Quantum Machine Intelligence (2024) 6:45
https://doi.org/10.1007/s42484-024-00177-w

RESEARCH ARTICLE

Hybrid genetic optimization for quantum feature map design

Rowan Pellow-Jarman[1,2] · Anban Pillay[1,3] · Ilya Sinayskiy[2,4] · Francesco Petruccione[2,4,5]

Received: 22 February 2023 / Accepted: 27 June 2024

Abstract
Kernel methods are an import class of techniques in machine learning. To be effective, good feature maps are crucial for mapping non-linearly separable input data into a higher dimensional (feature) space, thus allowing the data to be linearly separable in feature space. Previous work has shown that quantum feature map design can be automated for a given dataset using NSGA-II, a genetic algorithm, while both minimizing circuit size and maximizing classification accuracy. However, the evaluation of the accuracy achieved by a candidate feature map is costly. In this work, we demonstrate the suitability of kernel-target alignment as a substitute for accuracy in genetic algorithm-based quantum feature map design. Kernel-target alignment is faster to evaluate than accuracy and does not require some data points to be reserved for its evaluation. To further accelerate the evaluation of genetic fitness, we provide a method to approximate kernel-target alignment. To improve kernel-target alignment and root mean squared error, the final trainable parameters of the generated circuits are further trained using COBYLA to determine whether a hybrid approach applying conventional circuit parameter training can easily complement the genetic structure optimization approach. A total of eight new approaches are compared to the original across nine varied binary classification problems from the UCI machine learning repository, showing that kernel-target alignment and its approximation produce feature map circuits enabling comparable accuracy to the previous work but with larger margins on training data (in excess of 20% larger) that improve further with circuit parameter training.

Keywords Quantum computing · Machine learning · SVM · QSVM · Genetic algorithm · NSGA-II · Kernel-target alignment

1 Introduction

Quantum computers leverage quantum properties such as entanglement and promise the potential of a speed advantage over classical algorithms when applied to specialized problems. Some algorithms such as Shor's algorithm for factorization (Shor 1997) and Grover's search algorithm (Grover 1996) have been shown theoretically to outperform all known classical algorithms applied to the same tasks. Quantum algorithm design is made difficult by the unintuitive nature of quantum entanglement which must be used effectively to achieve an advantage over classical algorithms. Quantum machine learning seeks to apply quantum computation to machine learning tasks to achieve a quantum advantage over classical machine learning.

Quantum machine learning and classical machine learning show promise for automating many practical tasks that would otherwise require human intelligence, including disease diagnosis (Myszczynska et al. 2020), natural language processing (Khurana et al. 2022), and image classification (Horak and Sablatnig 2019). Machine learning algorithms are designed to learn functions from data (Good-

✉ Rowan Pellow-Jarman
rowanpellow@gmail.com

Anban Pillay
pillayw4@ukzn.ac.za

Ilya Sinayskiy
sinayskiy@ukzn.ac.za

Francesco Petruccione
petruccione@sun.ac.za

1 School of Mathematics, Statistics & Computer Science, University of KwaZulu-Natal (Westville Campus), Durban 4001, South Africa

2 National Institute for Theoretical and Computational Sciences (NITheCS), Stellenbosch 7600, South Africa

3 Centre for Artificial Intelligence Research (CAIR), Cape Town, South Africa

4 School of Chemistry and Physics, University of KwaZulu-Natal (Westville Campus), Durban 4001, South Africa

5 School of Data Science and Computational Thinking and Department of Physics, Stellenbosch University, Cape Town 7600, South Africa

Published online: 15 July 2024

fellow et al. 2016). These algorithms can be separated into three categories: supervised learning, unsupervised learning, and reinforcement learning. Supervised learning algorithms make use of prelabeled training data, while unsupervised learning algorithms do not (Alloghani et al. 2020). In reinforcement learning, agents learn behavior by interacting with an environment that provides rewards and punishments to guide them.

Kernel methods are an important class of techniques in both classical and quantum machine learning. The Support Vector Machine (SVM) (Boser et al. 1996) is an important classical supervised kernel method, due to its theoretical relations to other learning models (Jacot et al. 2018; Schuld 2021) and results regarding its generalization ability (Vapnik 1998; Vapnik and Chervonenkis 2015). It is built around an optimization algorithm for finding an optimal linear hyperplane that separates data points into two classes. The hyperplane is selected to maximize the minimum margin of any point in the training dataset, where the margin of a point is defined as the distance of the point from the separating hyperplane. Larger margin sizes have been theoretically linked to improved generalization performance (Vapnik 1998; Vapnik and Chervonenkis 2015). Non-linear decision boundaries can be achieved by mapping non-linear data to a higher dimensional feature space. The mapping function used is called a feature map, and the range of a feature map is called a feature space. By use of a technique known as the *"kernel trick"*, the decision boundary optimization problem can be reformulated in terms of a kernel function that computes the similarity of a pair of data points in the feature space (Boser et al. 1996). This obviates the need to explicitly compute feature map outputs for data points, so long as the corresponding kernel function can be computed.

The feature map must be carefully selected for effective separation of the data. For any kernel function and labeled training set combination, a quantity known as the kernel-target alignment of the kernel can be calculated. This indicates the degree of agreement between the kernel function and a hypothetical oracle kernel induced by the training labels that is well suited to the training data (Cristianini et al. 2001). A high kernel-target alignment has been shown in other works to correlate with improved classification performance (Cristianini et al. 2001), and it has been proposed for use as a metric for selecting suitable kernels for a dataset in classification problems (Cristianini et al. 2001; Hubregtsen et al. 2022).

The QSVM algorithm enhances the SVM algorithm by implementing the feature map function as a quantum circuit (see Section 3.1.2). While quantum feature map circuits are parameterized in the feature values of a single data point, they can also contain additional trainable parameters which can be optimized to improve the suitability of the kernel circuit for a specific dataset (Hubregtsen et al. 2022). A quantum circuit containing trainable parameters is called an *ansatz*. Prior work has used classical optimizers on trainable parameterized quantum kernels to maximize their kernel-target alignment, which resulted in positive effects on the classification accuracy of the resulting SVM models (Hubregtsen et al. 2022).

2 Related works

Quantum feature map circuits of fixed structure that make use of trainable parameter values are reported in Hubregtsen et al. (2022). The trainable parameter values are optimized using Stochastic Gradient Ascent to maximize the kernel-target alignment of the corresponding kernel functions. This was performed to test whether kernel-target alignment optimization could improve the performance of a fixed-structure quantum feature map on a given dataset. Increased kernel-target alignment had previously been shown in Cristianini et al. (2001) to correlate with improved classification ability. The technique described can be applied either to tailor an existing feature map to a dataset or to fully generate a feature map for a dataset from a feature map ansatz that has a predetermined circuit structure and parameter placement. The work made use of both classical noise-free simulations of quantum computers and real NISQ computers to run quantum circuits, reporting improvements in classification accuracy after kernel-target alignment maximization (Hubregtsen et al. 2022).

A second model training metric applicable to quantum kernel classifiers is a classifier's root mean squared error (RMSE). This is often optimized to train parameterized models for classification tasks. It can be better suited to training than direct evaluation of accuracy since it accounts for the magnitudes of misclassification errors rather than simply the number of errors that occur. Accuracy can be too insensitive to circuit parameter changes to show when a circuit has slightly improved if a data set is not sufficiently large.

Another approach to optimizing the choice of kernel circuit for a dataset is to optimize the selection of circuit gates used in the feature map in addition to the values of trainable circuit parameters. This approach has been applied to optimizing circuits applied to other problems (Ostaszewski et al. 2021). Genetically inspired algorithms have also been applied to circuit structure optimization since they are capable of combinatorial optimization (Lukac and Perkowski 2002; Bautu and Bautu 2007; Rasconi and Oddi 2019).

Altares-López et al. (2021) detailed the implementation of a genetic algorithm for automated feature map circuit design for use with QSVM classifiers that both maximizes classification accuracy and minimizes circuit size. The optimization was performed using a variation of genetic algorithm named NSGA-II (Deb et al. 2002) which customizes the usual

genetic selection and fitness evaluation operations of the genetic algorithm (as outlined in Sect. 3.3) to evaluate multiple fitness functions and preferentially select non-dominated solutions for crossover.

In a minimization problem with two fitness functions, a solution s with fitness values (a, b) is considered non-dominated with respect to another set of n solutions with fitness values $\{(f_i, g_i)|i \in \{1, 2, ..., n\}\}$ if and only if

$$\forall i \in \{1, 2, ..., n\}, (f_i < a) \implies (g_i > b)$$

, i.e., if and only if there are no solutions in the set with all fitness values superior to the corresponding fitness values of s. NSGA-II also makes use of elitism to guarantee the preservation of the solutions that best optimize at least 1 of the individual fitness functions.

In order to apply a genetic algorithm to quantum feature map design, the work also defined a binary string representation for encoding feature map circuits. The encoding strategy can be found in Altares-López et al. (2021) and is summarized here.

Each circuit gate is encoded in a sequence of 5 bits, the first three of which encode the type of gate applied and the last two of which encode a proportionality parameter for use in the case that a parameterized rotation gate was selected by the first three bits. The mapping of bits to gates and proportionality parameters is shown in Table 1.

Table 1 Mapping used on each consecutive 5-bit sequence of bits encoding a feature map gate to determine the gate type and a proportionality parameter value used in the case of parameterized gates

Bits	Gate	Bits	Parameter
000	H	00	π
001	CNOT	01	$\pi/2$
010	I	10	$\pi/4$
011	Rx	11	$\pi/8$
100	Rz		
101	I		
110	I		
111	Ry		

The available gates for the encoding to select from are Hadamard (H), CNOT, identity (I), and parameterized rotations around the X, Y, and Z axis of the Bloch sphere which are used to encode data point feature values into the circuit. We define the $R_x(\theta)$ gate as $\cos(\theta/2)1 - i\sin(\theta/2)X$, the $R_y(\theta)$ gate as $\cos(\theta/2)1 - i\sin(\theta/2)Y$, and the $R_z(\theta)$ gate as $\cos(\theta/2)1 - i\sin(\theta/2)Z$. If a parameterized rotation gate R_a around axis a is selected, the parameter selected by the last two of the five gate bits will be used to select a proportionality parameter p. When the gate R_a is applied to encode a feature value x_i, the gate applied will be $R_a(px_i)$. In the case of a CNOT gate being selected and this gate being applied to qubit i, qubit i will be used as the control qubit and qubit $(i + 1) \mod M$ will be used as the target qubit

Hyperparameters M and N which designate the maximum number of qubits and maximum number of gate layers respectively must be chosen before applying the genetic algorithm and are fixed for the duration of the genetic optimization. A single solution is represented as a bit string of length $5MN$, which holds the concatenation of the encodings of the individual gates in the feature map.

The gates in the encoding are applied successively with the target qubit and feature value to potentially encode being selected in a round-robin fashion. Stated explicitly, for each consecutive group of 5 gate bits, the ith gate description will be applied to qubit $i \mod M$ and will encode feature $i \mod N$ if it performs a parameterized rotation. This encoding strategy was selected for simplicity, although other strategies could also feasibly be attempted.

