PAPER • OPEN ACCESS

Automatic design of quantum feature maps

To cite this article: Sergio Altares-López *et al* 2021 *Quantum Sci. Technol.* **6** 045015

View the article online for updates and enhancements.

You may also like

- Biomass-derived Materials for Advanced Vanadium Redox Flow Batteries
Yuhui Ge, Ming Yong, Xiangkang Zeng et al.

- Corrigendum: Winter cold extremes over the eastern North America: Pacific origins of interannual-to-decadal variability (2023 *Environ. Res. Lett.*18 054006)
Yao Ge and Dehai Luo

- Complexities in the energy-transport co-transformation
David Daniels and Sonia Yeh

Delft Circuits
Pioneering i/o for advanced technologies
Connect to
your ambitions
Flexible, high-density, cryogenic i/o

This content was downloaded from IP address 109.81.162.232 on 16/10/2025 at 09:39

Quantum Science and Technology

OPEN ACCESS

RECEIVED
17 May 2021

REVISED
25 July 2021

ACCEPTED FOR PUBLICATION
4 August 2021

PUBLISHED
19 August 2021

Original content from this work may be used under the terms of the Creative Commons Attribution 4.0 licence.

Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.

PAPER

Automatic design of quantum feature maps

Sergio Altares-López[1,2], **Angela Ribeiro**[1] **and Juan José García-Ripoll**[3,*]

[1] Consejo Superior de Investigaciones Científicas, Centre for Automation and Robotics-CAR (CSIC-UPM), Ctra. Campo Real km. 0,200, 28500 Arganda, Spain
[2] Universidad Politécnica de Madrid, Programa de Doctorado de Automática y Robótica, Calle de José Gutiérrez Abascal 2, 28006 Madrid, Spain
[3] Consejo Superior de Investigaciones Científicas, Instituto de Física Fundamental IFF-CSIC, Calle Serrano 113b, 28006 Madrid, Spain
[*] Author to whom any correspondence should be addressed.

E-mail: juanjose.ripoll@csic.es

Keywords: quantum machine learning, genetic algorithms, artificial intelligence, automatic quantum classifier generation, optimization, quantum computing

Abstract

We propose a new technique for the automatic generation of optimal ad-hoc ansätze for classification by using quantum support vector machine. This efficient method is based on non-sorted genetic algorithm II multiobjective genetic algorithms which allow both maximize the accuracy and minimize the ansatz size. It is demonstrated the validity of the technique by a practical example with a non-linear dataset, interpreting the resulting circuit and its outputs. We also show other application fields of the technique that reinforce the validity of the method, and a comparison with classical classifiers in order to understand the advantages of using quantum machine learning.

1. Introduction

Quantum machine learning is an emerging field of research that bridges the progress in quantum computing hardware and algorithms with ideas and problems coming from artificial intelligence. The field is suffering steady and fast progress but already features a large corpus of algorithms and applications [1, 2]. On the one hand, we find applications, such as clustering [3], quantum anomaly detection [4], dimensionality reduction [5–7] or support vector machines (SVMs) [8], which build on the HHL algorithm for matrix inversions or related matrix–vector operations. On the other hand, we place algorithms that are ready for near-term intermediate-scale quantum devices, and which are typically based on parameterized quantum circuits [9] that implement autoencoders [10], SVMs and quantum classifiers [11, 12], or generative adversarial networks [13, 14], among other applications.

Due to their simplicity and immediate experimental access, this second framework has gained considerable interest. However, a key problem of parameterized circuit is their design, both from the point of view of the structure of the circuit as well as their parameterization. The structure and design condition the expressive power of the circuit [15, 16], and its capacity to explore the Hilbert space and encode probability distributions more efficiently than other generative models. However, too expressive circuits can be subject to local minima and barren plateaus [17, 18] that prevent reaching the optimal parameterizations. Partial solutions to these challenges include adaptive initialization strategies [19], pruning of circuits [20], density matrices and random features for distribution approximation [21], simultaneous optimization of parameters and rotation generators [22], or the implementation of global optimization strategies such as genetic algorithms [23–29] that optimize gates or structure.

In this work, we focus on the problem of supervised learning using quantum feature maps [11, 12] that are optimized with a genetic algorithm, which use mutations as operators, that spread out the search range and allow to avoid local minima [30, 31]. As compared to earlier variational works [25], we provide a comprehensive solution that automatically designs both the structure and the parameterization of the feature map circuit. This solution uses a multiobjective genetic algorithm to optimize the accuracy and generalization power of the map, while minimizing the circuit size. We test this method against both

© 2021 The Author(s). Published by IOP Publishing Ltd

synthetic and realistic benchmarks, finding remarkable accuracy for all problems. Moreover, the Pareto strategy and our weighting of gates seems to produce quantum feature maps that are largely uncorrelated. This hints at the possibility of constructing hybrid quantum-inspired strategies for machine learning based on these ideas.

The structure of this work is as follows. In section 2 we review the method of quantum feature maps and quantum kernels for supervised learning with SVMs. We introduce a new function (6) that seems to exhibit good separation properties and will be used in simulations. With this knowledge, section 3 introduces the algorithm for genetically designed quantum kernels. After a brief review of genetic algorithms in section 3.1, we introduce an encoding of quantum feature maps as binary strings of genes. The genetic map from section 3.2.1 is a small example with only five bits, but exemplifies how to encode structure, types of gates, dependence on the input parameters and numerical parameterization of the circuit. This contrasts with other works [25] where structure and parameters were optimized separately, with different methods. In section 3.2.2 we describe the fitness function, which is designed for a multiobjective optimization of the accuracy, the capacity for generalization and the simplicity of the quantum feature map. We also describe how a Pareto search and elitist genetic operations can be designed to help in this optimization. In section 3.2.4 we provide an algorithm of the training part. Section 4 presents the results of applying our algorithm to synthetic and realistic benchmarks, in sections 4.1 and 4.3, respectively, calculating the kernel matrix on a python emulator of an ideal quantum computer. In these examples we also see how the optimization converges to low-entanglement feature maps, while still having good accuracy and generalization. Based on this, in section 4.2 we discuss how such circuits could be amenable for interpretation. Finally, section 5 summarizes the main conclusions and possible avenues for future exploration.

2. Quantum kernel method

In this work we will focus on the supervised training of binary classifiers. Given a training dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^{L}$ with normalized feature vectors $\mathbf{x}_i \in \mathbb{R}^d$ and binary classes $y_i \in \{+1, -1\}$, we can design a model $f(\mathbf{x})$ that predicts the class of any other point, either in this set or in unseen data. The SVM is one of the earliest binary classification techniques. Developed for linearly separable data, this method constructs a hyperplane with normal $\mathbf{w}$ and displacement b such that the two classes $y = +1$ and $y = -1$ lay on opposite sides of the hyperplane. The classifier has a simple form, given by a sign function

$$f(\mathbf{x}) = \text{sign}\left(\mathbf{w}^{\mathrm{T}} \cdot \mathbf{x} + b\right). \tag{1}$$

The hyperplane is constructed using *support vectors* from the training set

$$\mathbf{w} = \sum_i \beta_i y_i \mathbf{x}_i, \tag{2}$$

in a way that maximizes the margin between those vectors and the hyperplane.

