Quantum Machine Intelligence (2019) 1:65–71
https://doi.org/10.1007/s42484-019-00007-4

REVIEW ARTICLE

Kernel methods in Quantum Machine Learning

Riccardo Mengoni[1] · Alessandra Di Pierro[1]

Received: 28 April 2019 / Accepted: 22 September 2019 / Published online: 15 November 2019

Abstract
Quantum Machine Learning has established itself as one of the most promising applications of quantum computers and Noisy Intermediate Scale Quantum (NISQ) devices. In this paper, we review the latest developments regarding the usage of quantum computing for a particular class of machine learning algorithms known as kernel methods.

Keywords Quantum Machine Learning · Quantum computing · Kernel methods

1 Introduction

In the era of big data, Machine Learning (ML) provides a set of techniques to identify patterns among huge datasets "without being explicitly programmed to perform that task" (Bishop 2016; Mitchell 1997). In the last few years, building on the great success of ML, a new interdisciplinary research topic going under the name of Quantum Machine Learning (QML) has emerged (Schuld 2015; Wittek 2014; Biamonte et al. 2017; Ciliberto et al. 2018; Dunjko and Briegel 2018; Arunachalam and Wolf 2017; Perdomo-Ortiz et al. 2018; Schuld and Petruccione 2018). The aim of QML is to merge in different ways quantum computing and data mining techniques in order to achieve improvements in both fields. As shown in Fig. 1, it is possible to distinguish four approaches to QML, depending on the nature of the dataset under study and the computation device being used (Dunjko et al. 2016).

The Classical-Classical (CC) class refers to ordinary machine learning or to machine learning algorithms that are inspired by the formalism of quantum mechanics. Here the dataset represents some classical system and the algorithm can run on a classical computer (Dong et al. 2019; Canabarro et al. 2019; Amin et al. 2018; Crawford et al. 2016; Stoudenmire and Schwab 2016; Sergioli et al. 2018; Levine et al. 2018). In the Classical-Quantum (CQ) class, algorithms rely on the advantages of quantum computation in order to speed up classical ML methods. Data are assumed to be classical in this class as well (Aïmeur et al. 2013; Mikhail et al. 2016; Wiebe et al. 2015; Barry et al. 2014; Lu and Braunstein 2014; Heim et al. 2015; Bottarelli et al. 2018). Quantum-Classical (QC) refers to the use of classical ML methods to analyse quantum systems (Agresti et al. 2019; Huembeli et al. 2019; Gray et al. 2018; Benedetti et al. 2019; Di Pierro et al. 2018; O'Driscoll et al. 2019; Iten et al. 2018). Finally, in the Quantum-Quantum (QQ) class, both the learning algorithm and the system under study are fully quantum (Yu et al. 2019).

Some very promising results have been obtained relatively to each of the four frameworks. In this paper, we have chosen to focus on the CQ section with the aim to review the main approaches that use quantum mechanics in order to obtain a computational advantage for a specific class of ML techniques called *kernel methods*. Our main motivation is to set a clear background for those who want to start investigations or carry out research in this field. A systematization of the current research in Quantum Machine Learning should include similar work in the other three sectors too, which we plan to accomplish in the future.

In the next section we will introduce kernel methods, with a particular attention to the Support Vector Machine (SVM) supervised learning model. Then, we will discuss the two main approaches to quantizing these methods. We have divided this discussion in two sections. In Section 3 we have collected those approaches which are aimed at the formulation of a quantum algorithm that implements a quantum version of the classical SVM. The second type of approaches is discussed in Section 4 and is aimed at exploiting the

✉ Riccardo Mengoni
riccardo.mengoni@univr.it

Alessandra Di Pierro
alessandra.dipierro@univr.it

[1] Department of Informatics, University of Verona, Verona, Italy

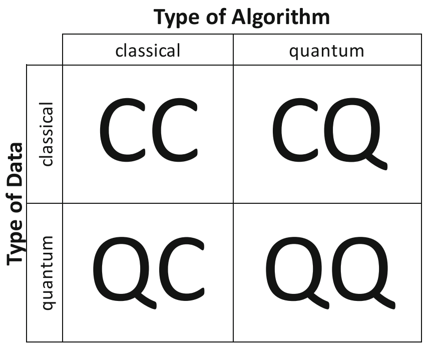


Fig. 1 The first letter in each box refers to whether the system under study is classical or quantum, while the second letter indicates whether a classical or quantum information processing device is used

power of quantum computing to deal specifically with classically intractable kernels.