Two fitness functions were optimized in the work: accuracy on a test set was maximized and a weighted size metric was simultaneously minimized. The unweighted size metric SM was calculated in terms of the number of qubits M, the number of single qubit gates N_{local}, and the number of entangling gates N_{CNOT}, by the expression

$$\text{SM} = \frac{N_{\text{local}} + 2N_{\text{CNOT}}}{M}.$$

The weighted size metric WS supplied to the genetic algorithm was given by the expression

$$\text{WS} = \text{SM} + \text{SM} \cdot \text{accuracy}^2.$$

The work was able to demonstrate the effectiveness of using NSGA-II with the devised feature map binary string encoding strategy, accuracy maximization, and weighted size minimization to automatically produce quantum feature map circuits for QSVM classification using only a dataset and a few hyperparameters as input. The generated circuits were also experimentally shown to generalize to unseen data. In addition to using few qubits and quantum gates, the circuits produced by the approach were observed to make little to no use of entanglement, meaning they could be efficiently simulated classically and the approach could constitute a quantum-inspired classical machine learning algorithm.

Some attempts have been made to enhance the genetic optimization and compare the approach to others. The work done in Chen and Chern (2022) is based on the algorithm put forward in Altares-López et al. (2021). They used a modified encoding scheme which encoded the proportionality parameter values using three bits instead of only two, doubling the number of encodable parameter values. A restricted choice of parameter values was one of the potential limitations of the algorithm designed in Altares-López et al. (2021). The algorithm was also further modified to optimize gate cost and classification accuracy in a single objective expression,

using a single objective genetic algorithm instead of a multi-objective genetic algorithm such as NSGA-II. Notably, this introduced a new hyperparameter used to weight the focus of the optimization between circuit size and accuracy, which is not done with NSGA-II.

The feature map generated by the genetic algorithm was compared with two other choices of ansatz: a hardware efficient ansatz proposed in Kandala et al. (2017) and a unitary decomposition ansatz proposed in Shende et al. (2006). Each ansatz was trained with COBYLA (Powell 1994), directly evaluating classification accuracy as a cost function. It was found that the feature map circuits generated by the genetic algorithm variation used in the work performed similarly to the hardware efficient ansatz, depending on the circuit depth hyperparameter selected when generating the hardware efficient ansatz. However, both were beaten by the unitary decomposition ansatz in accuracy, which achieved the highest accuracy at the cost of having a large, fixed size.

Our work investigates kernel-target alignment as a metric for automating quantum feature map design for the Quantum-Enhanced Support Vector Machine (QSVM) algorithm (Schuld and Killoran 2019; Rebentrost et al. 2013). Our work has two goals: investigate the suitability of using alternative cost functions to accuracy in the genetic optimization process described in Altares-López et al. (2021) and, secondly, investigate whether the problem of limited circuit parameter choices in the genetic algorithm can be addressed by a hybrid process of genetic and circuit parameter training.

We address the first goal by evaluating two alternative cost functions to the accuracy metric in Altares-López et al. (2021): firstly, kernel-target alignment for the genetic optimization step and, secondly, a heuristic estimation of the kernel-target alignment performing a fraction of the kernel evaluations. For the second goal, a hybrid method involving further optimizing the final choice of trainable circuit parameter values after the genetic algorithm terminates for each of the above approaches is evaluated. This final optimization uses COBYLA (Powell 1994) to maximize either kernel-target alignment or RMSE. The new approaches are compared to the original across several binary classification problems of varying difficulties. We show that even though the kernel-target alignment metric is less computationally expensive to compute in terms of quantum kernel evaluations and avoids the training of an SVM classifier, the performance of the constructed classifiers is comparable to the original approach and often achieves a better margin distribution on training data. It has been demonstrated theoretically that increased margin sizes indicates better generalization ability (Vapnik 1998; Vapnik and Chervonenkis 2015). The kernel-target alignment approximation heuristic is shown to perform marginally worse than exact kernel-target alignment optimization but at a fraction of the computational cost. The hybrid approaches are shown to improve margin sizes over the original. The original approach is also shown to sometimes overfit to the test data used to evaluate its accuracy metric, particularly on difficult problems.

In the following section, we give a more detailed explanation of the background topics involved in understanding this work and related works and explain our experimental setup. This is followed by a section covering our findings and interpretations. In the final section, we give an overview of the contributions made and suggest ideas for further research.

3 Methods

3.1 Binary classifiers using quantum kernels

3.1.1 Support Vector Machine (SVM)

The SVM algorithm is a classical supervised machine learning algorithm for binary classification problems that works by finding an optimal separating hyperplane between two classes of data points. The SVM algorithm is applicable when the data points can be represented by real-valued feature vectors. For simplifying definitions, the class labels are usually replaced with positive and negative one.

The *margin* of a single data point is defined as the distance from the data point to the SVM's chosen hyperplane. The margin of an SVM classifier refers to the minimum of the margins of the data points. The data points with minimum margin are known as the *support vectors*. The hyperplane chosen by the SVM is optimal in the respect that it maximizes the minimum of the margins of the training set data points by solving a quadratic programming optimization problem.

The SVM algorithm is also capable of classifying datasets with classes that are not linearly separable. This can be achieved by first mapping the data points to a higher dimensional space in such a way that they become linearly separable in the higher dimensional space (see Fig. 1 for an illustration). A function used to perform this mapping is called a *feature map*, and the range of the function is called the *feature space*. The choice of feature map must be suited to the dataset in order to classify it well, since it determines whether the data will become linearly separable after transformation.

A feature map $\phi(x)$ that maps a point into feature space has a corresponding *kernel function*

$$\kappa(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$$

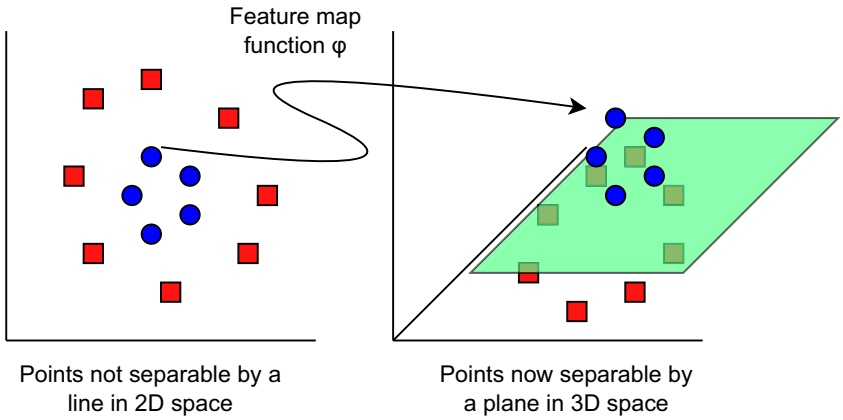


Fig. 1 An example illustrating how a feature map function could be used to make non-linearly separable points linearly separable in a higher dimensional space. In this case, the feature map could be implemented as a function that adds a third dimension to the points with decreasing value as distance from the central region of the points increases

which computes the inner product of a pair of data points in the feature space. The margin optimization problem can be equivalently reformulated as a dual problem in terms of the kernel function (Boser et al. 1996), which can sometimes avoid the explicit computation of the feature map. This advantage is often referred to as the *"kernel trick"*. Seeing that in many cases where a feature map can not be efficiently computed but the corresponding kernel function can be, the dual formulation of the problem increases the number of potential feature maps that can be applied to a dataset.

In the dual form of the SVM, the classifier output for a given class is determined using a coefficient sequence $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_n\}$ and offset sequence $b = \{b_1, b_2, ..., b_n\}$ chosen by the SVM algorithm during the hyperplane optimization. The decision function df outputs an indication of the distance of its input point from the hyperplane after mapping into feature space. It is defined in terms of the kernel function κ, the training samples $\{x_1, x_2, ..., x_n\}$, the α coefficients, and the b offsets as follows:

$$\mathrm{df}(x) = \sum_{i=1}^{n} (\alpha_i \kappa(x, x_i) + b_i).$$

The sign of $\mathrm{df}(x)$ is used to determine the predicted class of the argument point x:

$$\mathrm{Class}(x) = \mathrm{sign}(\mathrm{df}(x)).$$

3.1.2 Quantum-enhanced support vector machine

The Quantum-enhanced Support Vector Machine (QSVM) algorithm extends the SVM algorithm by performing the kernel computation on a quantum computer (Schuld and Killoran 2019; Rebentrost et al. 2013). A quantum circuit parameterized in the values of a single data point is used as a feature map to map the data points to a high dimensional quantum state in a quantum Hilbert space.

For a quantum feature map encoding data points into q qubits, the dimensionality of the feature Hilbert space is 2^q. Although a quantum computer can efficiently compute the quantum state feature space representation of data point, in general, the quantum state cannot be efficiently represented classically due to the exponentially increasing dimensionality of the feature space. To work around this classical limitation, the kernel function is computed directly on the quantum computer and the kernel-based formulation of the SVM is used. The kernel computation for a pair of data points can be efficiently performed by measuring the overlap of their corresponding states in the quantum feature space.

To train a QSVM model, the *Gram matrix* $K_{n\times n}$ of the training points must be computed. For n training points $\{x_1, x_2, ..., x_n\}$ and a quantum kernel function κ, the Gram matrix is defined by $K_{ij} = \kappa(x_i, x_j)$ where $i, j \in \{1, 2, ..., n\}$. In the case of a noise free quantum computer or simulator being used to execute κ, the symmetric property $\kappa(x_i, x_j) = \kappa(x_j, x_i)$ and the property that $\kappa(x_i, x_i) = 1$ can be used to reduce the number of required evaluations.

Assuming the stated properties, the n main diagonal entries of K (K_{11}, K_{22},..., K_{nn}) do not require kernel evaluations, since $K_{ii} = \kappa(x_i, x_i) = 1$. For the remaining $n^2 - n$ entries K_{ij} which are not on the main diagonal, there is a symmetric entry K_{ji} with the same value, since

$$K_{ij} = \kappa(x_i, x_j) = \kappa(x_j, x_i) = K_{ji}$$

.

This means that only half of the entries need to be explicitly computed by kernel evaluations. In effect, only $\frac{n^2-n}{2}$ kernel evaluations must be performed to construct K. In the case of a NISQ computer, this technique could potentially be

applied if measures were taken to mitigate noise and correct the kernel matrix such as in Hubregtsen et al. (2022).

Since the only difference between the SVM and QSVM algorithms is how the kernel computation is performed, the potential advantage of the QSVM algorithm lies in enabling the computation of kernel functions that are hard to estimate classically (Liu et al. 2021). While examples of such kernels have been discovered (Liu et al. 2021) for artificial datasets, it is an open question how best to design quantum feature maps to achieve a useful kernel with a quantum speed advantage.

3.2 Kernel quality metrics

3.2.1 Kernel-target alignment

Kernel-target alignment is a heuristic for kernel quality that measures the degree of similarity between two kernels or the degree of agreement between a kernel and a dataset (Cristianini et al. 2001). It is calculated using a matrix inner product between the Gram matrix constructed from the training samples and an oracle matrix constructed from the training labels, where the oracle matrix acts as a stand-in for the Gram matrix of a hypothetical kernel which is very well-suited to the data.

