\mathbf{x}}e^{-i\mathbf{t}\mathbf{x}'}c_{\mathbf{st}}, \tag{6.21}$$

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.

For the important class of Pauli generators, the kernel becomes a Fourier series.

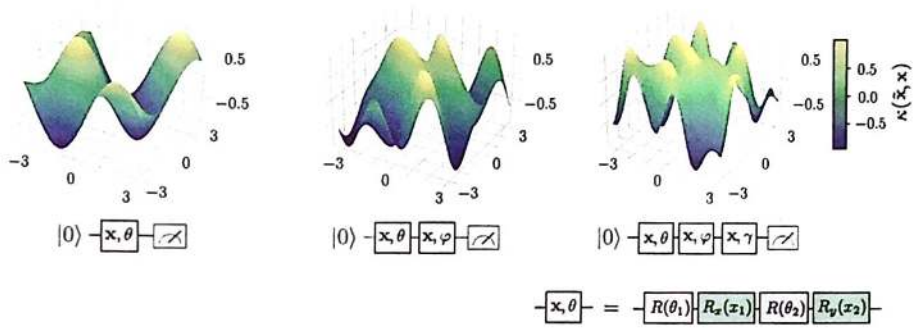

Fig. 6.6 Kernels generated by rotation embeddings. Plots of the quantum kernel $\kappa(\tilde{\mathbf{x}}, \mathbf{x})$ with $\tilde{\mathbf{x}} = (0, 0)$ using a very general data-encoding strategy that repeats the input encoding into a single-qubit one, two and three times. It is obvious that the repetition decreases the smoothness of the kernel by increasing the Fourier basis functions from which the kernel is inherently constructed

Corollary 6.1 (Fourier series representation of the quantum kernel) *For the setting described in Theorem 6.1, 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 Ω becomes the set of N-dimensional integer-valued vectors $\mathbf{n} = (n_1, \ldots, n_N)^T$, $n_1, \ldots n_N \in \mathbb{Z}$. In this case, the quantum kernel is a multi-dimensional Fourier series*

$$\kappa(\mathbf{x}, \mathbf{x}') = \sum_{\mathbf{n},\mathbf{n}' \in \Omega} e^{i\mathbf{n}\mathbf{x}} e^{-i\mathbf{n}'\mathbf{x}'} c_{\mathbf{n},\mathbf{n}'}, \tag{6.22}$$

Expressions (6.21) and (6.22) 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

$$\kappa(\mathbf{x}, \mathbf{x}') = \sum_{\mathbf{s} \in \Omega} e^{i\mathbf{s}(\mathbf{x}-\mathbf{x}')} c_{\mathbf{s}}. \tag{6.23}$$

Since $e^{-ix_i G} e^{ix_i' G} = e^{-i(x_i - x_i')G}$, this is true for all data embeddings that encode each original input into a separate physical subsystem, like angle 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.

6.4 · The RKHS of Quantum Kernels

We 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.

6.4.1 *Quantum Models as Linear Models*

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

Definition 6.3 *(Linear model)* Let $\mathcal{X}$ be a data domain and $\phi : \mathcal{X} \to \mathcal{F}$ a feature map. We call any function

$$f(x) = \langle \phi(x), w \rangle_{\mathcal{F}}, \tag{6.24}$$

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

From this definition, we immediately see that deterministic quantum models are linear models. In the following, we will refer to these models as f rather than f_θ to emphasise that we do not need trainable parameters for the statements in this section to hold.

Theorem 6.2 (Deterministic quantum models are linear models in data-encoding feature space) *Let $f(x) = tr\{\rho\mathcal{M}\}$ 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 6.3 (Quantum measurements are linear combinations of data-encoding states) *Let $f_{\mathcal{M}}(x) = tr\{\rho\mathcal{M}\}$ be a quantum model. There exists a measurement $\mathcal{M}_{\text{exp}} \in \mathcal{F}$ of the form*

$$\mathcal{M}_{\text{exp}} = \sum_k \gamma_k \rho(x^k) \tag{6.25}$$

with $x^k \in \mathcal{X}$, such that $f_{\mathcal{M}}(x) = f_{\mathcal{M}_{\text{exp}}}(x)$ 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

$$\mathcal{M} = \mathcal{M}_{\text{exp}} + R. \tag{6.26}$$

Since the trace is linear, we have

$$\mathrm{tr}\{\rho(x)\mathcal{M}\} = \mathrm{tr}\{\rho(x)\mathcal{M}_{\mathrm{exp}}\} + \mathrm{tr}\{\rho(x)R\}. \tag{6.27}$$