2 Kernel methods and SVM

Kernel methods (Theodoridis 2008) are classification algorithms that use a kernel function K in order to map data points, living in the input space V, to a higher dimensional feature space V', where separability between classes of data becomes clearer. Kernel methods avoid the explicit calculation of the point coordinates in the new space by means of so called *kernel trick*, which allows us to work in the feature space V' simply computing the kernel of pairs of data points in the input space (Theodoridis 2008).

Intuitively, the "trick" consists in considering the following scenario. Let $\phi : V \to V'$, be a map from the input space V to the enhanced feature space V'. Then a kernel $K : V \times V \to \mathbb{R}$ is a function

$$K(\mathbf{x}_i, \mathbf{x}_j) \equiv \left\langle \phi(\mathbf{x_i}), \phi(\mathbf{x_j}) \right\rangle,$$

representing the inner product $\langle \cdot, \cdot \rangle$ in V', that must satisfy the Mercer condition (Mercer et al. 1909; Mohri et al. 2012) of positive semi-definiteness, i.e., for all choices of n real numbers $(c_1, \ldots, c_n)$ the following relation must hold

$$\sum_{i=1}^{M}\sum_{j=1}^{M} K(\mathbf{x}_i, \mathbf{x}_j) c_i c_j \geq 0.$$

Clearly, calculating the kernel $K(\mathbf{x}_i, \mathbf{x}_j)$ is computationally cheaper than computing coordinates for each new point $\phi(\mathbf{x})$, and, on the other hand, we are never required to explicitly compute $\phi(\mathbf{x_i})$ at any stage of the algorithm. The existence of a concrete mapping $\phi : V \to V'$ is guaranteed by the *Mercer theorem* (Mercer et al. 1909; Mohri et al. 2012), provided that the kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ gives rise to a kernel matrix obeying the Mercer condition.

Support Vector Machine (SVM) is the best known example of kernel method. This supervised binary classifier learns the optimal discriminative hyperplane, based on an input set of M labelled vectors $\{(\mathbf{x}, y) \mid \mathbf{x} \in \mathbb{R}^N, \ y \in \{-1, +1\}\}$. This is achieved by maximizing the distance, i.e., the margin, between the decision hyperplane and the closest points, called support vectors (Cortes and Vapnik 1995).

The SVM optimization problem with hard-margin can be formulated as the problem to find

$$\arg\min_{(\mathbf{w},b)} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 \right\} \text{subject to the constraint } \forall_i \, y_i(\mathbf{w}\cdot\mathbf{x_i} - b) \geq 1,$$

where $(\mathbf{x_i}, y_i)$, with $i = 1 \ldots M$ and $y_i \in \{-1, +1\}$, is the pair of training vector and label, $\mathbf{w}$ is the vector which is normal to the discriminative hyperplane, and b is the offset of the hyperplane.

An important extension of the SVM method described above is the so called *soft margin* SVM, where the best hyperplane is the one that reaches the optimal trade-off between two factors: the minimization of the margin and the restraint of the point deviation from the margin; the latter is expressed by means of slack variables ξ_i tuned by a hyper-parameter C. A soft margin SVM optimization problem is of the form

$$\arg\min_{(\mathbf{w},b)} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{M} \xi_i \right\},$$

subject to the constraint

$$\forall_i \ y_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1 - \xi_i, \qquad \xi_i \geq 0. \tag{1}$$

Usually it is convenient to switch to the dual form, where Lagrange multipliers α_i are introduced in order to include the constraint in the objective function, by obtaining the formulation:

$$\arg\max_{(\alpha_i)} \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j),$$

with $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$, subject to $\sum_i \alpha_i y_i = 0$, $\forall_i \ \alpha_i \geq 0$.

It is worth noticing that only a sparse subset of the α_is are non-zero and that the corresponding $\mathbf{x}_i$ are the support vectors which lie on the margin and determine the discriminant hyperplane.

In this context, a non-linear classification boundary for the SVM is obtained by replacing the term $(\mathbf{x}_i^T \mathbf{x}_j)$ in the objective function with a kernel function $K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x_i})^T(\phi(\mathbf{x_j}))$ satisfying the Mercer condition of positive semi-definiteness. The Lagrangian optimization problem for the soft margin SVM now becomes

$$\arg\max_{(\alpha_i)} \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j),$$

subject to $\sum_i \alpha_i y_i = 0$ with $\forall_i \ \alpha_i \geq 0$.

Note that the dual form of the SVM optimization problem is quadratic in the parameter α_i and it can be efficiently solved with quadratic programming algorithms.

An alternative version of SVM that has a central role in the quantum formulation of the problem is the least-squares support vector machines (LS-SVM) (Suykens and Vandewalle 1999). Here, the constraint defined in Eq. 1 is replaced by the equality constraint

$$\forall_i \;\; y_i(\mathbf{w}\cdot\phi(\mathbf{x_i}) - b) = 1 - e_i,$$

where e_i are errors terms. In this way, optimal parameters $\boldsymbol{\alpha}$ and b that identify the decision hyperplane are found by solving a set of linear equations, instead of using quadratic programming.