For a set of training points $\{x_1, x_2, ..., x_n\}$ with corresponding labels $\{y_1, y_2, ..., y_n\}$ with $\forall i, y_i \in \{-1, 1\}$, a kernel function $\kappa(x_i, x_j)$, and with the *Frobenius* inner product for matrices defined as

$$\langle A, B\rangle_F = \sum_{i,j} A_{ij} B_{ij},$$

the kernel-target alignment can be computed as follows (Cristianini et al. 2001):

1. Compute the Gram matrix $K_{n\times n}$ using the kernel function and training points by the rule

   $$K_{ij} = \kappa(x_i, x_j).$$

2. Compute the oracle matrix $O_{n\times n}$ using the training labels by the rule

   $$O_{ij} = y_i y_j.$$

3. Compute the kernel-target alignment KTA using the Frobenius inner product as

   $$\text{KTA} = \frac{\langle K, O\rangle_F}{\sqrt{\langle K, K\rangle_F \langle O, O\rangle_F}}.$$

A high kernel-target alignment has been shown in other works to correlate with improved classification performance (Cristianini et al. 2001), and it has been proposed for use as a metric for selecting applicable kernels for a dataset in classification problems (Cristianini et al. 2001; Hubregtsen et al. 2022).

3.2.2 Root mean squared error

Root mean squared error (RMSE) is a common metric for measuring the error of a model. It is calculated as the square root of the mean of the squared errors of a classifier's predictions on each training set data point. In this work, the RMSE of a classifier is calculated using the errors of the decision function on training data with an adjustment to the error calculation. The adjustment is to account for there not being a definitively correct output of the SVM decision function for a given sample and label pair. The error is measured relative to a positive target decision function output m, which we set to one in this work.

We calculate the error for a decision function output a and training label b using the following rule:

$$\text{error}(a, b) = \begin{cases} (m - a) & \text{if } b = 1 \text{ and } a < m \\ (a - m) & \text{if } b = -1 \text{ and } a > -m \\ 0 & \text{otherwise} \end{cases}$$

This choice of error function means that only points not classified to the desired degree of confidence m contribute to the error calculation, and the errors of the considered points increase with distance from the target output.

For a set of training points $\{x_1, x_2, ..., x_n\}$ with corresponding labels $\{y_1, y_2, ..., y_n\}$ with $\forall i, y_i \in \{-1, 1\}$, the RMSE is calculated in terms of this adjusted error function and the decision function df by the following rule

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} \text{error}(\text{df}(x_i), y_i)^2}{n}}$$

3.3 Overview of genetic algorithms

Genetic algorithms are flexible metaheuristic algorithms inspired by the real-world evolutionary principles of natural selection, genetic inheritance, and random mutation. They are a popular choice of algorithm for optimizing complex objective functions in cases where algorithms known to produce a global optimum are unknown or infeasible.

Implementing a genetic algorithm first requires designing a solution representation on which genetic operations can be performed. The solution representation often (but not always) takes the form of a binary string, which is mapped by a problem-specific decoding function to a usable solution. A genetic algorithm manages a set of these solution representations, which is called a population. The initial population can

be a set of randomly generated solutions or chosen according to a problem-specific heuristic.

The optimization process is iterative and typically repeats until a suitable solution is found, a desired number of iterations has passed, or the rate of improvement of the solutions becomes low. In each iteration, genetically inspired operations are applied to the population to create a new replacement population. The population creation process involves fitness evaluation, selection, crossover, and mutation operations.

Fitness evaluation is performed by decoding a solution representation into a solution, then evaluating a numeric score of its suitability for the problem. This is performed for the entire population, after which point a selection operation is applied to select some solutions for crossover and mutation. It is important that better solutions are more likely to be selected for crossover, since this is the main mechanism driving improvement between generations. The solutions selected for crossover are referred to as *"parents"*.

The crossover operation is performed between two parent solutions to produce one or more new solutions, called child solutions. This is usually performed by taking a simple combination of the solution representations of the parents. In the case of a binary string solution representation, a simple crossover can be performed by combining two non-overlapping subsequences of the parents, taking a random number of bits from the first and the remainder from the corresponding positions in the second. A mutation operation can be applied to a child solution by randomly editing its representation by a small amount. This simulates the random mutation which occurs in real life and affects the diversity of available genetic material in the population.

A strategy often employed when determining which individuals will make up the next generation is to preserve the best performing of the solutions among the current generation and the newly created children. This is known as *elitism* and ensures that solutions can survive through multiple generations and potentially indefinitely, so long as they continue to outperform newer ones. This helps prevent regression of the achieved fitness due to chance as generations pass.

The general idea for a genetic algorithm is flexible enough that many variations and extensions of the discussed components have also been studied (Chahar et al. 2021).

3.4 Experiments

The algorithm for automated feature map design described in Altares-López et al. (2021) was reimplemented using the Julia programming language (Bezanson et al. 2017), the Yao quantum simulator framework (Luo et al. 2020), and the pymoo (Blank and Deb 2020) implementation of NSGA-II. All experiments were run with the maximum qubit count and feature map depth hyperparameters set to 6. The genetic algorithm population size was set to 100, with 15 new individuals being produced every generation. A total of 30% of the new individuals were produced by crossover; the rest were chosen randomly from the parents. In each generation, 70% of the population underwent mutation. When mutation occurred, 20% of the bits in the mutated solution were flipped. All experiments were run using a noise free quantum simulator provided by Yao.

Three configurations of the original algorithm were run on nine different datasets of varying difficulty (see Table 2) to compare their effectiveness. The first configuration max-

Table 2 Table showing the characteristics of the datasets and sample splits used

Dataset	Class -1	Class 1	Features (PCA)	Train	Test	Validation
Moons	Top left	Bottom right	2 (N/A)	210	90	500
Cancer	Benign	Malignant	30 (10)	210	90	124
Iris	Versicolor	Virginica	4 (N/A)	42	18	40
Digits	Eight	Nine	64 (10)	140	60	148
Circles	Outer	Inner	2 (N/A)	210	90	500
Random	Red	Blue	2 (N/A)	210	90	N/A
Voice	Acceptable	Unacceptable	309 (10)	28	12	44
SUSY	Background	Signal	18 (10)	210	90	500
SUSY reduced	Background	Signal	8 (N/A)	210	90	500

Not all points in the base datasets were used to ensure the sample split remained balanced in each of the sample sets. Other considerations in determining the data splits were experiment runtime while maintaining a sufficiently large ratio of test points to train points and a sufficiently large number of validation points. All datasets with more than 10 feature values were reduced to 10 features using principle component analysis (PCA). The moons, circles, and random datasets are artificial, with the moons and circles datasets being generated with Scikit-learn (Pedregosa et al. 2011). The rest of the datasets are sourced from the UCI Machine Learning Repository (Dua and Graff, 2017) either directly or indirectly through Scikit-learn (Pedregosa et al. 2011)

imized accuracy on a test set and minimized weighted size, as in the original work (Altares-López et al. 2021). The second configuration maximized kernel-target alignment on the training data, ignoring the test data, and minimized the unweighted size metric also defined in Altares-López et al. (2021). The third configuration maximized an approximation of kernel-target alignment on the training data and minimized the same unweighted size metric.

Before performing the genetic optimization, the datasets are split into three disjoint subsets, namely training data, testing data, and validation data. The training data is used to evaluate kernel-target alignment and its approximation, as well as train the QSVM model for a given feature map circuit. The testing data is only used to evaluate the accuracy metric in the first approach. The validation data is used to determine the generalization ability of the generated models and must be separate from the testing data since the first approach can indirectly access the testing data through the accuracy metric and potentially overfit to it.

In order to calculate the kernel-target alignment approximation for n training points, the n points are divided into a disjoint complementary subsets of size roughly n/a. The number of subsets a can be adjusted based on the number of training points to balance speed and precision. The kernel target alignment is calculated on each of the subsets in turn, then averaged.

Assuming the properties of the kernel function are not used to accelerate the Gram matrix computation, n^2 kernel evaluations are required to compute the exact kernel-target alignment, and only

$$a(n/a)^2 = (n^2)/a$$

evaluations are required to compute the approximation, giving a factor a speedup. If the kernel properties are used, then

$$(n^2 - n)/2 = n^2/2 - n/2$$

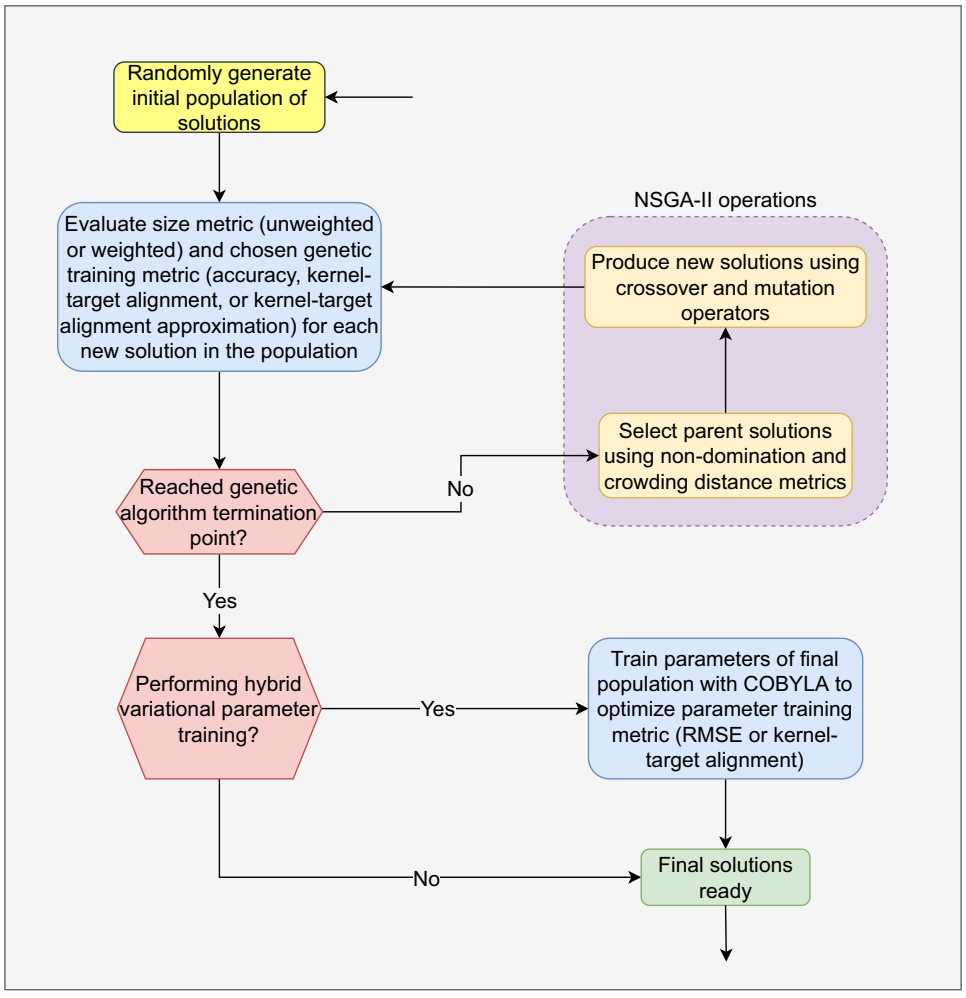


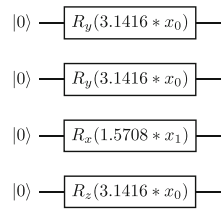
Fig. 2 A flow diagram outlining the algorithm followed to genetically train quantum feature map circuits. The diagram also shows how a hybrid method involving circuit parameter training can be performed after genetic optimization

kernel evaluations are required to compute the exact kernel-target alignment. Therefore, the number of kernel evaluations required when evaluating the kernel-target alignment approximation can be derived as

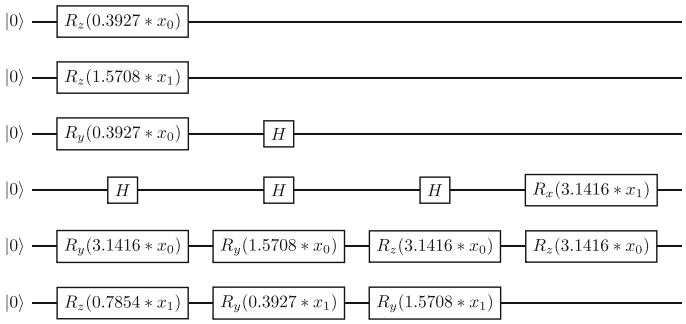
$$a\left(\frac{(\frac{n}{a})^2 - \frac{n}{a}}{2}\right) = \frac{n^2}{2a} - \frac{n}{2},$$

meaning the factor speedup by evaluating the approximation is larger than a, but should approach a as n increases. In this work, we use a value of $a = 5$ in all experiments involving the kernel-target alignment approximation.