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. If the measurement depends on trainable parameters (via a variational circuit applied before the measurement), the model is also not, in general, linear in the 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 "measurement vector" $\sum_i \mu_i |\mu_i\rangle \otimes |\mu_i\rangle \in \mathcal{F}_v$.

$$\begin{aligned} f(x) &= \langle\phi(x)|\mathcal{M}|\phi(x)\rangle && (6.28)\\ &= \sum_i \mu_i |\langle\mu_i|\phi(x)\rangle|^2 && (6.29)\\ &= \Big(\langle\phi(x)| \otimes \langle\phi^*(x)|\Big)\Big(\sum_i \mu_i |\mu_i\rangle \otimes |\mu_i^*\rangle\Big), && (6.30) \end{aligned}$$

or using the vectorised density matrix notation introduced above

$$f(x) = \langle\!\langle \rho(x) \,|w\rangle\!\rangle, \tag{6.31}$$

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

6.4.2 *Describing the RKHS*

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. We 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[2]):

Definition 6.4 *(Reproducing kernel Hilbert space)* Let $\mathcal{X} \neq \emptyset$. The reproducing kernel Hilbert space of a kernel κ 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

$$\langle f, g \rangle_F = \sum_{ij} \alpha_i \beta_j \kappa(x^i, x^j), \tag{6.32}$$

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

Note that according to Theorem 6.1 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 κ as the quantum kernel, the definition of the inner product reveals with

$$\langle \kappa(x, \cdot), \kappa(x', \cdot) \rangle_F = \kappa(x, x'), \tag{6.33}$$

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*

$$\langle f, \kappa(x, \cdot) \rangle_F = f(x) \text{ for all } f \in F, \tag{6.34}$$

which also shows that the kernel is the evaluation functional δ_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 Fig. 6.7). 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 Figs. 6.5 and 6.6. By feeding another data point x' into this distance 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 centred 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

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

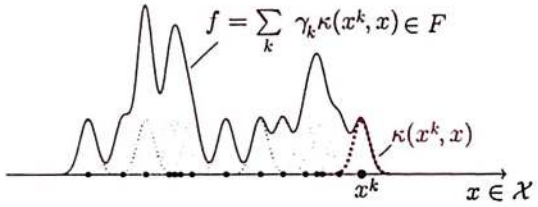

Fig. 6.7 Intuition for the functions living in the reproducing kernel Hilbert space (RKHS). The RKHS F contains functions that are linear combinations of kernel functions where one "slot" is fixed in a possible data sample $x^k \in \mathcal{X}$. This illustration of one such function $f \in F$, using 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 and therefore learn

Theorem 6.4 *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

$$f(x) = \sum_k \gamma_k \kappa(x^k, x) \tag{6.35}$$

$$= \sum_k \gamma_k \mathrm{tr}\{\rho(x^k)\rho(x)\} \tag{6.36}$$

$$= \mathrm{tr}\{\sum_k \gamma_k \rho(x^k)\rho(x)\} \tag{6.37}$$

$$= \mathrm{tr}\{\mathcal{M}\rho(x)\}. \tag{6.38}$$

Using Theorem 6.3, 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 [10]).

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 [10] Definition 4.52)

Definition 6.5 *(Universal kernel)* A continuous kernel κ on a compact metric space $\mathcal{X}$ is called universal if the RKHS F of κ 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

$$\|f - g\|_\infty \leq \epsilon. \tag{6.39}$$

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 [10] 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 σ a Pauli matrix) to encode data: since such rotations are 2π-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. Other examples for universal kernels are kernels of the form $\kappa(x, x') = \sum_{k=1}^{\infty} c_k \langle x', x\rangle^k$ (see Corollary 4.57 in [10]). Vice versa, the universality proof for a type of quantum model in [18] suggests that some quantum kernels of the form (6.1) are universal in the asymptotic limit of exponentially large circuits.

We 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 [2]). 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 δ_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 [19, 20].

6.5 Kernel-Based Training

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.