The LS-SVM problem can hence be formulated as

$$F\begin{pmatrix} b \\ \boldsymbol{\alpha}\end{pmatrix} = \begin{pmatrix} 0 & \mathbf{1}^T \\ \mathbf{1} & K+\gamma^{-1}I\end{pmatrix}\begin{pmatrix} b \\ \boldsymbol{\alpha}\end{pmatrix} = \begin{pmatrix} 0 \\ \mathbf{y}\end{pmatrix}, \tag{2}$$

where F is a $(M+1)\times(M+1)$ matrix, $\mathbf{1}^T \equiv (1,1,1\ldots)^T$, K is the kernel matrix and γ^{-1} is the trade-off parameter that plays a similar role to C in soft margin SVM. Binary class labels are denoted by the vector $\mathbf{y}\in([-1,1]^M)^T$.

Solving the quadratic programming problem or the least-squares SVM has complexity $O(M^3)$ (Wittek 2014). A bottleneck slowing down the computation is determined by the kernel: for a polynomial kernel $K(\mathbf{x}_i, \mathbf{x}_j)$ of the form $(\mathbf{x_i}^T\mathbf{x_j} + c)^d$, the best algorithm takes $O(M^2 d)$, although in other cases the complexity could be much higher, e.g., for those kernels depending on a distance whose calculation is itself an NP problem.

3 Quantum SVM

The first quantum approach to SVM is due to Anguita et al. (2003). In their work, they consider a discretized version of the SVM, which also takes into account the generalization error of the classifier. This setting inhibits the use of well-known quadratic programming algorithms and optimization can turn into a problem in the NP complexity class.

The authors propose to represent different configurations of the Lagrangian multipliers, α_i, as quantum states $|\alpha_0\alpha_1..\alpha_M\rangle$, and then use Grover algorithm in order to perform an exhaustive search over the configuration space in order to find the maximum of the cost function. It is well known that this task can be accomplished by the Grover quantum algorithm with complexity $O(\sqrt{2^M})$ rather than the $O(2^M)$ required by classical algorithms.

A different approach was proposed by Rebentrost, Mohseni and Lloyd (Rebentrost et al. 2014), which presented a completely new quantum algorithm that implements SVM on a circuit-based quantum computer. This formulation has become very popular in the last few years and it is often referred to as the Quantum SVM (QSVM) algorithm. In order to understand QSVM it is necessary to clarify that classical input training vectors $\mathbf{x}$ are represented by means of quantum states of the form

$$|\mathbf{x}\rangle = \frac{1}{|\mathbf{x}|}\sum_{k=1}^{N}(\mathbf{x})_k\,|k\rangle\,,$$

where the components of the vectors $\mathbf{x}$ are encoded in the amplitude of the quantum state. The authors claim that this whole set of M states could in principle be constructed querying a Quantum Random Access Memory (QRAM), which uses $O(MN)$ hardware resources but only $O(\log MN)$ operations to access them (Giovannetti et al. 2008).

The preliminary step of the QSVM algorithm exploits the fact that dot products can be estimated faster using the QRAM and repeating the SWAP test algorithm on a quantum computer (Buhrman et al. 2001). More precisely, if the desired accuracy is ϵ, then the overall complexity of evaluating a single dot product $\mathbf{x_i}^T\mathbf{x_j}$ is $O(\epsilon^{-1}\log N)$. Calculating the kernel matrix takes therefore $O(M^2\epsilon^{-1}\log N)$, instead of $O(M^2 N\log(1/\epsilon))$ required in the classical case.

The main idea of the QSVM algorithm is to use the LS-SVM formulation of Eq. 2 and rewrite it in terms of quantum states as

$$\hat{F}\,|b,\boldsymbol{\alpha}\rangle = |\mathbf{y}\rangle\,,$$

where $\hat{F} = F/\mathrm{tr}(F)$, with $||F|| \leq 1$. Then the optimal parameters b and $\boldsymbol{\alpha}$ are obtained by applying the efficient quantum matrix inversion algorithm (Harrow et al. 2009). This algorithm requires the simulation of matrix exponentials $e^{-i\hat{F}\Delta t}$, which can be performed in $O(\log N)$ steps (Lloyd et al. 2014).