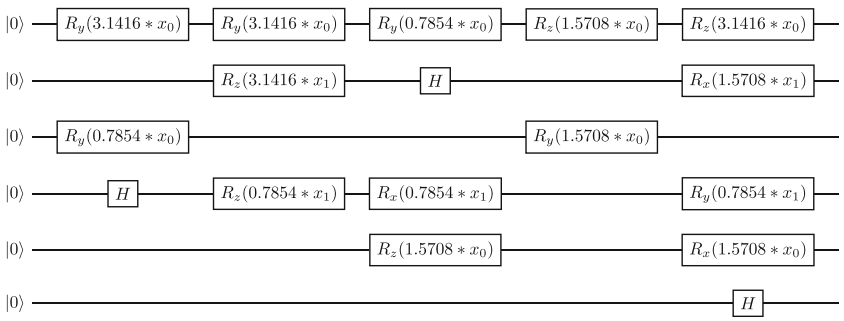
After the genetic optimization in each configuration completes, we attempt further improvement by further optimizing just the proportionality parameters encoded in the last two bits of the gate representation using an implementation of COBYLA (Powell 1994) provided by the NLopt optimization library (Johnson 2011). This allows the parameter values to not be restricted to one of only four possibilities. This optimization aims to either minimize RMSE or maximize kernel-target alignment using the training set to evaluate the metrics. The COBYLA optimizer is allowed one hundred evaluations of the cost function to perform the optimization. A flow diagram outlining the algorithmic process can be seen in Fig. 2.



(a) Approach 1 - Accuracy, Weighted size



(b) Approach 2 - Kernel-target alignment, Unweighted size



(c) Approach 3 - Kernel-target alignment approximation, Unweighted size

Fig. 3 The circuits with highest validation set accuracy produced by the three base genetic approaches when creating quantum feature maps for the Moons dataset. **a** shows the best produced circuit when training to maximize accuracy and minimize weighted size as in the original work, **b** shows the best circuit when training to maximize the exact kernel-target alignment and minimize unweighted size, and **c** shows the best circuit when training to maximize the approximation of the kernel-target alignment and minimize unweighted size. Circuits **b** and **c** are significantly larger. Unused gate layers and qubits are omitted from the diagrams

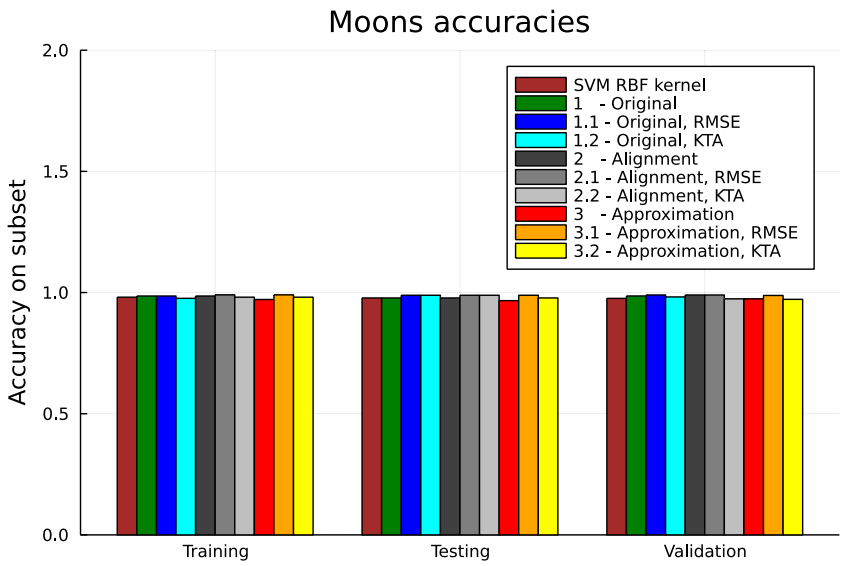


Fig. 4 A graph showing the classification accuracies of the best models produced by various approaches of quantum feature map design on the Moons dataset, compared with a classical RBF kernel for reference. All approaches can be seen to achieve comparable accuracy across the different subsets

The COBYLA cost functions for RMSE and kernel-target alignment both require computing a Gram matrix each evaluation, meaning the number of kernel evaluations performed is $\frac{100(n^2-n)}{2}$. This can be contrasted with the genetic optimization of accuracy or kernel-target alignment, where at least as many kernel evaluations are performed to evaluate the fitness of just the first generation of 100 solutions in the genetic algorithm. In the subsequent 1199 generations $15 \times 1199 = 17985$, more Gram matrix evaluations are performed for a total of 18,085, meaning the final parameter training for the entire output population requires roughly 55% of the number of kernel evaluations performed in the genetic optimization in the cases of genetically optimizing accuracy or the exact kernel-target alignment.

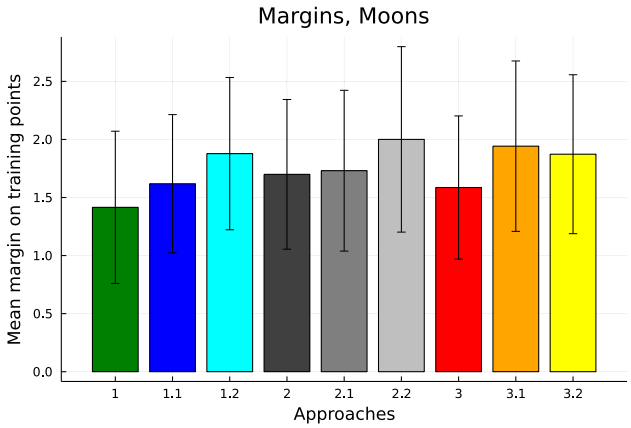


Fig. 5 A graph showing the mean margin of the Moons training set points for the best classifiers produced by each approach, with error bars showing standard deviation. Circuit parameter training and genetic training of kernel-target alignment are both shown to increase the mean margin size. The approach numbering corresponds to the numbering used in Fig. 4

We name the three base approaches 1, 2, and 3, respectively. Each approach has two additional sub-approaches defined for further training of RMSE or kernel-target alignment, for a total of nine approaches. The RMSE and kernel-target alignment variations are named with a.1 and.2 suffix, respectively. We graph the classification accuracies, average margins, ROC curves, feature map circuits, and confusion matrices of the best models produced by each approach, where the best model of a population is taken to be the one achieving the highest validation set accuracy. For two dimensional datasets, decision boundaries are also graphed. Note that results for individual datasets were not averaged over multiple runs of the genetic algorithm or circuit parameter optimization to reduce experiment runtime.

The code implementing the experiments and result graphing can be found on GitHub (Pellow-Jarman 2022).

4 Results

As in the original work by Altares-López et al. (2021), the feature map circuits produced by each of the approaches tend to make little to no use of entangling gates (see Fig. 3). However, the circuits produced by optimizing the kernel-target alignment-based metrics tend to produce significantly larger circuits overall (see Fig. 3). This could be explained by the fact that the weighted size metric in the genetic optimization was replaced with an unweighted size metric in those

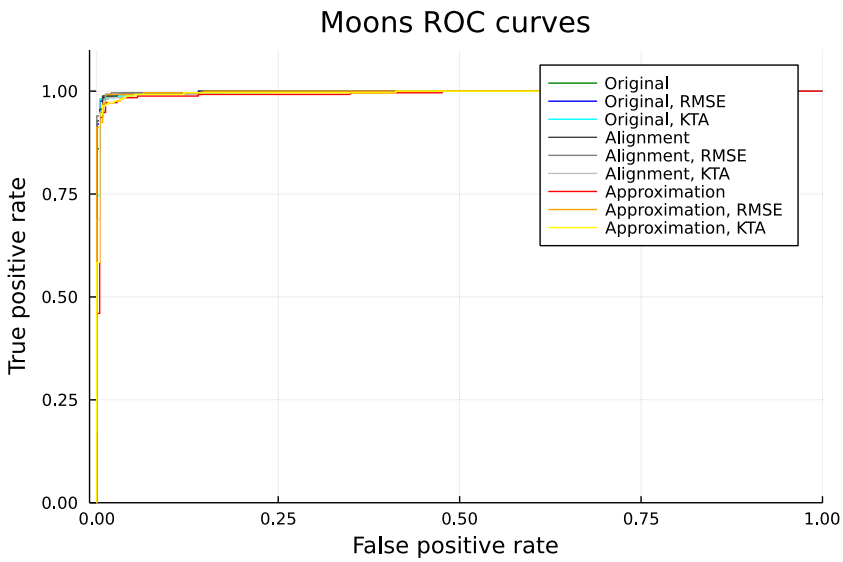


Fig. 6 A graph showing the ROC curves of the best models produced by various approaches of quantum feature map design on the Moons dataset. All of the produced models are shown to perform similarly on the dataset

approaches due to the weighted size metric depending on the test set accuracy, which was not evaluated. A total of 1200 generations were allocated for the genetic algorithm to run. We chose 1200 iterations since this was tested as being sufficient for the convergence of the original approach, and keeping it constant allows for directly comparing metrics in conditions where the faster evaluation time of kernel-target alignment compared with accuracy is trivially satisfied. The optimization of the circuit size did not converge in the allocated 1200 generations in approaches 2 and 3, which can be inferred from the presence of redundant gates. This could possibly be addressed by allowing more generations to pass or using a size metric weighted in kernel-target alignment instead of accuracy, similarly to the original approach. Another possible explanation for the larger circuit size is that a circuit achieving perfect accuracy may still be able to improve its kernel-target alignment; in the accuracy maximization case, the genetic algorithm is able to shift focus to minimizing circuit size after achieving 100% accuracy, but the same cannot be done as easily when maximizing kernel-target alignment since its limiting value of one is more difficult to achieve. Additionally, the high mutation rate of 70% could be reduced to attempt to reach convergence in the allocated 1200 generations. For the produced circuits to have