6.5.1 Training as Optimising Over the RKHS

As we saw in Sect. 2.3.1, in machine learning we want to find optimal models, or those that minimise the cost functions derived from learning problems. We also introduced training as the process of solving an *empirical risk minimisation problem*. A slightly more general form called *regularised* empirical risk minimisation for deterministic quantum models can be cast as follows:

Definition 6.6 *(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

$$\mathcal{R}_L(f) = \int_{\mathcal{X}\times\mathcal{Y}} L(x, y, f(x)) \, \mathrm{d}p(x, y) \tag{6.40}$$

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

$$\hat{\mathcal{R}}_L(f) = \frac{1}{M} \sum_{m=1}^{M} L(x^m, y, f(x^m)). \tag{6.41}$$

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 $\mathcal{M}$,

$$\inf_{\mathcal{M}\in\mathcal{F}} \lambda \|\mathcal{M}\|_{\mathcal{F}}^2 + \hat{\mathcal{R}}_L(\mathrm{tr}\{\rho(x)\mathcal{M}\}), \tag{6.42}$$

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

We saw in Sect. 6.4 that quantum models are equivalent to functions in the RKHS of the quantum kernel, which allows us to replace the term $\hat{\mathcal{R}}_L(\mathrm{tr}\{\rho(x)\mathcal{M}\})$ in the empirical risk by $\hat{\mathcal{R}}_L(f)$, $f \in F$.

But what about the regularisation term? Since with Theorem 6.3, we can write

$$\|\mathcal{M}\|_{\mathcal{F}}^2 = \mathrm{tr}\{\mathcal{M}^2\} \tag{6.43}$$

$$= \sum_{ij} \gamma_i \gamma_j \mathrm{tr}\{\rho(x^i)\rho(x^j)\} \tag{6.44}$$

$$= \sum_{ij} \gamma_i \gamma_j \kappa(x^i, x^j) \tag{6.45}$$

$$= \langle \sum_i \gamma_i \kappa(x^i, \cdot), \sum_i \gamma_i \kappa(x^i, \cdot) \rangle_F \tag{6.46}$$

$$= \langle f, f \rangle_F, \tag{6.47}$$

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. (6.42) is equivalent to

$$\inf_{f \in F} \gamma \|f\|_F^2 + \hat{\mathcal{R}}_L(f), \tag{6.48}$$

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.

6.5.2 Optimal Measurements and the Representer Theorem

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 [11], Theorem 5.1):

Theorem 6.5 (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*

$$f_{\mathrm{opt}} = \operatorname*{argmin}_{f \in F} \left\{ \hat{\mathcal{R}}_L(f) + g\left(\|f\|_F\right) \right\}, \tag{6.49}$$

admits a representation of the form

$$f_{\text{opt}}(x) = \sum_{m=1}^{M} \alpha_m \, \kappa(x^m, x), \tag{6.50}$$

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

Note that the crucial difference to the form in Theorem (6.3) 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

$$f_{\text{opt}}(x) = \sum_{m=1}^{M} \alpha_m \, \text{tr}\{\rho(x)\rho(x^m)\} = \sum_{m=1}^{M} \alpha_m \, |\langle\phi(x)|\phi(x^m)\rangle|^2. \tag{6.51}$$

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

Theorem 6.6 (Optimal measurements) *For the settings described in Theorem 6.5, the measurement that minimises the regularised empirical risk can be written as an expansion in the training data* x^m, $m = 1 \ldots M$

$$\mathcal{M}_{\text{opt}} = \sum_{m} \alpha_m \rho(x^m), \tag{6.52}$$

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

Proof This follows directly by noting that

$$f_{\text{opt}}(x) = \sum_{m=1}^{M} \alpha_m \, \text{tr}\{\rho(x)\rho(x^m)\} \tag{6.53}$$

$$= \text{tr}\{\rho(x) \sum_{m=1}^{M} \alpha_m \rho(x^m)\} \tag{6.54}$$

$$= \text{tr}\{\rho(x)\mathcal{M}_{\text{opt}}\} \tag{6.55}$$

As shown in Fig. 6.4, 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. (6.52) 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. (6.52). For this, we 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. We will also use the (here much more intuitive) vectorised notation in which the quantum model $f(x) = \text{tr}\{\rho(x)\mathcal{M}\}$ becomes $f(x) = \langle\!\langle \mathcal{M} \, | \rho(x) \rangle\!\rangle$, with the vectorised measurement $|\mathcal{M}\rangle\!\rangle = \sum_k \gamma_k |\rho(x^k)\rangle\!\rangle$.