Moreover, we can add an ancillary qubit, initially in state $|0\rangle$, and use the quantum phase estimation algorithm (Nielsen and Chuang 2011) to express the state $|\mathbf{y}\rangle$ in the eigenbasis $|e_i\rangle$ of $\hat{F}$ and store approximations of the eigenvalues λ_i of $\hat{F}$ in the ancilla qubit:

$$|\mathbf{y}\rangle\,|0\rangle \rightarrow \sum_{i=1}^{M+1}\langle e_i\,|\mathbf{y}\rangle\,|e_i\rangle\,|\lambda_i\rangle.$$

Now apply an inversion of the eigenvalue with a controlled rotation and un-compute the eigenvalue qubit to obtain

$$\sum_{i=1}^{M+1}\frac{\langle e_i\,|\mathbf{y}\rangle}{\lambda_i}\,|e_i\rangle = \hat{F}^{-1}\,|\mathbf{y}\rangle = |b,\boldsymbol{\alpha}\rangle\,.$$

In the training set basis, the solution state for the LS-SVM is

$$|b,\boldsymbol{\alpha}\rangle = \frac{1}{b^2+\sum_{k=1}^{M}\alpha_k^2}\left(b\,|0\rangle + \sum_{k=1}^{M}\alpha_k\,|k\rangle\right).$$

The process of classifying new data $|\mathbf{x}\rangle$ with trained $|\boldsymbol{\alpha}, \beta\rangle$ requires the implementation of the *query oracle*

$$|\tilde{u}\rangle = \frac{1}{\left(b^2 + \sum_{k=1}^{M} \alpha_k^2 |\mathbf{x_k}|^2\right)^{\frac{1}{2}}} \left(b\,|0\rangle\,|0\rangle + \sum_{k=1}^{M} |\mathbf{x_k}|\,\alpha_k\,\,|k\rangle\,\,|\mathbf{x_k}\rangle\right) \tag{3}$$

and also the query state

$$|\tilde{x}\rangle = \frac{1}{M|\mathbf{x}|^2 + 1}\left(|0\rangle\,|0\rangle + \sum_{k=1}^{M} |\mathbf{x}|\,|k\rangle\,|\mathbf{x}\rangle\right). \tag{4}$$

The classification is obtained by computing the inner product $\langle\tilde{x}|\tilde{u}\rangle$ via a swap test (Buhrman et al. 2001). This means that, with the help of an ancillary qubit, the state $|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle_a\,|\tilde{u}\rangle + |1\rangle_a\,|\tilde{x}\rangle)$ is constructed and then measured in the state $|\phi\rangle = \frac{1}{\sqrt{2}}(|0\rangle_a - |1\rangle_a)$ with a success probability given by $P = |\langle\psi|\phi\rangle|^2 = \frac{1}{2}(1 - \langle\tilde{x}|\tilde{u}\rangle)$. The probability P can be estimated to accuracy ϵ in $O(\frac{P(1-P)}{\epsilon^2})$. The class label is decided depending on the value of P: if it is greater than $\frac{1}{2}$, then $|\mathbf{x}\rangle$ is labelled -1; if it is less than $\frac{1}{2}$, then the label of $|\mathbf{x}\rangle$ is 1.

The overall time complexity for both training and classification of the LS-SVM is of $O(\log(NM))$.

In the QSVM algorithm, kernelization can be achieved by acting on the training vector basis, i.e., by mapping each $|\mathbf{x}_i\rangle$ to a d-fold tensor product

$$|\phi(\mathbf{x}_i)\rangle = |\mathbf{x}_i\rangle_1 \otimes |\mathbf{x}_i\rangle_2 \otimes ... \otimes |\mathbf{x}_i\rangle_d.$$

This allows us to obtain polynomial kernels of the form

$$K(\langle\mathbf{x}_i|\mathbf{x}_j\rangle) \equiv \langle\phi(\mathbf{x}_i)|\phi(\mathbf{x}_j)\rangle = \langle\mathbf{x}_i|\mathbf{x}_j\rangle^d$$

that can be computed in $O(d\epsilon^{-1}\log N)$. Note that in the QSVM, the kernel evaluation is directly performed in the high dimensional feature quantum space, while in classical SVM the kernel trick avoids such expensive calculation. However, this is no problem in the quantum case thanks to the exponential quantum speed-up obtained in the evaluation of inner products.

An experimental implementation of the QSVM have been shown in Li et al. (2015) and Patrick et al. (2018). Also, in Windridge et al. (2018), the authors propose a quantized version of Error Correction Output Codes (ECOC) which extends the QSVM algorithm to the multi-class case and enables it to perform an error correction on the label allocation.