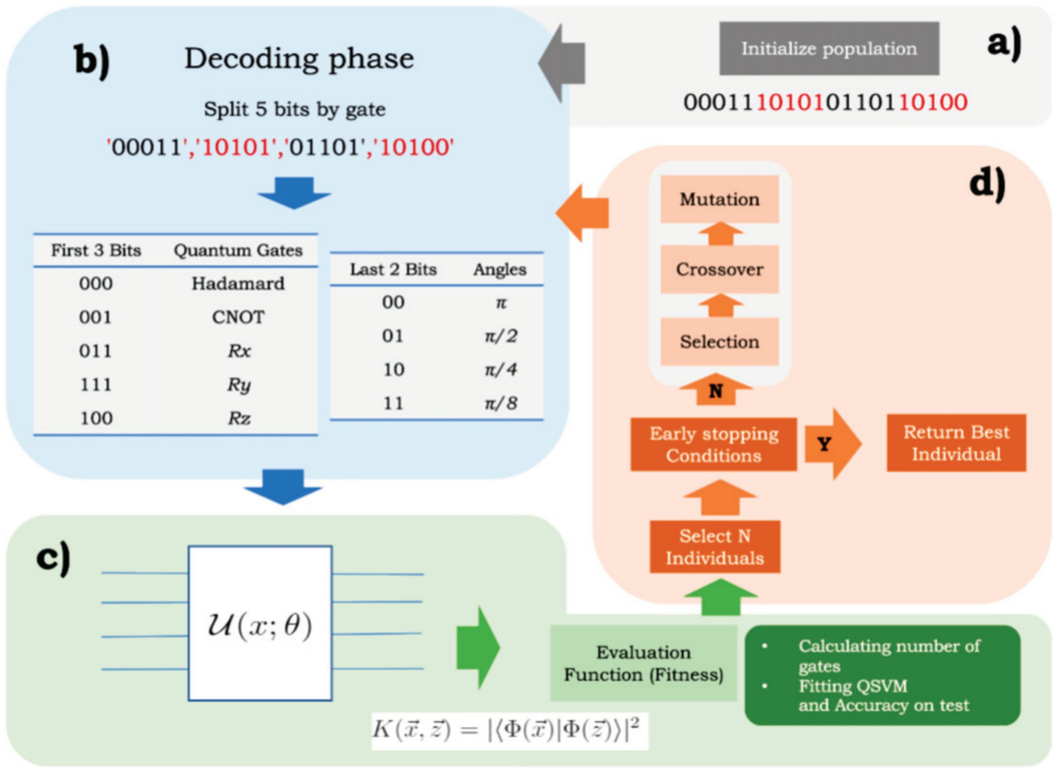
There are various techniques that turn SVM into a universal classifier, working with data that is not linearly separable. One is to construct additional features or variables out the original vectors, enlarging the dimension of the classification space $\tilde{\mathbf{x}}_i := \mathbf{\Phi}(\mathbf{x}_i) \in \mathbf{R}^r$, with $r \gg d$. By raising the dimensionality, the so called *feature map* can transform the problem into a linearly separable one. Interestingly, the classifier can be inferred from a *kernel function* that encodes the scalar product between the new features

$$K(\mathbf{x}, \mathbf{x}') = \mathbf{\Phi}(\mathbf{x})^{\mathrm{T}} \mathbf{\Phi}(\mathbf{x}'). \tag{3}$$

This can be seen from the expression of the hyperplane $\mathbf{w}$ in terms of the new features, and how this all fits into the final classifier

$$\begin{aligned} f(x) &= \text{sign}\left(\sum_i \beta_i y_i \mathbf{\Phi}(\mathbf{x}_i)^{\mathrm{T}} \mathbf{\Phi}(\mathbf{x}) + b\right) \\ &= \text{sign}\left(\sum_i \beta_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right) \in \{+1, -1\}, \end{aligned} \tag{4}$$

which determines the class based on the sign of the output. Importantly, by Mercer's theorem [32], we do not need to know the form of the feature map—which may even be an infinite-dimensional function, but just a kernel function $K(\mathbf{x}, \mathbf{x}')$ that has the right positivity properties to encode a scalar product.



First 3 Bits	Quantum Gates
000	Hadamard
001	CNOT
011	*Rx*
111	*Ry*
100	*Rz*

Last 2 Bits	Angles
00	π
01	$\pi/2$
10	$\pi/4$
11	$\pi/8$

Figure 1. Technique scheme. (a) The initial population is initialized. (b) We can see the decoding process based on five bits per quantum gate and angle. (c) This individual is used in the fitness function as feature map for QSVM in order to calculate accuracy and number of gates. (d) The genetic algorithm process continue with phases already seen until early stopping condition so that the most optimized ansatz for this dataset is provided.

When developing quantum classifiers for classical data, the usual approach is to engineer a feature map from classical to quantum features $|\Phi(\mathbf{x})\rangle$ [11, 12]. This map can be trivial—e.g. encoding data into quantum register states or quantum amplitudes, but more generally it is a parameterized unitary transformation, built from quantum gates that are depend on the input features $|\Phi(\mathbf{x})\rangle := \mathcal{U}(\mathbf{x};\boldsymbol{\theta})|0\rangle^n$, and on some additional controls $\boldsymbol{\theta}$. The feature map can be combined with further classification circuits or measurements, to create the so called *quantum neural networks*. However, as argued in references [11, 12, 33], we could simply use those circuits to evaluate a kernel

$$K(\mathbf{x},\mathbf{x}') = |\langle\Phi(\mathbf{x})|\Phi(\mathbf{x}')\rangle|^2 = \left|\langle 0^n|\mathcal{U}(\mathbf{x};\boldsymbol{\theta})^\dagger\mathcal{U}(\mathbf{x}';\boldsymbol{\theta})|0^n\rangle\right|^2, \tag{5}$$

and derive the corresponding SVM classifier $f(\mathbf{x};\boldsymbol{\theta})$.