We saw in Sect. 2.5.1 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

$$\mathbf{w} = (\mathbf{X}^\dagger\mathbf{X})^{-1}\mathbf{X}^\dagger\mathbf{y}, \tag{6.56}$$

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

$$\mathbf{X} = \begin{pmatrix} x_1^1 & \dots & x_N^1 \\ \vdots & \ddots & \vdots \\ x_1^M & \dots & x_N^M \end{pmatrix}, \tag{6.57}$$

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

$$\mathbf{w} = \mathbf{X}^\dagger\left(\mathbf{X}(\mathbf{X}^\dagger\mathbf{X})^{-2}\mathbf{X}^\dagger\mathbf{y}\right) = \mathbf{X}^\dagger\alpha = \sum_m \alpha_m \mathbf{x}^m, \tag{6.58}$$

where $\alpha = (\alpha_1, \dots, \alpha_M)^T$.

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

$$|\mathcal{M}\rangle\!\rangle = \sum_m y^m \left(\sum_{m'} |\rho(\mathbf{x}^{m'})\rangle\!\rangle\langle\!\langle\rho(\mathbf{x}^{m'})|\right)^{-1} |\rho(\mathbf{x}^m)\rangle\!\rangle, \tag{6.59}$$

by making use of the fact that in our notation

$$\mathbf{X}^\dagger\mathbf{X} \Longleftrightarrow \sum_m |\rho(\mathbf{x}^m)\rangle\!\rangle\langle\!\langle\rho(\mathbf{x}^m)|, \tag{6.60}$$

and

$$\mathbf{X}^\dagger\mathbf{y} \Longleftrightarrow \sum_m y^m |\rho(\mathbf{x}^m)\rangle\!\rangle. \tag{6.61}$$

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. (6.58) and writing $\sum_m |\rho(\mathbf{x}^m)\rangle\!\rangle\langle\!\langle\rho(\mathbf{x}^m)|$ in its diagonal form

$$\sum_m |\rho(\mathbf{x}^m)\rangle\!\rangle\langle\!\langle\rho(\mathbf{x}^m)| = \sum_k h_k |h_k\rangle\!\rangle\langle\!\langle h_k|, \tag{6.62}$$

we have

$$|\mathcal{M}\rangle\!\rangle = \sum_m \alpha_m |\rho(\mathbf{x}^m)\rangle\!\rangle, \tag{6.63}$$

with

$$\alpha_m = \sum_k h_k^{-2} \langle\!\langle h_k \,|\rho(\mathbf{x}^m)\rangle\!\rangle \sum_{m'} y^{m'} \langle\!\langle h_k \,|\rho(\mathbf{x}^{m'})\rangle\!\rangle. \tag{6.64}$$

The optimal measurement in "matrix form" reads

$$\mathcal{M} = \sum_m \alpha^m \rho(\mathbf{x}^m) = \sum_m \alpha^m |\phi(\mathbf{x}^m)\rangle\langle\phi(\mathbf{x}^m)|, \tag{6.65}$$

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.

6.5.3 *The Impact of the Kernel on 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, but also defines which functions are penalised in regularised empirical risk minimisation problems. This is beautifully described in [11] Sect. 4.3, and we 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 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

$$\langle f, f\rangle_F = \langle \Upsilon f, \Upsilon f\rangle_{L_2} = \int_{\mathcal{X}} (\Upsilon f(x))^2 dx. \tag{6.66}$$

The operator Υ 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 Υ, and therefore, a unique kind of regularisation. This is summarised in Theorem 4.9 in [11], which we will reprint here without proof.

Theorem 6.7 (RKHS and Regularisation Operators) *For every RKHS with reproducing kernel κ, there exists a corresponding regularisation operator $\Upsilon : F \to D$ (where D is an inner product space) such that for all $f \in F$*

$$\langle \Upsilon\kappa(x, \cdot), \Upsilon f(\cdot)\rangle_D = f(x), \tag{6.67}$$

and in particular

$$\langle \Upsilon\kappa(x, \cdot), \Upsilon\kappa(x', \cdot)\rangle_D = \kappa(x, x'). \tag{6.68}$$

Likewise, for every regularisation 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 κ 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.