4 Quantum computation of hard kernels

In this section we review the main proposals having as a core idea the computation of classically hard kernel via a quantum device. In this context, we can recognize two common threads. On one side, a hybrid classical-quantum learning model takes classical input and evaluates a kernel function on a quantum devices, while classification is performed in the standard classical manner (e.g employing a SVM algorithm). In the second approach instead, a kernel based variational quantum circuit is trained to classify input data. More specifically, a variational quantum circuit (Mcclean et al. 2016) is a hybrid quantum-classical algorithm employing a quantum circuit $U(\theta)$ that depends on a set of parameters θ which are varied in order to minimize a given objective function (see Fig. 2). The quantum circuit is hence trained by a classical iterative optimization algorithm that at every step finds best candidates θ starting from random (or pre-trained) initial values.

Schuld and Killoran recently explored this concepts (Schuld and Killoran 2019) remarking the strict relation between quantum states and feature maps. The authors explain that the key element in both quantum computing and kernel methods is to perform computations in a high dimensional (possibly infinite) Hilbert space via an efficient manipulation of inputs.

In fact it is possible to interpret the encoding of classical inputs $\mathbf{x}_i$ into a quantum state $|\phi(\mathbf{x})\rangle$ as a feature map ϕ which maps classical vectors to the Hilbert space associated with a system of qubits. As said before, two ways of exploiting this parallelism are described.

In the first approach, called by the authors *implicit*, a quantum device takes classical input and evaluates a kernel function as part of a hybrid classification model. This requires the use of a quantum circuit $U_\phi(x)$ implementing the mapping

$$\phi : \mathbf{x} \rightarrow |\phi(\mathbf{x})\rangle = U_\phi(x)|000..0\rangle$$

and which is able to produce a kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle 000..0|\,U_\phi^\dagger(x_i)U_\phi(x_j)\,|000..0\rangle$$

In order for quantum computing to be helpful, such kernel shouldn't be efficiently simulated by a classical computer. It is therefore posed the question of what type of feature map circuits U_ϕ leads to powerful kernels for classical

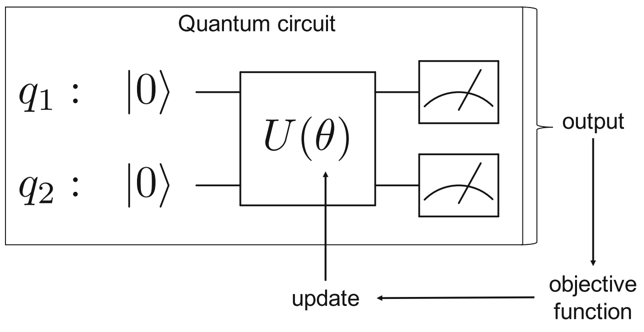


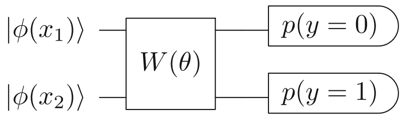
Fig. 2 Schematisation of a variational quantum circuit

learning models like SVM but at the same time are classically intractable. The authors suggest that a way to achieve such a goal is to employ non-Gaussian elements (e.g., cubic phase gate or photon number measurements) as part of the quantum circuit $U_\phi(x)$ implementing the mapping to the feature space.

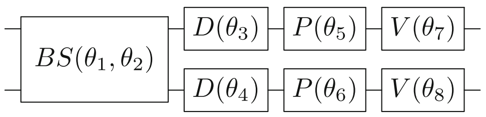
The second approach, addressed in the paper as *explicit*, uses a variational quantum circuit to directly learn a decision boundary in the quantum Hilbert space. In their example, the authors first translate classical input to a quantum squeezed state

$$\mathbf{x} \rightarrow |\phi(\mathbf{x})\rangle = \frac{1}{\sqrt{\cosh(c)}} \sum_{n=0}^{\infty} \frac{\sqrt{(2n)!}}{2^n n!} (-\exp^{i\mathbf{x}} \tanh(c))^n |2n\rangle,$$

then apply to $|\phi(\mathbf{x})\rangle$ the parametrized continuous-variable circuit:



where $W(\theta)$ is a repetition of the following gates:



The components of such gates are, more explicitly,

$$BS(\theta_1, \theta_2) = e^{\theta_1}(e^{i\theta_2}\hat{a}_1^\dagger \hat{a}_2 - e^{-i\theta_2}\hat{a}_1 \hat{a}_2^\dagger),$$

with $\theta_1, \theta_2 \in \mathbb{R}$ and $\hat{a}, \hat{a}^\dagger$ creation and annihilation operators;

$$D(z) = e^{\sqrt{2}i(\mathrm{Im}(z)\hat{x} - \mathrm{Re}(z)\hat{p})},$$

with complex displacement z and finally the quadratic and cubic phase gates

$$P(u) = e^{i\frac{u}{2}\hat{x}^2} \text{ and } V(u) = e^{i\frac{u}{3}\hat{x}^3}.$$