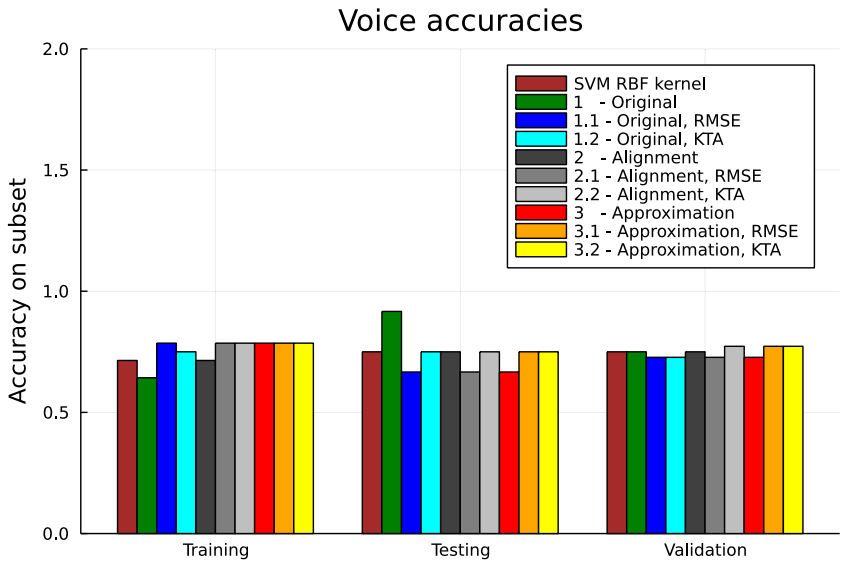


Fig. 7 A graph showing the classification accuracies of the best models produced by various approaches of quantum feature map design on the Voice dataset, compared with a classical RBF kernel for reference. Genetic accuracy maximization is shown to overfit to the testing data used to evaluate the accuracy metric, justifying the necessity of a separate validation set

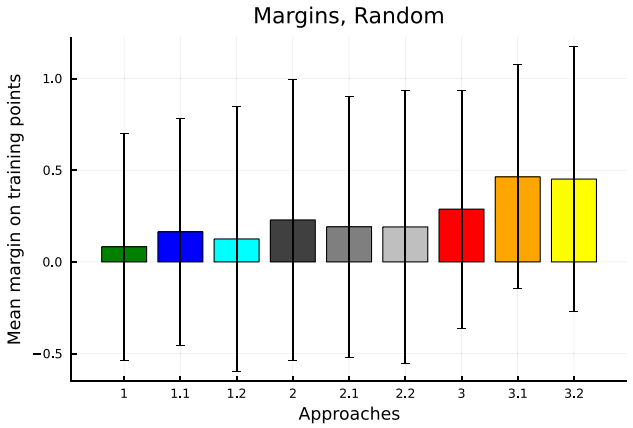


Fig. 8 A graph showing the mean margin of the Random training set points for the best classifiers produced by each approach, with error bars showing standard deviation. The approach numbering corresponds to that in Fig. 4

the potential of a quantum advantage over classical SVM methods, the kernels corresponding to the generated feature map circuits must be difficult to estimate classically. While this is not the case for the feature maps generated with the methods in our results, future works may focus on extending the method to encourage the selection of entangling gates where it might be effective.

Our experiments show that substituting kernel-target alignment or approximated kernel-target alignment for accuracy in the genetic optimization process produces feature map circuits with accuracy comparable to the original approach across all datasets (see Figs. 4 and 6 for example). Further optimizing the final population's trainable parameters using COBYLA was often able to improve the average margin sizes of the classifiers (see Fig. 5) on training data and sometimes able to improve validation set classification accuracy (see Figs. 4 and 9), showing that a hybrid approach performing further optimization of the final population parameter values is worth attempting despite the computational cost if improving accuracy is important (Fig. 6). Training the parameters of a single solution for 100 cost evaluations requires only half a percent of the kernel evaluations as the genetic optimization process, so a smaller subset of the final solutions could be trained at a much lower cost. Additionally, the untrained parameters encoded in the solution binary strings are not lost if further training is performed and can still be used if they happen to perform better than the trained ones.

The results demonstrate that on difficult datasets such as the SUSY, SUSY reduced features, Voice, and Random datasets, the original approach's models can overfit to the testing data used to evaluate the accuracy metric (see Fig. 7). This is likely due to the fact that test set accuracy is directly optimized in the genetic algorithm without regard to training set accuracy. Since the kernel-target alignment approaches make use of only the training data during the genetic optimization, they do not suffer from the same drawback, although they do not show improvement on validation data for the difficult problems. This problem could possibly be avoided by shuffling the training and testing data each generation, although this would make the accuracy metric depend on the generation at which the accuracy was evaluated and could prevent caching of solution fitnesses in the genetic algorithm. A second possible solution is to average the accuracy over subsets of the data. Given a dataset of n points, this can be performed while requiring at most $\frac{n^2-n}{2}$ kernel evaluations in the worst case, since the Gram matrix for the entire dataset can be computed once and used as a

Table 3 Table showing the average margin size of the best classifier produced by each approach, averaged across the nine datasets with equal weighting given to each dataset

Approach	Average margin	Absolute change	Percentage change
1-Accuracy (original work)	0.838	N/A	N/A
Accuracy, RMSE training	0.993	+0.154	+18.42%
Accuracy, KTA training	0.971	+0.133	+15.87%
2-Alignment	1.043	+0.205	+24.46%
Alignment, RMSE training	1.016	−0.028	−2.66%
Alignment, KTA training	1.090	+0.047	+4.49%
3-Approximation	1.065	0.226	+26.99%
Approximation, RMSE training	1.145	+0.080	+7.54%
Approximation, KTA training	1.124	+0.059	+5.59%

For this purpose, the best classifier is defined as the classifier achieving the highest validation set accuracy for the target dataset. The improvement columns for the base approaches (2 and 3) show how genetic optimization of kernel-target alignment and its approximation improve on the margins achieved in the original work. For the hybrid approaches (with additional RMSE and KTA parameter training), the columns show change relative to the base approaches

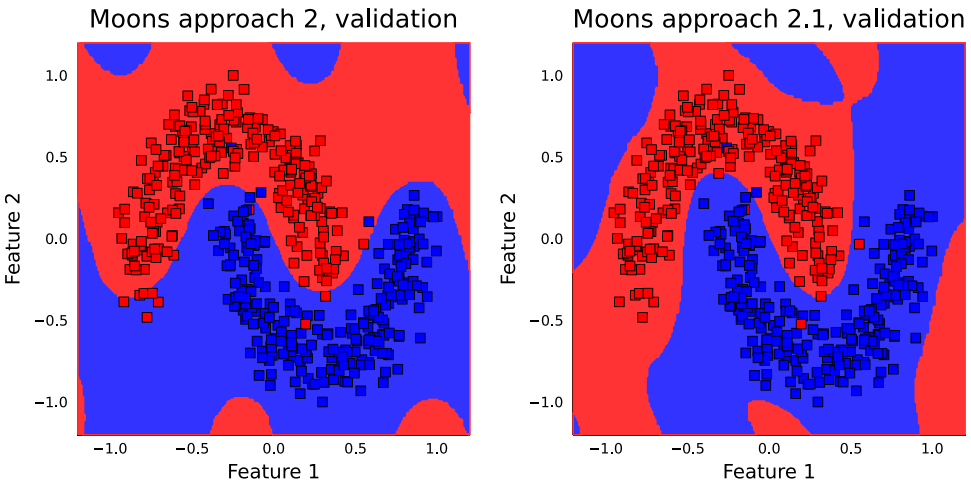


Fig. 9 Decision boundaries of the best classifiers for approaches 2 and 2.1 on the Moons validation data. Further parameter training on training data to minimize RMSE after genetically designing the feature maps to maximize kernel-target alignment is shown to improve classification ability on validation data. The approach numbering scheme is the same as that used in Fig. 4

cache to look up the kernel output for any pair of points when creating models with arbitrary choices of training and testing subsets.

The margins of classifiers trained with the second and third approach tended to be larger than those trained with the first (see Figs. 5 and 8, as well as Table 3). This could be due to the fact that the kernel-target alignment metric and its approximation are evaluated on the training subset as opposed to accuracy which is evaluated on the testing subset, leading to the former two approaches having higher confidence on the training subset. In either case, increased margin size is an indicator of improved generalization ability according to theoretical works (Vapnik 1998; Vapnik and Chervonenkis 2015) showing that margin size bounds VC dimension and VC dimension bounds expected generalization error. Further parameter training on the final generation also tended to show some improvements in margin sizes, even on easier datasets such as the Moons and Circles datasets where there was not much effect on overall classification accuracy. This improvement in margin can be seen visually in the decision boundary graphs of the classifiers (see Fig. 9).

5 Conclusion

In this paper, we compared our implementation of the approach defined in Altares-López et al. (2021) with adjustments to the genetic algorithm cost functions. These adjustments were aimed at investigating the suitability of kernel-target alignment as an alternative metric to test set accuracy and at reducing the number of kernel evaluations required by the approach. The new approaches were shown to still be effective at designing accurate classifiers with fewer kernel evaluations, although at the cost of increased circuit size. They were also shown to often produce classifiers with better margins on training data. We also put forward a hybrid approach extending the original work by applying COBYLA (Powell 1994) to further optimize the trainable parameters of the produced quantum feature map circuits after the termination of the genetic algorithm to attempt further improvement, at a lower additional computational cost than the genetic algorithm's base cost. This parameter training was also shown to be capable of improving margin sizes and sometimes accuracy without increasing the circuit gate cost.

There is still more work to be done in accelerating the genetic algorithm while keeping gate costs low. A potential avenue to achieving this goal is the use of a multi-phase genetic algorithm in which the cost function is initially easy to evaluate but increases in precision after a set number of generations passes. For example, the a parameter of the kernel-target alignment approximation could be made to decrease as generations pass for the approximation to become more accurate at the cost of more kernel evaluations, or the cost function could be switched from kernel-target alignment to classification accuracy to reduce gate cost once a predetermined kernel-target alignment has been achieved.

The original approach could also further be extended to have the gate encoding for parameterized gates select a classical data encoding function such as those used in Suzuki et al. (2020) to introduce classical nonlinearity to the encoding, potentially allowing for even lower gate cost or higher accuracy circuits to be produced.