In this work we also test a different type of quantum kernel

$$K(\mathbf{x},\mathbf{x}') = \mathrm{Re}\langle\Phi(\mathbf{x})|\Phi(\mathbf{x}')\rangle = \mathrm{Re}\langle 0^n|\mathcal{U}(\mathbf{x};\boldsymbol{\theta})^\dagger\mathcal{U}(\mathbf{x}';\boldsymbol{\theta})|0^n\rangle. \tag{6}$$

By not squaring the scalar product between vectors, the function resembles more the original motivation for $K(\mathbf{x},\mathbf{x}')$. Moreover, as we have confirmed numerically, this kernel allows for sharper separations and more easy convergence of the optimizer for larger datasets composed of 22 features (cf figure 7(a)). However, while this choice is neutral from a classical simulation point of view, it is more complicated to evaluate in a quantum computer. Unlike reference [12], to estimate $K(\mathbf{x},\mathbf{x}')$ we would need to use an ancillary qubit, prepared in a quantum superposition, and controlling the $\mathcal{U}$ operation, to estimate the scalar product as the result of an interference process.

This quantum kernel method is usually combined with some kind of iterative update of the parameters $\boldsymbol{\theta}$, to maximize a cost function that includes the accuracy of the model and some other regularizations. Depending on the expressive power of the underlying feature map, this approach can lead to *barren plateaus* and other obstacles that prevent a good training. For that reason, in this work we explore a global optimization method that trains both the parameters $\boldsymbol{\theta}$ *as well as the structure* of the quantum circuit $\mathcal{U}$, by using evolutionary artificial intelligence techniques, also known as *genetic algorithms*.

3. Genetically designed quantum kernels

3.1. Overview of genetic algorithms

Genetic algorithms are meta-heuristic optimization techniques based on the theory of evolution. The algorithms perform a guided search in a space of solutions, evolving a population of individuals with encoded the feature maps, through the application of *genetic operations*. In each algorithm iteration or *generation*, the resulting *offspring* is selected in order to improve one or more objectives. As a result of evolutionary pressure, the collective is more likely to select the best suited individuals from a very large configuration space, in an efficient way.

A very important ingredient in the genetic algorithm is the *fitness* function. This function depends on some metric that we wish to maximize (or minimize), as well as other regularizations. However, as we show in this work, it is possible for a genetic algorithm to achieve more than one goal, performing a multiobjective optimization process. In this case, the individuals in each generation that maximize (or minimize) the fitness function are called the *Pareto front*. More precisely, a solution x is called dominated by another solution y only if y is equal or better than x solution with respect to all the objectives of the function. The Pareto front is the set of non-dominated solutions which satisfy all objectives defined in the fitness function, and which is progressively improved by the evolution algorithm.

A key ingredient for the success and utility of a genetic algorithm is the choice of genetic operations that evolve the population of individuals, as seen in figure 1(d). The *selection* operator chooses a subset of the existing population to create a new generation using the crossover and mutation operators. The *mutation* operator randomly alters the information of selected individuals to explore places far from the solution space. The *crossover* is an stochastic operation that allows even more drastic explorations, by allowing two individuals to exchange their genetic information. Note that, while the probability of selection is proportional to the fitness of the individuals, the mutation and crossover probabilities are fixed values that have been tuned for performance.

Finally, the genetic algorithm typically involves some *stopping early conditions,* statements which determine whether the evolution process has achieved its goal. Some possible strategies are checking the convergence or saturation of the fitness objective, a minimum accuracy threshold that keeps the process going for further steps or defining a maximum number of generations.

3.2. Genetic quantum feature map

We will now describe a multiobjective genetic algorithm that automatically designs and optimizes quantum classifiers based on quantum feature maps and SVMs. The algorithm explores a configuration space of parameterized quantum circuits that potentially represent feature maps. It looks for those circuits that, once trained in a quantum support vector machine (QSVM) method, maximize the accuracy with which they generalize to the validation data set, while minimizing the complexity of the circuit, which is measured in terms of circuit depth and difficulty of operations.

The complete algorithm is summarized in figure 1. The process starts with an initial population of individuals, represented by bit strings. The evaluation function decodes each individual, creating an associated quantum circuit (see section 3.2.1). This circuit, together with the training dataset, is used to implement a quantum kernel SVM algorithm, computing the fitness function (see section 3.2.2). The best individuals have more probability to be elected and subjected to the different genetic operations, such as mutation, crossover and selection, creating a new generation of individuals or quantum circuits. The whole process is repeated until we meet the convergence criteria.

3.2.1. Encoding

The first step for engineering our algorithm is to design a map from the genetic information to the quantum circuit that we wish to characterize. In our model, the genes will be binary strings that encode local, entangling and parameterized quantum gates. To create a minimalistic encoding that exemplifies all types of gates, we use five bits per gene $s_0s_1s_2s_3s_4$, as shown in figure 1(b). We aim to create a quantum circuit acting on M qubits with a maximum of N layers and use $M \times N \times 5$ bits in total. For simplicity, the order of actuation of the genes is sequential. Thus the ith gene operates on the jth qubit of the quantum register and possibly depends on the kth a variable from the input data $\mathbf{x} \in \mathbb{R}^d$, with $j = i \bmod M$ and $k = i \bmod d$.

The mapping from bits to gates is also very straightforward. The first three bits $s_0s_1s_2$ determine whether the gate is fixed—a Hadamard or a CNOT gate, or whether it is a local rotation parameterized by a value in the input data, $R_\alpha(\theta_i x_k) = \exp(-i\theta_i x_k \sigma_k^\alpha)$. When the gate is parameterized, the first bits $s_0s_1s_2$ select the rotation axis, the last two bits select a proportionality parameter $\theta_i = \pi 2^{-s_3-2s_4}$. When the gate is a CNOT,