6.5.4 Kernel-Based Learning Is Suprisingly Simple

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 [9] 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 [21]—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 us 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 6.8 (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 κ. 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 6.5, the solution to the regularised empirical risk minimisation problem

$$f_{\text{opt}} = \inf_{f \in F} \lambda \| f \|_F^2 + \hat{\mathcal{R}}_L(f) \tag{6.69}$$

has a representation of the form

$$f_{\text{opt}}(x) = \sum_m \alpha_m \text{tr}\{\rho(x^m)\rho(x)\}. \tag{6.70}$$

We can therefore write

$$\hat{\mathcal{R}}_L(f) = \frac{1}{M} \sum_m L(x^m, y^m, \sum_{m'} \alpha_{m'} \kappa(x^m, x^{m'})). \tag{6.71}$$

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 α_m.

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

$$\| f \|_F^2 = \sum_{m,m'} \alpha_m \alpha_{m'} \text{tr}\{\rho(x^m)\rho(x^{m'})\} = \sum_{m,m'} \alpha_m \alpha_{m'} \kappa(x^m, x^{m'}) = \boldsymbol{\alpha}^T \mathbf{K} \boldsymbol{\alpha}, \tag{6.72}$$

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 α_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. (6.51) can be done by solving the optimisation problem

$$\inf_{\boldsymbol{\alpha} \in \mathbb{R}^M} \frac{1}{M} \sum_m L(x^m, y^m, \sum_{m'} \alpha_{m'} \kappa(x^m, x^{m'})) + \lambda \boldsymbol{\alpha}^T \mathbf{K} \boldsymbol{\alpha}, \tag{6.73}$$

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

The *support vector machine* can be constructed as a special case of kernel-based training which uses a special convex loss function, namely the hinge loss, for L

$$L(f(x), y) = \max(0, 1 - f(x)y), \tag{6.74}$$

where one assumes that $y \in \{-1, 1\}$. As derived in Sect. 2.5.4.3, 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. (6.73) reduces to

$$\alpha_{\text{opt}} = \max_{\alpha} \sum_{m} \alpha_m - \frac{1}{2} \sum_{m,m'} \alpha_m \alpha_{m'} y^m y^{m'} \kappa(x^m, x^{m'}). \tag{6.75}$$

Training a support vector machine with hinge loss and a quantum kernel κ is equivalent to finding the general quantum model that minimises the hinge loss. The "quantum support vector machine" in [2, 3] 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.

6.6 Comparing Kernel-Based and Variational Training

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. (6.73), compare to models whose measurements are trained variationally? Let us revisit Fig. 6.4 in light of the results of the previous section.

We saw in Sect. 6.5.4 how kernel-based training optimises the measurement over a subspace spanned by M encoded training inputs by finding the best coefficients α_m, $m = 1 \ldots M$. We also saw in Sect. 6.5.2 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 [22] because of their poor scaling. Training involves computing the pairwise distances between all training data in the Gram matrix of Eq. (6.73), which has at least a runtime of $\mathcal{O}(M^2)$ in the number of training samples M.[3] In contrast, training neural networks takes time $\mathcal{O}(M)$ that only depends linearly on the number of training

[3] 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. (6.51). However, in maximum margin classifiers, or support vector machines in the stricter sense, most α_m coefficients are zero, and only the distances to a few "support vectors" are needed.

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 [23, 24] 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 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.[4] 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 speedup 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 [25]. 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 [26–28]. While these speedups only hold up under very strict assumptions of data loading oracles, they imply quadratic speedups for rather general settings. 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 [17]), 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

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

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.