6 A. Appendix

For Figs. 10, 11, 12, 13, 14, 15, and 16, we show the accuracy of the best produced kernels on each dataset, with the classical RBF kernel also shown for comparison. These figures show that the newly proposed approaches achieve comparable classification accuracy to the original approach, despite not directly evaluating classification accuracy during training. The graphs for the difficult problems (Random,

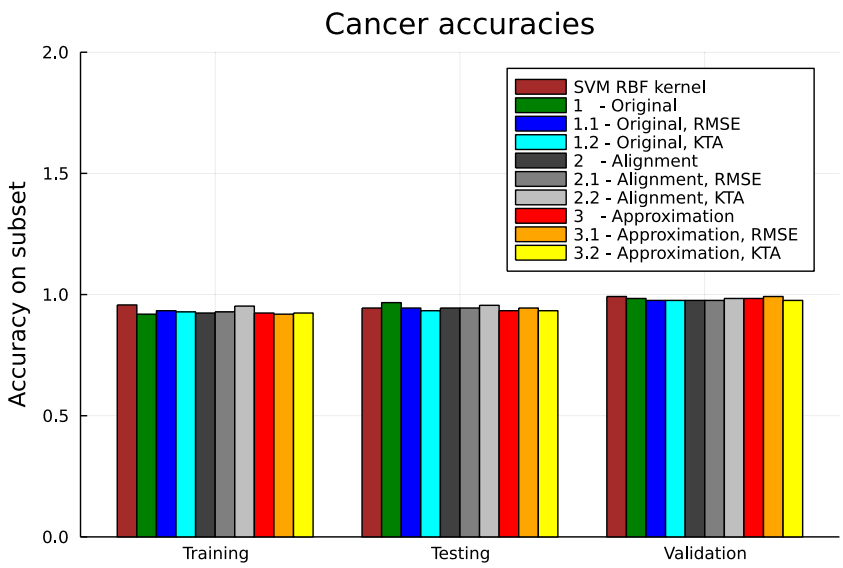


Fig. 10 A graph showing the classification accuracies of the best models produced by various approaches of quantum feature map design on the Cancer dataset, compared with a classical RBF kernel for reference

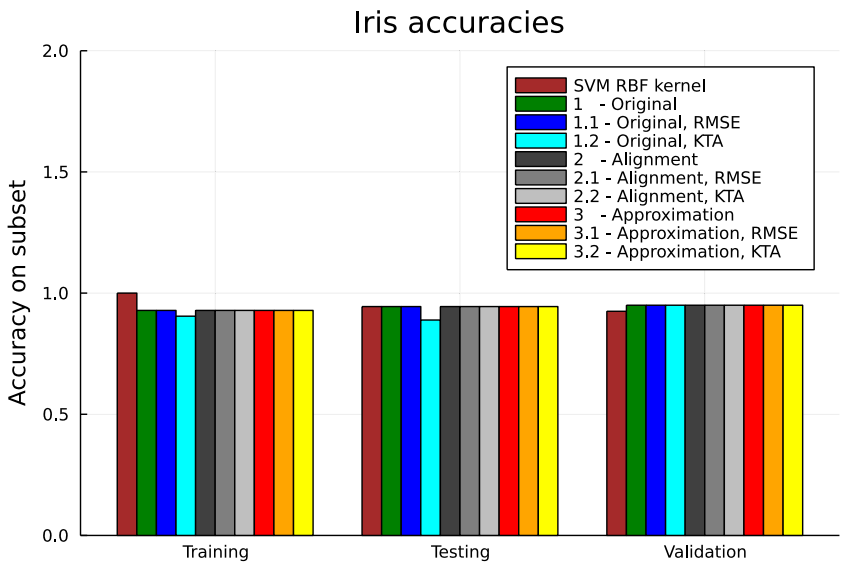


Fig. 11 A graph showing the classification accuracies of the best models produced by various approaches of quantum feature map design on the Iris dataset, compared with a classical RBF kernel for reference

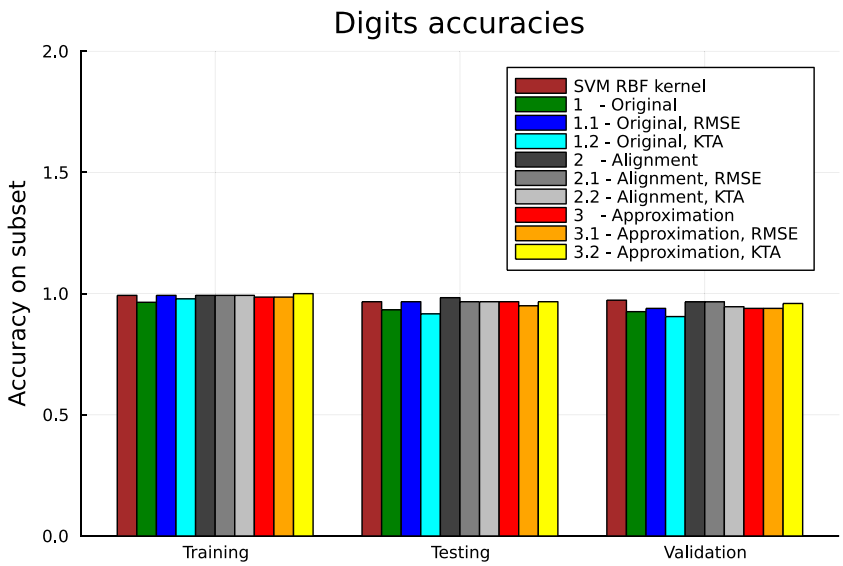


Fig. 12 A graph showing the classification accuracies of the best models produced by various approaches of quantum feature map design on the Digits dataset, compared with a classical RBF kernel for reference

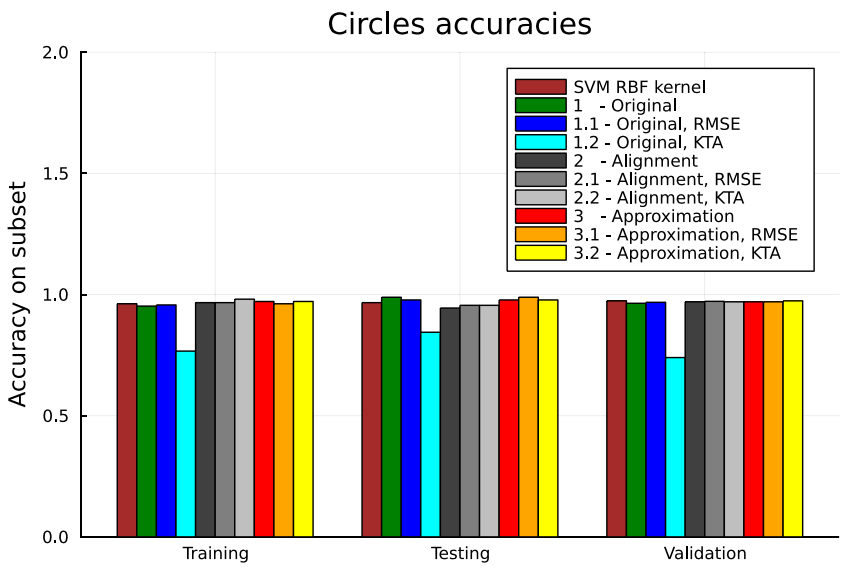


Fig. 13 A graph showing the classification accuracies of the best models produced by various approaches of quantum feature map design on the Circles dataset, compared with a classical RBF kernel for reference

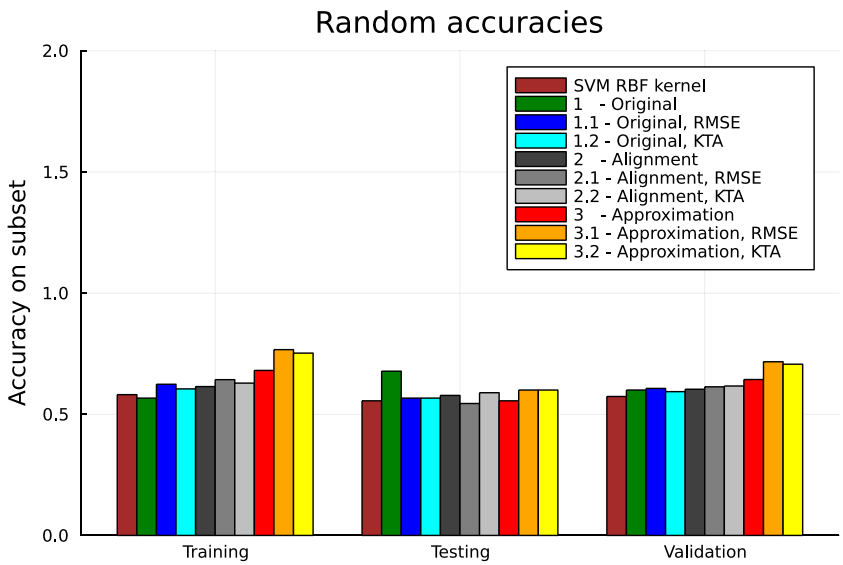


Fig. 14 A graph showing the classification accuracies of the best models produced by various approaches of quantum feature map design on the Random dataset, compared with a classical RBF kernel for reference. In the case of this dataset, the validation set is the union of the training and testing points. This gives an idea of the extent to which the approach was able to memorize all the points

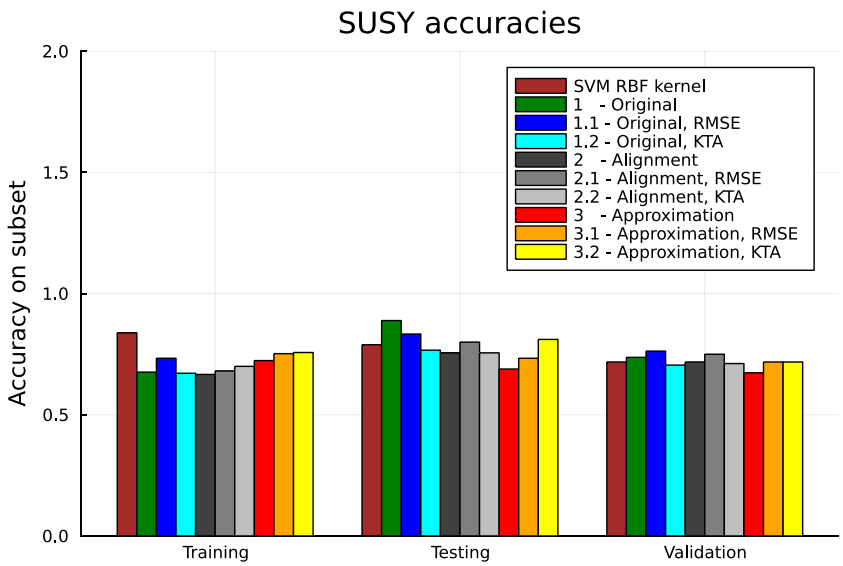


Fig. 15 A graph showing the classification accuracies of the best models produced by various approaches of quantum feature map design on the SUSY dataset, compared with a classical RBF kernel for reference

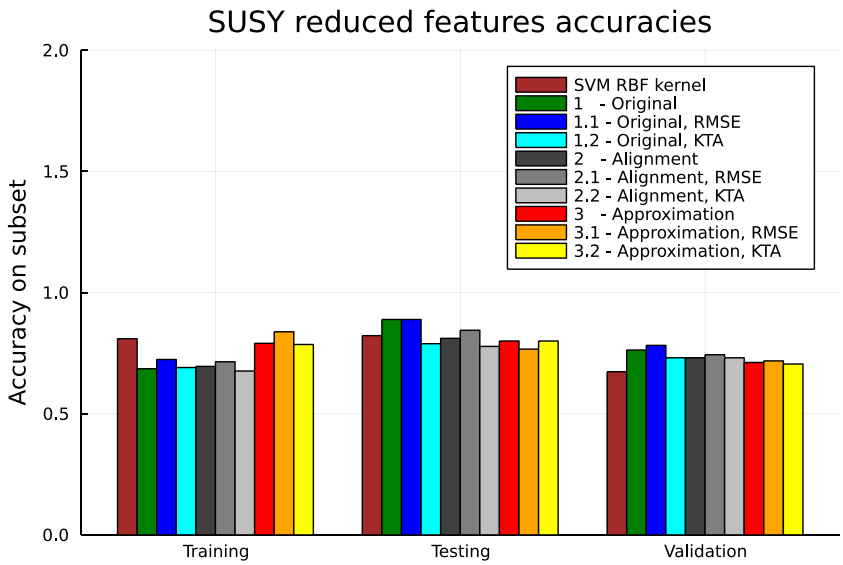


Fig. 16 A graph showing the classification accuracies of the best models produced by various approaches of quantum feature map design on the SUSY reduced features dataset, compared with a classical RBF kernel for reference

Voice, SUSY, SUSY reduced features) show that the original approach has a tendency to overfit to the testing data used to evaluate the accuracy metric and that a separate validation set must be reserved to estimate the generalization of its models. The approaches based on maximizing the approximation of kernel-target alignment are also seen to overfit to training data in these cases more than the other approaches do.