The probability of measuring the Fock state $|n_1, n_2\rangle$ in the state $|2, 0\rangle$ or $|0, 2\rangle$ is interpreted as the probability that the classifier predicts class $y = 0$ or $y = 1$

$$p(|2, 0\rangle) = p(y = 0) \quad \text{and} \quad p(|0, 2\rangle) = p(y = 1)$$

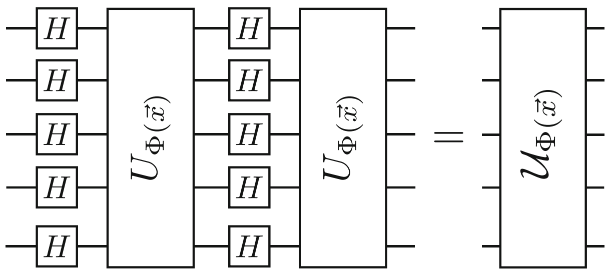
The authors trained such a model on the 'moons' dataset using stochastic gradient descent and showed that training loss s converges to zero after about 200 iterations.

Along the same path, simultaneously to Schuld and Killoran (2019), Havlicek et al. (2019) propose two classifiers that map classical data into quantum feature Hilbert space in order to get a quantum advantage. Again, one SVM classifier is based on a variational circuit that generates a separating hyperplane in the quantum feature space, while the other classifier only estimates the kernel function on the quantum computer.

The two methods are tested on an artificial dataset $\mathbf{x} \in T \cup S \equiv \Omega \subset (0, 2\pi]^2$ where T and S are respectively the training and test sets. This classical input is previously encoded as $\phi_S(\mathbf{x}) \in \mathbb{R}$ where $\phi_S(\mathbf{x}) = (\pi - x_1)(\pi - x_2)$.

On the basis that, in order to obtain an advantage over classical approaches, feature maps need to be based on a circuit that is hard to simulate with classical means, the authors propose a feature map on n-qubits generated by the unitary

$$\mathcal{U}_\Phi(\mathbf{x}) = U_{\Phi(\mathbf{x})} H^{\otimes n} U_{\Phi(\mathbf{x})} H^{\otimes n}$$



where H is the Hadamard gate and

$$U_{\Phi(\mathbf{x})} = \exp\left(i \sum_{S \subseteq [n]} \phi_S(\mathbf{x}) \prod_{k \in S} Z_k \right),$$

with Z_k being the phase shift gate of angle k and S the test set. Such circuit acts on $|0\rangle^n$ as initial state and uses classical data previously encoded in $\phi_S(\mathbf{x})$.

The exact classical evaluation of the inner-product (i.e., kernel) between two states obtained using a circuit $U_\Phi(\mathbf{x})$ is $\#P$ - hard because it is associate to a Tutte partition function which is hard to simulate classically (Goldberg and Guo 2017).

A different approach is taken in Di Pierro et al. (2017), where the same idea of using quantum computation to evaluate a kernel is discussed in the context of Topological Quantum Computation (TQC).

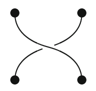
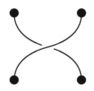
TQC represent a model of quantum computing polynomially equivalent to the circuit based where, instead of using qubits and gates, the computation is performed braiding two-dimensional quasi particles called anyons (Pachos 2012). Moreover, it is well known that some computational problems, such as the approximation of the Jones Polynomial, i.e., an invariant of links and knots, have a more straightforward implementation in TQC (Aharonov et al. 2006).

The approach proposed in Di Pierro et al. (2017) is based on an encoding of input classical data $\mathbf{x}$ in the form of binary strings into braids, which in TQC are expressed by means of evolution operators $\mathbf{B}$. This encoding is constructed by

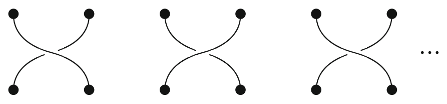
Table 1 Rundown of the main results and references

Category	Method	Title
Quantum version of SVM	Grover algorithm	Quantum optimization for training support vector machines (Anguita et al. 2003)
Quantum version of SVM	HHL algorithm	Quantum support vector machine for big data classification (Rebentrost et al. 2014)
Experimental	NMR 4-qubit quantum processor	Experimental implementation of a quantum support vector machine (Li et al. 2015)
Experimental	IBM quantum experience	Quantum algorithm Implementations for beginners (Patrick et al. 2018)
Quantum version of SVM and ECOC	HHL algorithm	Quantum error-correcting output codes (Win-] dridge et al. 2018)
Kernel methods	Variational quantum circuit	Quantum machine learning in feature Hilbert spaces (Schuld and Killoran 2019)
Kernel methods	Variational quantum circuit	Supervised learning with quantum-enhanced feature spaces (Havlicek et al. 2019)
Kernel methods	Topological quantum computation	Hamming distance kernelisation via topological quantum computation (Di Pierro et al. 2017)

mapping the bit value 0 to the crossing operator σ_i, and the bit value 1 to the adjoint crossing operator $\sigma_{i^\dagger}$:

$\mathbf{0} \longrightarrow \sigma_i =$  $\mathbf{1} \longrightarrow \sigma_i^\dagger =$ 

Hence, a given binary string of length n is uniquely represented by a pairwise braiding of $2n$ strands, i.e., by a braid $B \in B_{2n}$ as shown below.