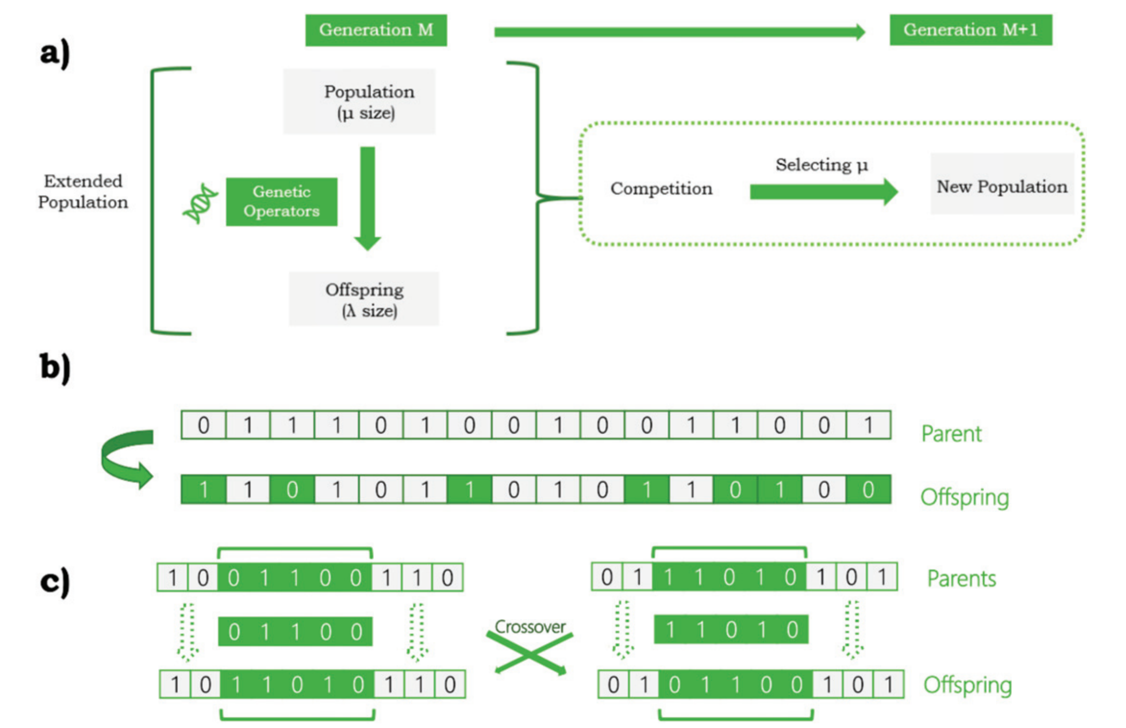


Figure 2. Genetic operators. (a) Genetic algorithm $\mu + \lambda$ strategy. The initial population μ, after applying the genetic operators, produces an offspring of λ size. This offspring competes with the parents and from this competition a number of individuals equal to the size of the individual population μ is selected. This process is repeated throughout the generations of evolution. (b) Mutation. A *bit flip* mutation affects a fraction of the genes, giving rise to a different individual. (c) Crossover. Randomly selected genes (marked in green) are exchanged between two individuals, keeping the rest of the genes chain constant.

it acts on consecutive qubits, j and $j + 1 \bmod M$. All other combinations of bits not reflected in figure 1(b) are taken to be just identity operations.

Note that the previous selection consists of a couple of Clifford gates and a uniform sampling of rotations parameterized by the input data. With some additional bits one could construct a more general dataset, $R_\alpha(\theta_i)$ with gates that depend on no parameters. Such a set, with our choice of angles, would include CNOT, H, S and T gates, and be universal. However, for the problems we discuss below, this more complete approach has not been necessary.

Finally, note that our encoding allows for (i) feature selection, (ii) elimination of gates, (iii) change in the feature weight and (iv) combinations of different features along the same qubits. The elimination of gates and features happens whenever a gene is mutated to represent the identity or a gate that does not depend on a feature. This simple mechanism allows suppressing gates that are irrelevant, or less important, reducing the size of the circuit, the number of layers, and even eliminating qubits from the kernel—e.g. when a qubit has no gate, it does not influence the outcome. The same mechanism allows features to be eliminated from the ansatz, or to emphasize the role of other features, or combinations thereof, through their placement in the circuit, creating complex non-linear dependencies among them. Finally, another important characteristic of the method seems to be the scaling of features by the mutable weights θ_i mentioned before. This tuneability seems to compensate the reduced set of non-parameterized gates.

3.2.2. Fitness function

Our fitness function is designed to maximize the accuracy and minimize the complexity of the variational circuit. To measure the latter, we introduce a *size metric*, labeled SM, which assigns different costs to the number of local gates N_{local} and the number of entangling gates N_{CNOT}, weighted as follows

$$\text{Size Metric (SM)} = \frac{N_{\text{local}} + 2N_{\text{CNOT}}}{N_{\text{qubits}}}. \tag{7}$$

The second ingredient in the fitness function is the *accuracy* of the encoded circuit. To compute this metric, we divide the data into a training set and a test set. We use the quantum circuit and the training set to compute the classifier $f(\mathbf{x})$ in the quantum kernel SVM. We then estimate the accuracy of the model $f(\mathbf{x})$ over the test set, as the fraction of points that are properly classified.

We aim to maximize both quantities in a multiobjective optimization process, creating a *Pareto front* that carefully balances the relative importance of both figures of merit. A high weight in accuracy can produce a collapse into one individual loosing the necessary genetic diversity to be able to minimize the quantum circuit size along the evolution. On the other hand, a very small number of gates can hinder the power of the quantum kernel to separate the features. In order to achieve a proper balance, we engineer a fitness function that increases the relevance of the SM as the accuracy approaches its limiting value 1, using the following multiobjective fitness function

$$\text{Weights Control}(W_{\mathrm{C}}) = \mathrm{SM} + \mathrm{SM} * \text{accuracy}^2. \tag{8}$$

3.2.3. Genetic operators

In the genetic algorithm we use selection, mutation and crossover operators. The selection operator is a multiobjetive *non-sorted genetic algorithm* II (NSGA-II). This algorithm decides which individuals survive to the next generation based on Pareto-dominance and density-based metrics [34]. This algorithm has a strong tendency to keep individuals with higher fitness, because it selects the individuals after ordering the population by dominance. This means that the NSGA-II selects as parents the best individuals with highest accuracy in test and less number of layers are taken from each generation.

Since our feature maps are coded in binary format, we can use a *bit flip* mutation operator (cf figure 2(b)). The value p_{mut} indicates the probability of an individual to be mutated. Once an individual is selected for mutation, each gene can be flipped with a probability p_{ind}. As for the crossover operator, we implement a binary swap of contiguous bit substrings (see figure 2(c)). The value p_{cross} determines probability that a crossover takes place, while the beginning and end of the swapped bits are randomly chosen along the complete strings.

We increase the elitism of the algorithm with a mu plus lambda $(\mu + \lambda)$ algorithm. This strategy modifies how we create the next generation of individuals, establishing a competition between the current population (size μ) and the offspring (λ) that is obtained by genetic operators, as sketched in figure 2(a). This competition ensures genetic diversity while it also preserves the best individuals that have been obtained through the evolutionary algorithm [35].

All hyperparameters previously mentioned—crossover and mutation probabilities and population sizes—were optimized and tested, to achieve a good compromise between convergence speed and optimal classification. After several tests, the optimal hyperparameters were found to be 30% probability of crossover and 70% mutation, with a 20% probability of bits being mutated. This is an interesting balance that allows exploring drastic changes in the population through crossover, while maintaining a high rate of small changes through mutation. While this may seem very aggressive, the random component is kept in check by the high elitism of the competition between children and parents in the mu plus lambda strategy.