Figures 17, 18, 19, 20, 21, 22, and 23 show the margins of the same best produced kernels on the training data from each dataset. The approaches based on maximizing kernel-target alignment and its approximation tend to have larger average margin sizes, and final parameter training for RMSE and kernel-target alignment also tends to increase the margin sizes. The exception to the trend was the Iris dataset, which was also the second smallest dataset with only 42 training set points.

Figures 24, 25, 26, 27, 28, 29, 30, and 31 show the ROC curves of the best produced models for each dataset. Across the datasets, the curves are mostly similar when comparing approaches.

Figures 32, 33, and 34 show the decision boundaries resulting from the best feature map produced by each approach, for the datasets with two dimensional feature vectors. These can be inspected to see the improvements made by parameter training as well as the complexity of the decision rules produced by each approach.

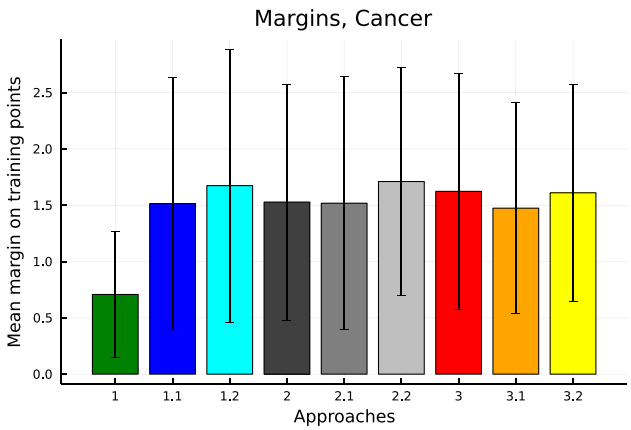


Fig. 17 A graph showing the mean margin of the Cancer training set points for the best classifiers produced by each approach, with error bars showing standard deviation

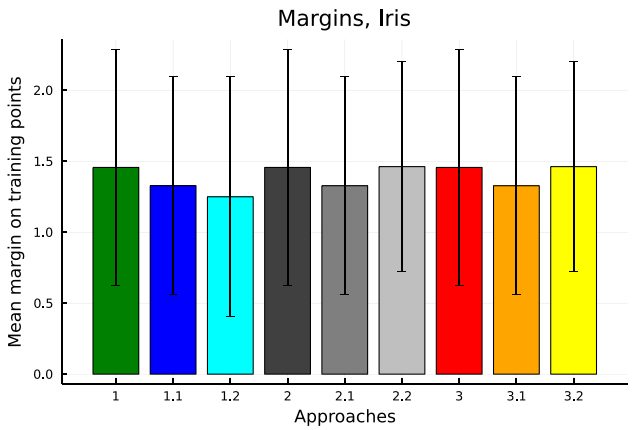


Fig. 18 A graph showing the mean margin of the Iris training set points for the best classifiers produced by each approach, with error bars showing standard deviation

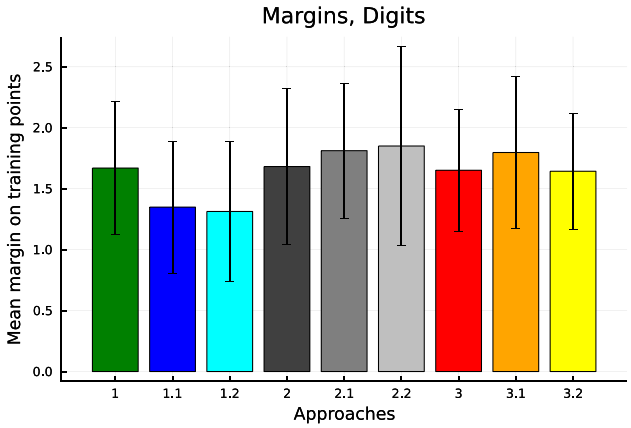


Fig. 19 A graph showing the mean margin of the Digits training set points for the best classifiers produced by each approach, with error bars showing standard deviation

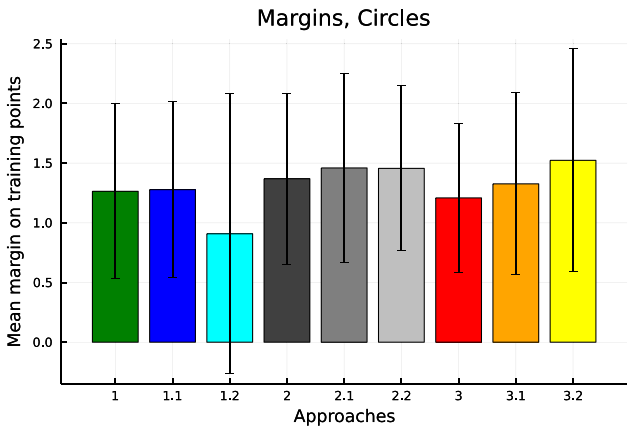


Fig. 20 A graph showing the mean margin of the Circles training set points for the best classifiers produced by each approach, with error bars showing standard deviation

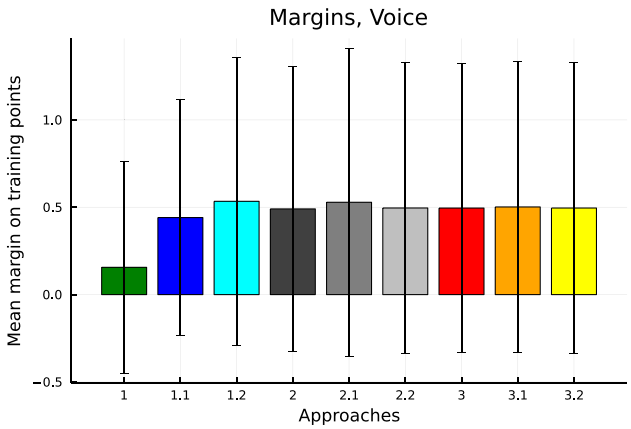


Fig. 21 A graph showing the mean margin of the Voice training set points for the best classifiers produced by each approach, with error bars showing standard deviation

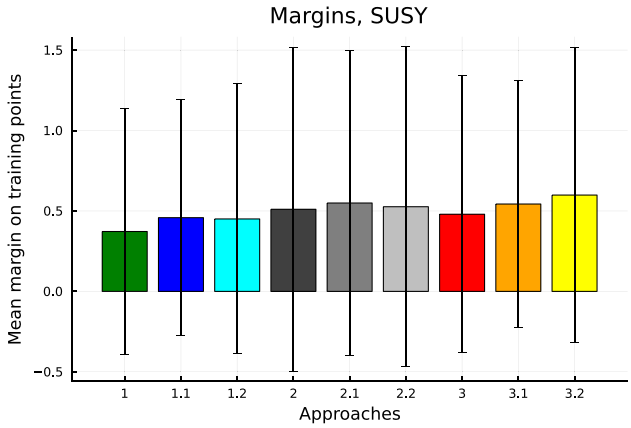


Fig. 22 A graph showing the mean margin of the SUSY training set points for the best classifiers produced by each approach, with error bars showing standard deviation

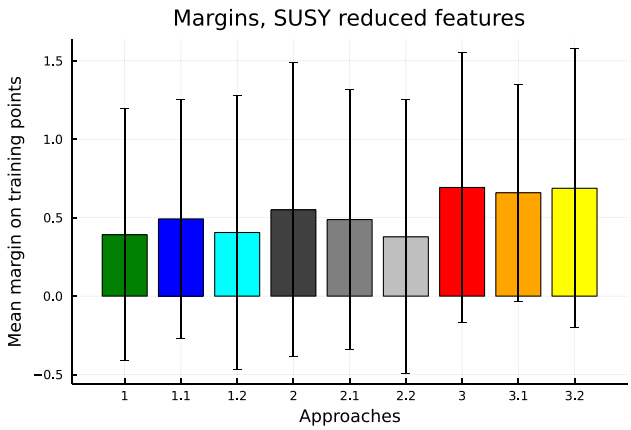


Fig. 23 A graph showing the mean margin of the SUSY reduced features training set points for the best classifiers produced by each approach, with error bars showing standard deviation

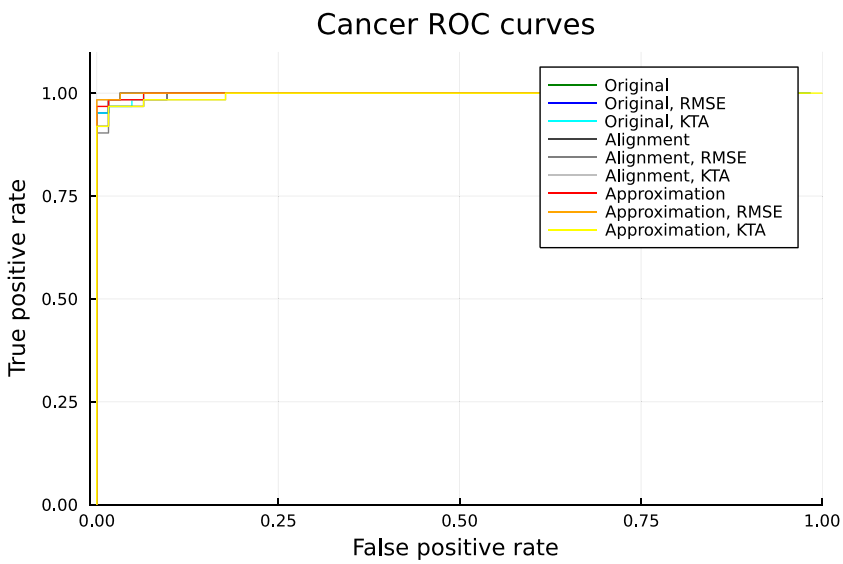


Fig. 24 A graph showing the ROC curves of the best models produced by various approaches of quantum feature map design on the Cancer dataset