References

1. Schuld, M.: Quantum machine learning models are kernel methods (2021). arXiv preprint arXiv:2101.11020
2. Schuld, M., Killoran, N.: Quantum machine learning in feature Hilbert spaces (2018). arXiv preprint arXiv:1803.07128v1
3. Havlíček, V., Córcoles, A.D., Temme, K., Harrow, A.W., Kandala, A., Chow, J.M., Gambetta, J.M.: Supervised learning with quantum-enhanced feature spaces. Nature, **567**(7747), 209–212 (2019)
4. Blank, C., Park, D.K., Rhee, J.K.K., Petruccione, F.: Quantum classifier with tailored quantum kernel. npj Quantum Inf. **6**(1), 1–7 (2020)
5. Park, D.K., Blank, C., Petruccione, F.: The theory of the quantum kernel-based binary classifier. Phys. Lett. A, **384**(21), 126422 (2020)
6. Kübler, J.M., Muandet, K., Schölkopf, B.: Quantum mean embedding of probability distributions. Phys. Rev. Res. **1**(3), 033159 (2019)
7. Lloyd, S., Schuld, M., Ijaz, A., Izaac, J., Killoran, N.: Quantum embeddings for machine learning (2020). arXiv preprint arXiv:2001.03622
8. Liu, Y., Arunachalam, S., Temme, K.: A rigorous and robust quantum speed-up in supervised machine learning (2020). arXiv preprint arXiv:2010.02174
9. Huang, H.Y., Broughton, M., Mohseni, M., Babbush, R., Boixo, S., Neven, H., McClean, J.R.: Power of data in quantum machine learning (2020). arXiv preprint arXiv:2011.01938
10. Steinwart, I., Christmann, A.: Support Vector Machines. Springer Science & Business Media (2008)
11. Schölkopf, B., Smola, A.J.: Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press (2002)
12. Pérez-Salinas, A., Cervera-Lierta, A., Gil-Fuster, E., Latorre, J.I.: Data re-uploading for a universal quantum classifier. Quantum **4**, 226 (2020)
13. McClean, J.R., Boixo, S., Smelyanskiy, V.N., Babbush, R., Neven, H.: Barren plateaus in quantum neural network training landscapes. Nat. Commun. **9**(1), 1–6 (2018)
14. Wolf, M.: Quantum channels and operations: Guided tour (2012). https://www-m5.ma.tum.de/foswiki/pub/M5/Allgemeines/MichaelWolf/QChannelLecture.pdf
15. Jagadish, V., Petruccione, F.: An invitation to quantum channels. Quanta **7**(1), 54–67 (2018)
16. Bergholm, V., Izaac, J., Schuld, M., Gogolin, C., Blank, C., McKiernan, K., Killoran, N.: Pennylane: Automatic differentiation of hybrid quantum-classical computations (2018). arXiv preprint arXiv:1811.04968
17. Chatterjee, R., Yu, T.: Generalized coherent states, reproducing kernels, and quantum support vector machines (2016). arXiv preprint arXiv:1612.03713
18. Schuld, M., Sweke, R., Meyer, J.J.: The effect of data encoding on the expressive power of variational quantum machine learning models (2020). arXiv preprint arXiv:2008.08605
19. Benedetti, M., Garcia-Pintos, D., Perdomo, O., Leyton-Ortega, V., Nam, Y., Perdomo-Ortiz, A.: A generative modeling approach for benchmarking and training shallow quantum circuits. npj Quantum Inf. **5**(1), 1–9 (2019)
20. Cheng, S., Chen, J., Wang, L.: Information perspective to probabilistic modeling: Boltzmann machines versus born machines. Entropy **20**(8), 583 (2018)
21. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press (2004)
22. Rahimi, A., Recht, B., et al.: Random features for large-scale kernel machines. In: NIPS, vol. 3, p. 5. Citeseer (2007)

23. Mitarai, K., Negoro, M., Kitagawa, M., Fujii, K.: Quantum circuit learning (2018). arXiv preprint arXiv:1803.00745
24. Schuld, M., Bergholm, V., Gogolin, C., Izaac, J., Killoran, N.: Evaluating analytic gradients on quantum hardware. Phys. Rev. A **99**(3), (2019)
25. Harrow, A.W., Hassidim, A., Lloyd, S.: Quantum algorithm for linear systems of equations. Phys. Rev. Lett. **103**(15), 150502 (2009)
26. Wiebe, N., Braun, D., Lloyd, S.: Quantum algorithm for data fitting. Phys. Rev. Lett. **109**(5) (2012)
27. Rebentrost, P., Mohseni, M., Lloyd, S.: Quantum support vector machine for big data classification. Phys. Rev. Lett. **113** (2014)
28. Lloyd, S., Mohseni, M., Rebentrost, P.: Quantum principal component analysis. Nat. Phys. **10**, 631–633 (2014)