3.2.4. Training algorithm

The training algorithm consists on six steps, which are repeated until convergence. **Step 1**: first, we create an initial population of individuals as binary chains with $M \times N \times 5$ bits, where M and N are the maximum number of qubits and layers, respectively—hyperparameters fixed throughout the optimization. **Step 2**: as sketched in figure 1(b), each individual is decoded and transformed into a quantum circuit. Figure 3 illustrates the outcome of decoding a small string. **Step 3**: using this ansatz circuit, we compute the kernel of a QSVM, training a classifier with the training dataset. **Step 4**: we compute the accuracy of the classifier using the test set. This is the metric which tends to be maximized. **Step 5**: we compute the *effective size* of the circuit, as a weighted sum of the number of local and entangling gates in the circuit. This is the metric that tends to be minimized. **Step 6**: based on the two fitness objectives—accuracy and circuit size, we apply the genetic algorithm operators of selection, crossover and mutation, producing the next generation of individuals. We then return to **step 2** and repeat process until convergence.

4. Results and discussion

4.1. Toy model

In order to proof this new method, we use the Moons synthetic non-linear dataset with two classes, shown in figure 4(a), and generated using Scikit [36]. The 150 datapoints are scaled between $[-1, +1]$ as a preprocessing step, and randomly split into a training (70%) and test (30%) sets. We use the training set to train the circuits and the test set to predict and calculate the accuracy with respect to the ground truth. For this experiment, the initial number of qubits and the number of layers are set to 6. During evolution, the circuits are progressively optimised by the genetic algorithm, minimising the size of the ansatz while

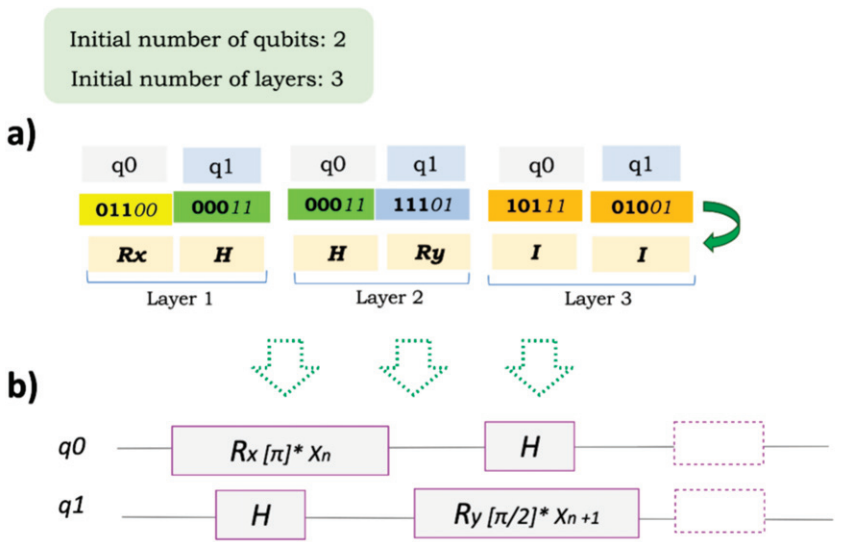
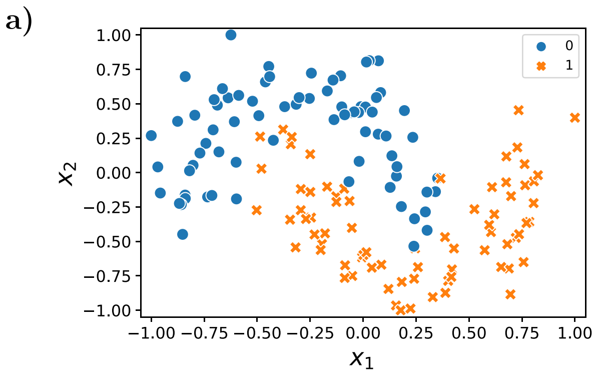
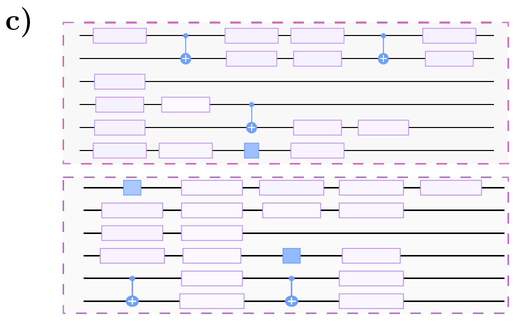


Figure 3. Scheme of ansatz decoding. (a) Initial number of qubits and number of layers predefined by user. A binary chain is created as $M \times N \times 5$. Each five bits build a gate. First three bits decode quantum gate type and two last bits the angle. As it can be seen, *layer* 3 is composed by combinations of bits not reflected in figure 1(b) acting as identity operators and allowing to reduce the size of the circuit. (b) We observe the resultant ansatz after the decoding phase, based on figure 1(b) for this binary chain.



b)

Variables	Value
Generations	5000
Population (μ)	100
Offspring (λ)	15
Qubits	6
Max. Layers	6
Crossover Prob.	0.3
Mutation Prob.	0.7
Mutation Prob. Ind.	0.2



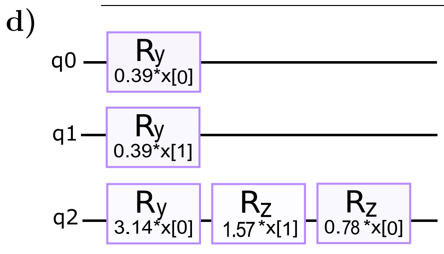


Figure 4. (a) Dataset composed by 150 points with a non-linear pattern and a binary target. (b) Hyperparameters used to optimize the QSVM circuit with our genetic algorithm. (c) Structure of circuits created in the first generations of the genetic algorithm. (d) Best ansatz resultant in the evolution with 1.0 accuracy.

maximising the accuracy. As illustrated in figure 4(b), we optimize the circuits over 5000 generations, using a population of 100 individuals.

The initial circuits make use of all available qubits and all layers, as shown in figure 4(c). Already in these circuits we observe the penalty associated to CNOT gates decreasing the number of entangling unitaries, as compared to other ansätze in the literature. More interestingly, the *Pareto front* combined with the elitist strategies is capable of further realizing that no entanglement at all is required to fit this model. Thus, after 5000 generations, the algorithm produces the simple uncorrelated circuit from figure 4(d), which fits the test set with perfect accuracy.