$\mathbf{010}\ldots \longrightarrow$ 

Therefore, applying the braiding $\mathbf{B}_u$ associated with the binary string u to the vacuum state of the anyonic quantum system $|\psi\rangle$ defines an embedding ϕ into the Hilbert space $\mathcal{H}$ of the anyonic configurations:

$\phi : u \to \mathbf{B}_u\,|\psi\rangle$

The authors finally show that scalar product of anyonic quantum states obtained with such mapping generates a kernel that depends on the hamming distance between the input strings as follows

$$K(u,v) \equiv \langle\psi|\,\mathbf{B}_u^\dagger \mathbf{B}_v\,|\psi\rangle = \left(\frac{\langle \mathbf{Hopf}\rangle}{d}\right)^{d_H(u,v)}$$
$$= \left(\frac{A^4 + A^{-4}}{A^2 + A^{-2}}\right)^{d_H(u,v)}$$

where $\langle \mathbf{Hopf}\rangle$ indicates the Kaufman polynomial (Kauffman 1987) in the variable A that is associated to the so called Hopf link, $d = A^2 + A^{-2}$ and $d_H(u,v)$ is the hamming distance between input strings u and v.

Despite this example does not provide a computationally hard kernel, the authors suggest that a more complex braid mapping of the input may lead naturally to a classically intractable kernel since the calculation of Kaufman polynomial belongs to the $\#P$ - hard class (Goldberg and Guo 2017).

5 Conclusion

In this paper, we have reviewed the main approaches to the design of algorithms for kernel methods in ML, which exploit the power of quantum computing to achieve a computational advantage with respect to the classical approaches. We divided the literature on this problem into two main categories. On the one side, there are attempts to formulate quantum versions of support vector machine running on a gate model quantum computer. On the other side, we grouped different approaches whose core idea relies on the use of quantum computing techniques in order to deal with classically intractable kernels. In Table 1, we give a schematic description of the various results that we have discussed together with the relative article where they appear.

References

Agresti I et al (2019) Pattern recognition techniques for boson sampling validation. Phys Rev X 9:14

Aharonov D, Jones V, Landau Z (2006) A polynomial quantum algorithm for approximating the Jones polynomial. In: Proceedings of the 38th annual ACM symposium on theory of computing, pp 427–436

Aïmeur et al (2013) Quantum speed-up for unsupervised learning. Mach Learn 90:261–287

Amin MH et al (2018) Quantum Boltzmann machine. Phys Rev X 8:11

Anguita D et al (2003) Quantum optimization for training support vector machines. Neural Netw 16:763–770

Arunachalam S, Wolf Ronaldde (2017) A survey of quantum learning theory, arXiv:1701.06806

Barry J et al (2014) Quantum partially observable Markov decision processes. Phys Rev A 90:032311

Benedetti M et al (2019) Adversarial quantum circuit learning for pure state approximation. New J Phys 21:043023

Biamonte J et al (2017) Quantum machine learning. Nature 549:195–202

Bishop C (2016) Pattern recognition and machine learning, vol 738. Springer, New York

Bottarelli L et al (2018) Biclustering with a quantum annealer. Soft Comput 22:6247–6260

Buhrman H, Cleve R, Watrous J, De Wolf R (2001) Quantum fingerprinting. Phys Rev Lett 87:4

Canabarro A, Fernandes Fanchini F, Malvezzi AL, Pereira R, Chaves R (2019) Unveiling phase transitions with machine learning. arXiv:1904.01486

Ciliberto C et al (2018) Quantum machine learning: a classical perspective. Proc R Soc A: Math Phys Eng Sci 474:20170551

Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20:273–297

Crawford D et al (2016) Reinforcement learning using quantum Boltzmann machines, arXiv:1612.05695

Di Pierro A et al (2017) Distance kernelisation via topological quantum computation theory and practice of natural computing. Lect Notes Comput Sci 10687:269–280

Di Pierro A et al (2018) Homological analysis of multi-qubit entanglement. Europhys Lett 123:30006

Dong XY, Pollmann F, Zhang XF (2019) Machine learning of quantum phase transitions. Phys Rev B 99:121104