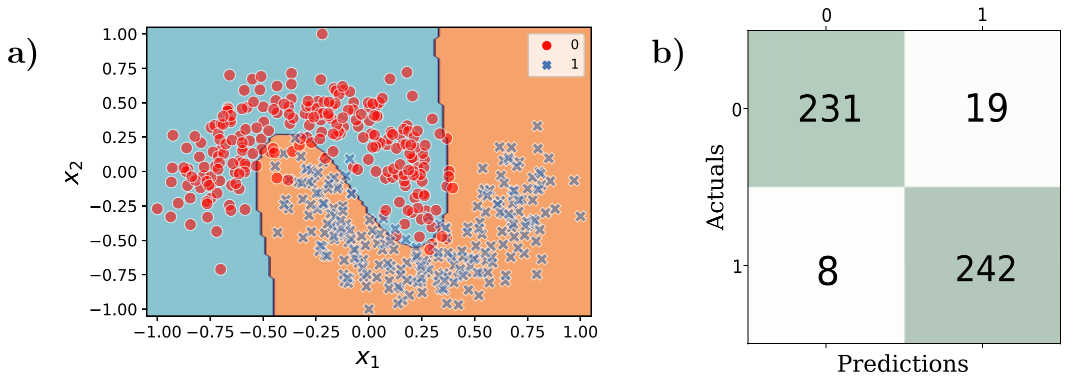


Figure 5. (a) Validation dataset, together with the predictions and decision boundary from the generated model. (b) Confusion matrix produced by the application of the QSVM model onto the validation dataset.

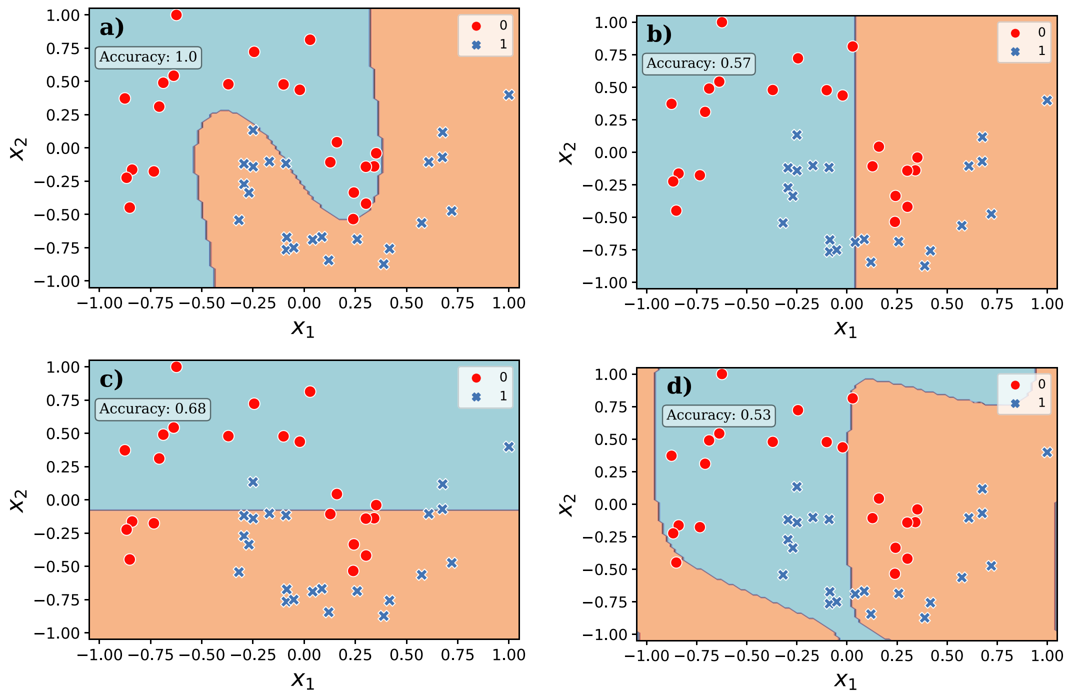


Figure 6. (a) Data points and prediction boundaries from the full quantum kernel SVM. (b), (c) and (d) are the decision boundaries provided by the circuits on the first, second and third qubit, respectively.

The fact that the generated model has perfect accuracy is useless, if it cannot generalize to other data from the same distribution. Once the training-evolution process has finished, we *validate* the utility of the model using additional datapoints, a *validation set*, with 500 points generated by the same synthetic algorithm. The same scaling preprocessing step $[-1, +1]$ that is applied to the training data is also applied to these validation datapoints. Figure 5(a) shows both the validation dataset used and the predictions made by the quantum support vectorial, defined by the decision boundary. Figure 5(b) also illustrates the confusion matrix of this validation process, considering both real and predicted labels, and identifying the incorrectly classified data. The confusion matrix allows us to conclude that the QSVM extrapolates to unseen data with same distribution, because 473 out of 500 data in the dataset have been correctly classified. In other words, a 94.6% of correct classified data or 0.946 accuracy.

4.2. Interpretability

As we see above and will see in later examples, the strong penalty on entangling gates makes the genetic algorithm prefer circuits that have smaller clusters of uncorrelated qubits. Ideally, only the gates that are essential for the modelization are included. The result is a circuit that can be decomposed as a tensor product of separate unitaries, and a quantum kernel that is a scalar product of separate kernels, as in $K(\mathbf{x}, \mathbf{x}') = \prod_{i=1}^{m} K_i(\mathbf{x}, \mathbf{x}')$. Note that each separate kernel $K_i(\mathbf{x}, \mathbf{x}')$ may actually depend only on a subset of

Table 1. Results from applying the genetic engineering of QSVM to other model problems in supervised machine learning. We also provide the accuracy of other classical methods for supervised machine learning: a k-NN, a linear SVM, and an SVM with a polynomial kernel of degree 2 (poly).

	Parkinson [37]	IoT irrigation [38]	Drug classification [39]
Circuit	Figure 7(a)	Figure 7(b)	Figure 7(c)
Accuracy (*test*)	**1.0**	**1.0**	**1.0**
Generations	5000	1000	500
#Attributes	22	2	5
#Classes	2	2	5
Max qubits	8	5	5
Max depth	15	5	5
Mutation probability (p_{mut})	0.7	0.7	0.7
Mutation ind. prob. (p_{ind})	0.2	0.2	0.2
Crossover prob. (p_{cross})	0.3	0.3	0.3
k-NN accuracy	0.82	1.0	0.70
SVM (linear) accuracy	0.89	1.0	0.87
SVM (poly-2) accuracy	0.89	1.0	0.65