Dunjko V, Briegel HJ (2018) Machine learning & artificial intelligence in the quantum domain: a review of recent progress. Rep Prog Phys 81:074001

Dunjko V et al (2016) Quantum-enhanced machine learning. Phys Rev Lett 117:6

Giovannetti V, Lloyd S, Maccone L (2008) Quantum random access memory. Phys Rev Lett 100:4

Goldberg LA, Guo H (2017) The complexity of approximating complex-valued ising and tutte partition functions. Computational Complexity 26:765–833

Gray J et al (2018) Machine-learning-assisted many-body entanglement measurement. Phys Rev Lett 121:6

Harrow AW, Hassidim A, Lloyd S (2009) Quantum algorithm for linear systems of equations. Phys Rev Lett 103:4

Havlicek V, Córcoles AD et al (2019) Supervised learning with quantum-enhanced feature spaces. Nature 567:2019–212

Heim B et al (2015) Quantum versus classical annealing of ising spin glasses. Science 348:215–217

Huembeli P et al (2019) Automated discovery of characteristic features of phase transitions in many-body localization. Phys Rev B 99:6

Iten R et al (2018) Discovering physical concepts with neural networks, arXiv:1807.10300

Kauffman LH (1987) State models and the Jones polynomial. Topology 26:395–407

Levine Y et al (2018) Deep learning and quantum entanglement: fundamental connections with implications to network design. In: International conference on learning representations

Li Z, Liu X, Xu N, Du J (2015) Experimental realization of a quantum support vector machine. Phys Rev Lett 114:5

Lloyd S, Mohseni M, Rebentrost P (2014) Quantum principal component analysis. Nat Phys 10:631–633

Lu S, Braunstein SL (2014) Quantum decision tree classifier. Quantum Inf Process 13:757–770

Mcclean JR, Romero J, Babbush R, Aspuru-Guzik A (2016) The theory of variational hybrid quantum-classical algorithms. New J Phys 18:023023

Mercer J et al (1909) Functions of positive and negative type and their connection the theory of integral equations, 209 Philosophical Transactions of the Royal Society of London

Mikhail V et al (2016) Altaisky towards a feasible implementation of quantum neural networks using quantum dots. Appl Phys Lett 108:103108

Mitchell T (1997) Machine learning. McGraw Hill, New York

Mohri M et al (2012) Foundations of machine learning, vol 432. MIT Press, Cambridge

Nielsen MA, Chuang IL (2011) Quantum computation and quantum information. Cambridge University Press, New York

O'Driscoll L et al (2019) A hybrid machine learning algorithm for designing quantum experiments. Quantum Mach Intell 1:1–11

Pachos JK (2012) Introduction to topological quantum computation. Cambridge University Press, New York

Patrick J et al (2018) Coles quantum algorithm implementations for beginners, arXiv:1804.03719

Perdomo-Ortiz A et al (2018) Opportunities and challenges for quantum-assisted machine learning in near-term quantum computers. Quantum Sci Technol 3:030502

Rebentrost P, Mohseni M, Lloyd S (2014) Quantum support vector machine for big data classification. Phys Rev Lett 113:5

Schuld M, Killoran N (2019) Quantum machine learning in feature Hilbert spaces. Phys Rev Lett 122:6

Schuld M, Petruccione F (2018) Supervised learning with quantum computers, vol 287. Springer International Publishing, Berlin

Schuld M, Sinayskiy I, Petruccione F (2015) An introduction to quantum machine learning. Contemp Phys 56(2):172–185

Sergioli G et al (2018) A quantum-inspired version of the nearest mean classifier. Soft Comput 22:691–705

Stoudenmire E, Schwab DJ (2016) Supervised learning with tensor networks. Advances in neural information processing systems (NIPS Proceedings) 29:4799–4807

Suykens JAK, Vandewalle J (1999) Least squares support vector machine classifiers. Neural Process Lett 9:293–300

Theodoridis S (2008) Pattern recognition, vol 984. Elsevier Academic Press, Cambridge

Wiebe N et al (2015) Quantum algorithms for nearest-neighbours methods for supervised and unsupervised learning. Quantum Info Comput 15:316–356

Windridge D, Mengoni R, Nagarajan R (2018) Quantum error-correcting output codes. Int J Quantum Info 16:1840003

Wittek P (2014) Quantum machine learning, vol 176. Elsevier Academic Press, Cambridge

Yu S, Albarrán-Arriagada F, Retamal JC, Wang YT, Liu W, Ke ZJ, Meng Y, Li ZP, Tang JS, Solano E, Lamata L, Li CF, Guo GC (2019) Adv Quantum Technol 2(7–8):1800074