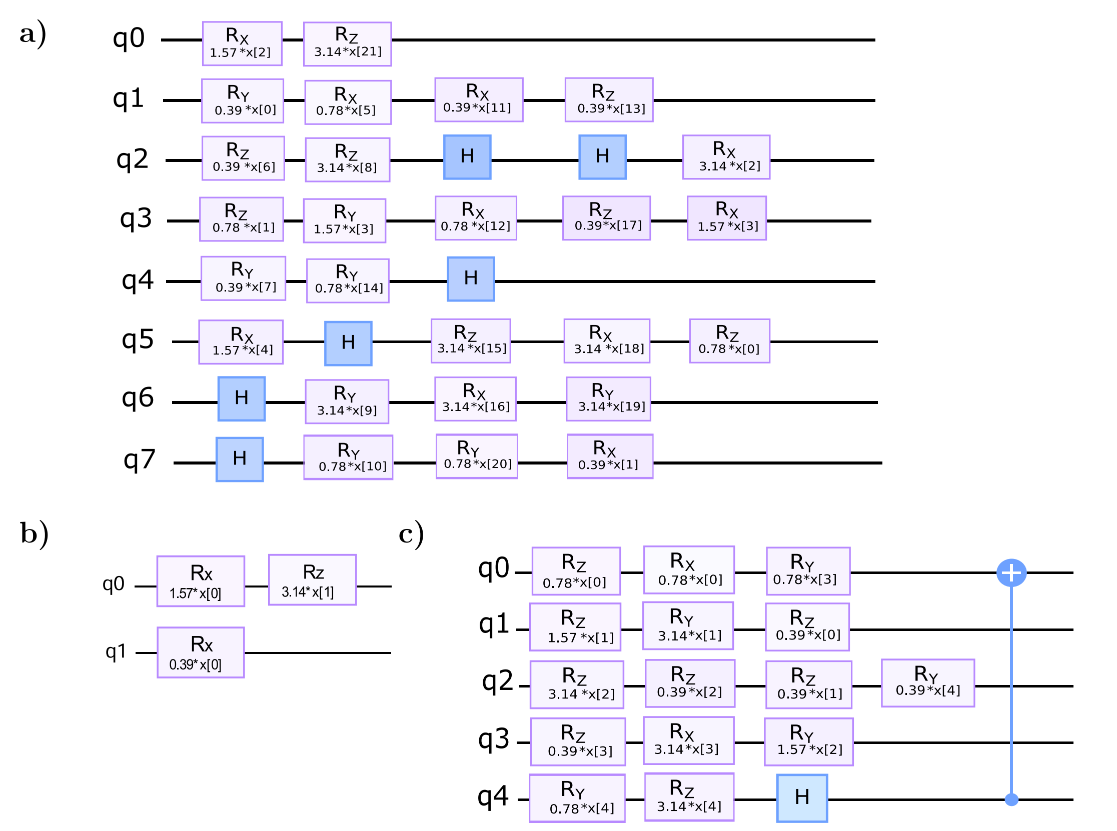

a)
q0 Rx 1.57*x[2] Rz 3.14*x[21]
q1 Ry 0.39*x[0] Rx 0.78*x[5] Rx 0.39*x[11] Rz 0.39*x[13]
q2 Rz 0.39*x[6] Rz 3.14*x[8] H H Rx 3.14*x[2]
q3 Rz 0.78*x[1] Ry 1.57*x[3] Rx 0.78*x[12] Rz 0.39*x[17] Rx 1.57*x[3]
q4 Ry 0.39*x[7] Ry 0.78*x[14] H
q5 Rx 1.57*x[4] H Rz 3.14*x[15] Rx 3.14*x[18] Rz 0.78*x[0]
q6 H Ry 3.14*x[9] Rx 3.14*x[16] Ry 3.14*x[19]
q7 H Ry 0.78*x[10] Ry 0.78*x[20] Rx 0.39*x[1]
b)
q0 Rx 1.57*x[0] Rz 3.14*x[1]
q1 Rx 0.39*x[0]
c)
q0 Rz 0.78*x[0] Rx 0.78*x[0] Ry 0.78*x[3]
q1 Rz 1.57*x[1] Ry 3.14*x[1] Rz 0.39*x[0]
q2 Rz 3.14*x[2] Rz 0.39*x[2] Rz 0.39*x[1] Ry 0.39*x[4]
q3 Rz 0.39*x[3] Rx 3.14*x[3] Ry 1.57*x[2]
q4 Ry 0.78*x[4] Rz 3.14*x[4] H

Figure 7. Best individuals provided automatically from this technique for each supervised learning problem in table 1. (a) Parkinson problem [37], (b) IoT irrigation [38] and (c) drug classification [39].

the features, or a combination thereof. We therefore suggest to study the classification induced by each kernel K_i separately and by their combination, as a strategy to provide *interpretations* of the rules that the evolutionary strategy has produced.

An example of this study is performed in figure 6 for our synthetic model. Figure 6(a) illustrates the boundaries of the complete kernel, which has accuracy 1.0, while later figures 6(b)–(d) show the boundaries induced by each separate kernel. As we can see, both qubits one and two, provide linear hyperplanes, being qubit three the one that provides a degree of non-linearity achieved by applying three rotations. Finally, the combination of these qubits forms the desired non-linear pattern. Interestingly, the single-qubit boundaries have a lower classification accuracy, of 0.57, 0.68 and 0.53, respectively, but their nonlinear combination in the final kernel gives the right predictions.

4.3. Other use cases

We have applied the method to other problems that are standard benchmarks for classical supervised learning techniques. Table 1 lists three problems, with the characteristics of the datasets, the hyperparameters of the genetic algorithm and the resulting accuracy in the test set. As seen in our experiments, the technique performs well for datasets with a high number of features as well as for datasets with more than two classes. The comparison with non-quantum classification methods also shows an advantage of the QSVM technique after making same preprocessing and split to data, specially in the highly complex multiclass classification of drugs which is treated with *one*-vs-*all* algorithm.

Figure 7 illustrates the structure and parameterization of the quantum feature maps that optimally classify these benchmarks. Interestingly, two of the circuits are uncorrelated and have no CNOT gates, while the third one, for multiclass drug classification, has just one entangler gate. This is relevant for several reasons. First, it illustrates the power of individual qubits a quantum classifiers, a realization already introduced in reference [40]. Second, the structures we have obtained, having little or no correlation, admit an efficient classical simulation which constitutes in itself a type of *quantum-inspired* machine-learning technique.

5. Summary and outlook

In this work we have explored the global optimization of quantum feature maps in a quantum kernel SVM algorithm using evolutionary multiobjective algorithms. The feature map is built as a parameterized quantum circuit that depends on the input data. The genetic algorithm stored the structure of the circuit, the actual gates and the functional dependence on the data as a string of binary-encoded genes. The algorithm evolves a population of individuals with genetic operators that seek to maximize the accuracy of these feature maps in modeling the data, while minimizing the complexity of the circuit. This is implemented using a nonlinear fitness function that combines both goals, and simultaneously applying a Pareto front selection strategy for the individuals.

We have applied this algorithm both to synthetic and to realistic benchmarks in the field of supervised machine learning, both single- and multiclass classification. The algorithm produces 100% accurate classifiers that can still generalize to unseen data since this metric is obtained from test sets. Moreover, the classifiers have a simple structure, with minimal or no correlations, which still capture the underlying nonlinear patterns. We attribute the simplicity of these circuits to the classification power of single qubits and single-qubit operations [40], which is enhanced by the combination of multiple parallel circuits. We believe that the resulting circuits are amenable to further interpretation strategies, in a simpler way than neural networks or other ML ansätze. Moreover, our results suggest the power of product states as another quantum-inspired variational strategy for supervised learning.

Our work leaves many avenues for exploration. The gene encoding that we have implemented contains a minimalistic set of entangling, local and parameterized gates, with sufficient precision for the problems we have explored. This can be extended in various ways, such as enlarging the set of weights in the parameterization, changing the order in which parameters appear in the circuit, including also more local and entangling gates, including free parameters θ_i that can be optimized using SPSA or other strategies, etc. If we focus on entanglement-free ansätze, we also find a rich avenue to explore the implementation of these models as standalone tools for machine learning, or developing a more clear strategy for the interpretation of the resulting classifiers—e.g. developing a kind of rule-based explanation of the model.

Acknowledgments

The authors gratefully acknowledges the computer resources at Artemisa, funded by the European Union ERDF and Comunitat Valenciana as well as the technical support provided by the Instituto de Física Corpuscular, IFIC (CSIC-UV). This work has been supported by Spanish Project PGC2018-094792-B-100 (MCIU/AEI/FEDER, EU), CAM/FEDER Project No. S2018/TCS-4342 (QUITEMAD-CM), and CSIC Platform PTI-001.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI.

ORCID iDs

Sergio Altares-López https://orcid.org/0000-0002-0847-6113
Angela Ribeiro https://orcid.org/0000-0001-5807-8132
Juan José García-Ripoll https://orcid.org/0000-0001-8993-4624

References

[1] Biamonte J, Wittek P, Pancotti N, Rebentrost P, Wiebe N and Lloyd S 2017 *Nature* **549** 195–202
[2] Schuld M and Petruccione F 2018 *Supervised Learning with Quantum Computers* (Berlin: Springer)
[3] Kerenidis I, Landman J, Luongo A and Prakash A 2018 q-means: a quantum algorithm for unsupervised machine learning (arXiv:1812.03584)
[4] Liu N and Rebentrost P 2018 *Phys. Rev.* A **97** 042315
[5] Lloyd S, Mohseni M and Rebentrost P 2014 *Nat. Phys.* **10** 631–3
[6] Cong I and Duan L 2016 *New J. Phys.* **18** 073011
[7] Duan B, Yuan J, Xu J and Li D 2019 *Phys. Rev.* A **99** 032311
[8] Rebentrost P, Mohseni M and Lloyd S 2014 *Phys. Rev. Lett.* **113** 130503
[9] Benedetti M, Lloyd E, Sack S and Fiorentini M 2019 *Quantum Sci. Technol.* **4** 043001
[10] Romero J, Olson J P and Aspuru-Guzik A 2017 *Quantum Sci. Technol.* **2** 045001
[11] Schuld M and Killoran N 2019 *Phys. Rev. Lett.* **122** 040504
[12] Havlíček V, Córcoles A D, Temme K, Harrow A W, Kandala A, Chow J M and Gambetta J M 2019 *Nature* **567** 209–12
[13] Dallaire-Demers P L and Killoran N 2018 *Phys. Rev.* A **98** 012324
[14] Lloyd S and Weedbrook C 2018 *Phys. Rev. Lett.* **121** 040502
[15] Du Y, Hsieh M H, Liu T and Tao D 2020 *Phys. Rev. Res.* **2** 033125
[16] Sim S, Johnson P D and Aspuru-Guzik A 2019 *Adv. Quantum Tech.* **2** 1900070
[17] McClean J R, Boixo S, Smelyanskiy V N, Babbush R and Neven H 2018 *Nat. Commun.* **9** 4812
[18] Holmes Z, Sharma K, Cerezo M and Coles P J 2021 arXiv:2101.02138
[19] Grant E, Wossnig L, Ostaszewski M and Benedetti M 2019 *Quantum* **3** 214
[20] Sim S, Romero J, Gonthier J F and Kunitsa A A 2021 *Quantum Sci. Technol.* **6** 025019
[21] González F A, Gallego A, Toledo-Cortés S and Vargas-Calderón V 2021 arXiv:2102.04394
[22] Ostaszewski M, Grant E and Benedetti M 2021 *Quantum* **5** 391
[23] Li R, Alvarez-Rodriguez U, Lamata L and Solano E 2017 *Quantum Meas. Quantum Metrol.* **4** 1–7
[24] Lamata L, Alvarez-Rodriguez U, Martín-Guerrero J D, Sanz M and Solano E 2018 *Quantum Sci. Technol.* **4** 014007
[25] Chivilikhin D, Samarin A, Ulyantsev V, Iorsh I, Oganov A R and Kyriienko O 2020 MoG-VQE: multiobjective genetic variational quantum eigensolver (arXiv:2007.04424)
[26] Zhao T, Carleo G, Stokes J and Veerapaneni S 2020 *Mach. Learn.: Sci. Technol.* **2** 02LT01
[27] Anand A, Degroote M and Aspuru-Guzik A 2021 *Mach. Learn.: Sci. Technol.* **2** 045012
[28] Rattew A G, Hu S, Pistoia M, Chen R and Wood S 2019 arXiv:1910.09694
[29] Bilkis M, Cerezo M, Verdon G, Coles P J and Cincio L 2021 arXiv:2103.06712
[30] Zhao N, Wu Z, Zhao Y and Quan T 2010 *Expert Syst. Appl.* **37** 4805–10
[31] Zhou X, He X and Chen B 2002 *Tsinghua Sci. Technol.* **7** 28–31
[32] Géron A 2019 *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* (Massachusetts: O'Reilly Media)
[33] Schuld M 2021 Quantum machine learning models are kernel methods (arXiv:2101.11020)
[34] Barán B, Carballude A and Villagra M 2021 *SN Comput. Sci.* **2** 19
[35] Gutiérrez Reina D, Tapia Córdoba A and Rodríguez Del Nozal A 2020 *Algoritmos Genéticos con Python* (Spain: Alfaomega Marcombo)
[36] Thirion B, Varoquaux G, Gramfort A, Michel V, Grisel O, Louppe G and Nothman J 2011 Scikit-datasets (generate samples of synthetic data sets) https://github.com/scikit-learn/scikit-learn
[37] Little M, McSharry P, Roberts S, Costello D and Moroz I 2007 *Biomed. Eng. Online* **6** 23
[38] Patel H 2020 Intelligent irrigation system (by using temperature and moisture data) https://kaggle.com/harshilpatel355/autoirrigationdata
[39] Tripathi P 2020 Drug classification dataset https://kaggle.com/prathamtripathi/drug-classification
[40] Pérez-Salinas A, López-Núñez D, García-Sáez A, Forn-Díaz P and Latorre J I 2021 One qubit as a universal approximant (arXiv:2102.04032)