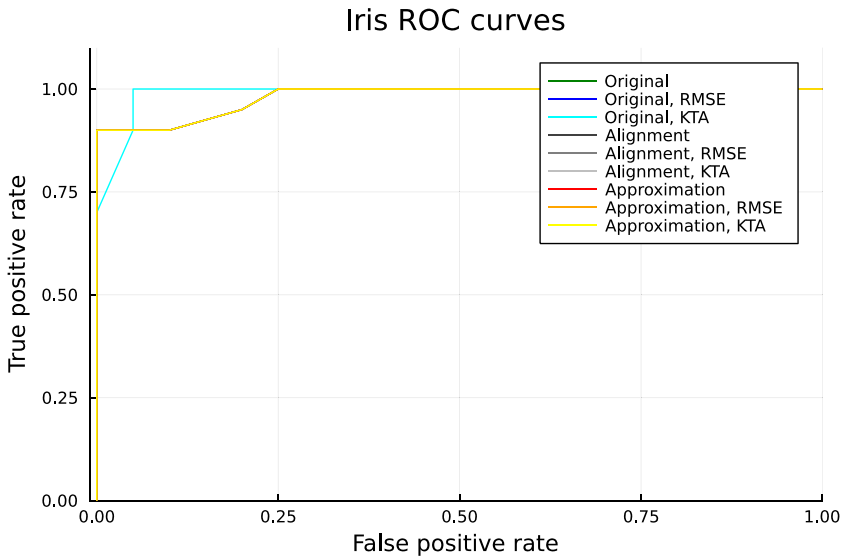


Fig. 25 A graph showing the ROC curves of the best models produced by various approaches of quantum feature map design on the Iris dataset

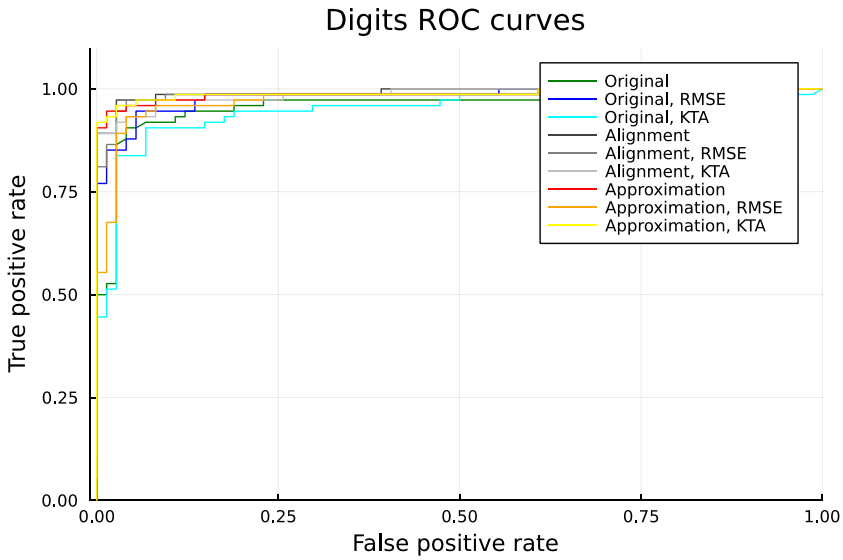


Fig. 26 A graph showing the ROC curves of the best models produced by various approaches of quantum feature map design on the Digits dataset

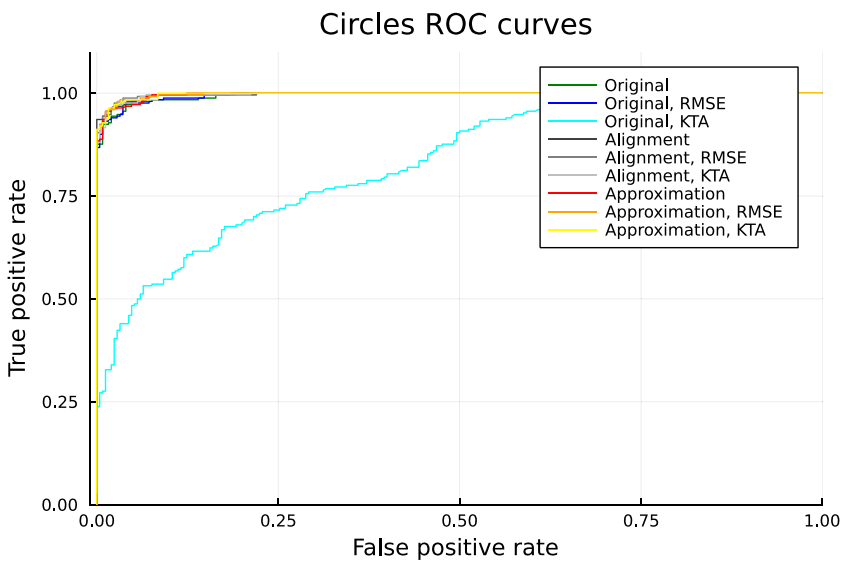


Fig. 27 A graph showing the ROC curves of the best models produced by various approaches of quantum feature map design on the Circles dataset

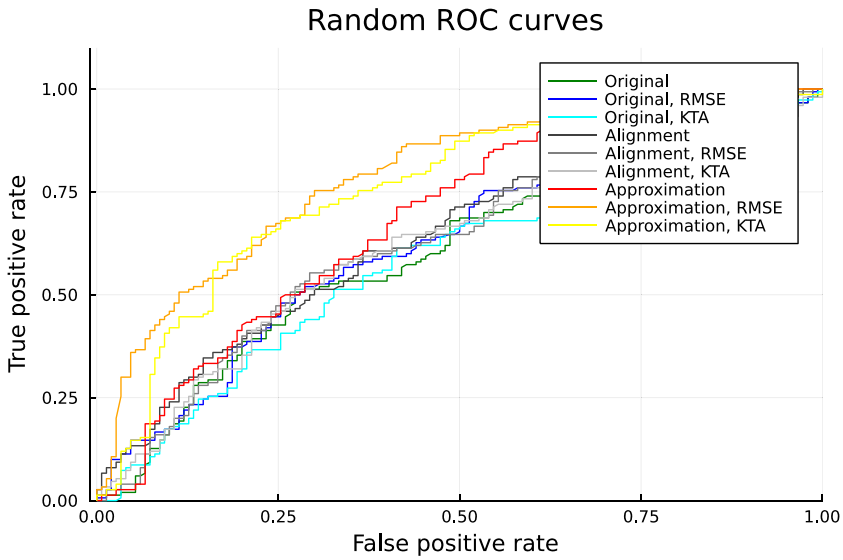


Fig. 28 A graph showing the ROC curves of the best models produced by various approaches of quantum feature map design on the Random dataset

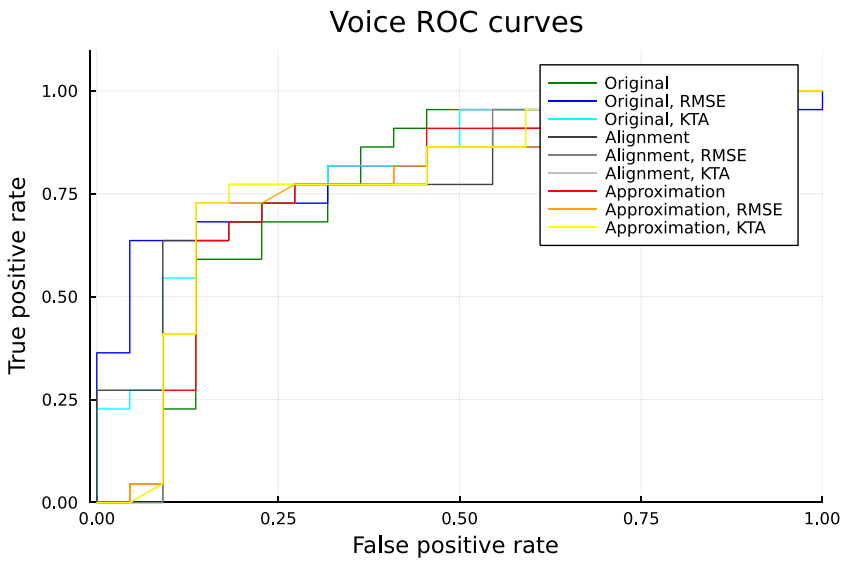


Fig. 29 A graph showing the ROC curves of the best models produced by various approaches of quantum feature map design on the Voice dataset

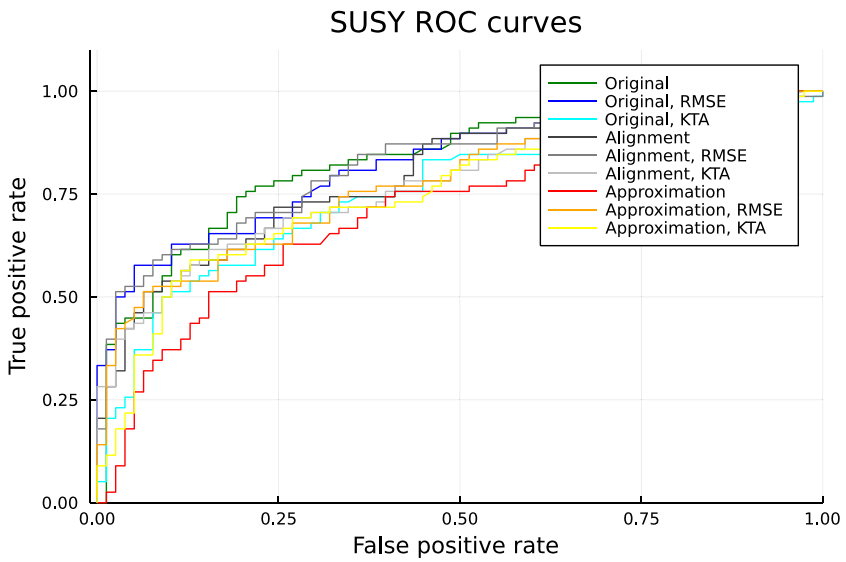


Fig. 30 A graph showing the ROC curves of the best models produced by various approaches of quantum feature map design on the SUSY dataset

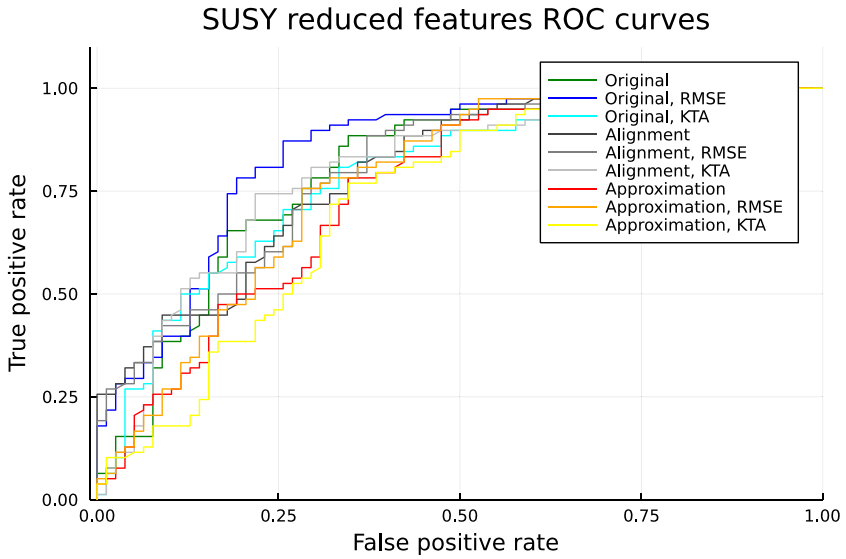


Fig. 31 A graph showing the ROC curves of the best models produced by various approaches of quantum feature map design on the SUSY reduced features dataset

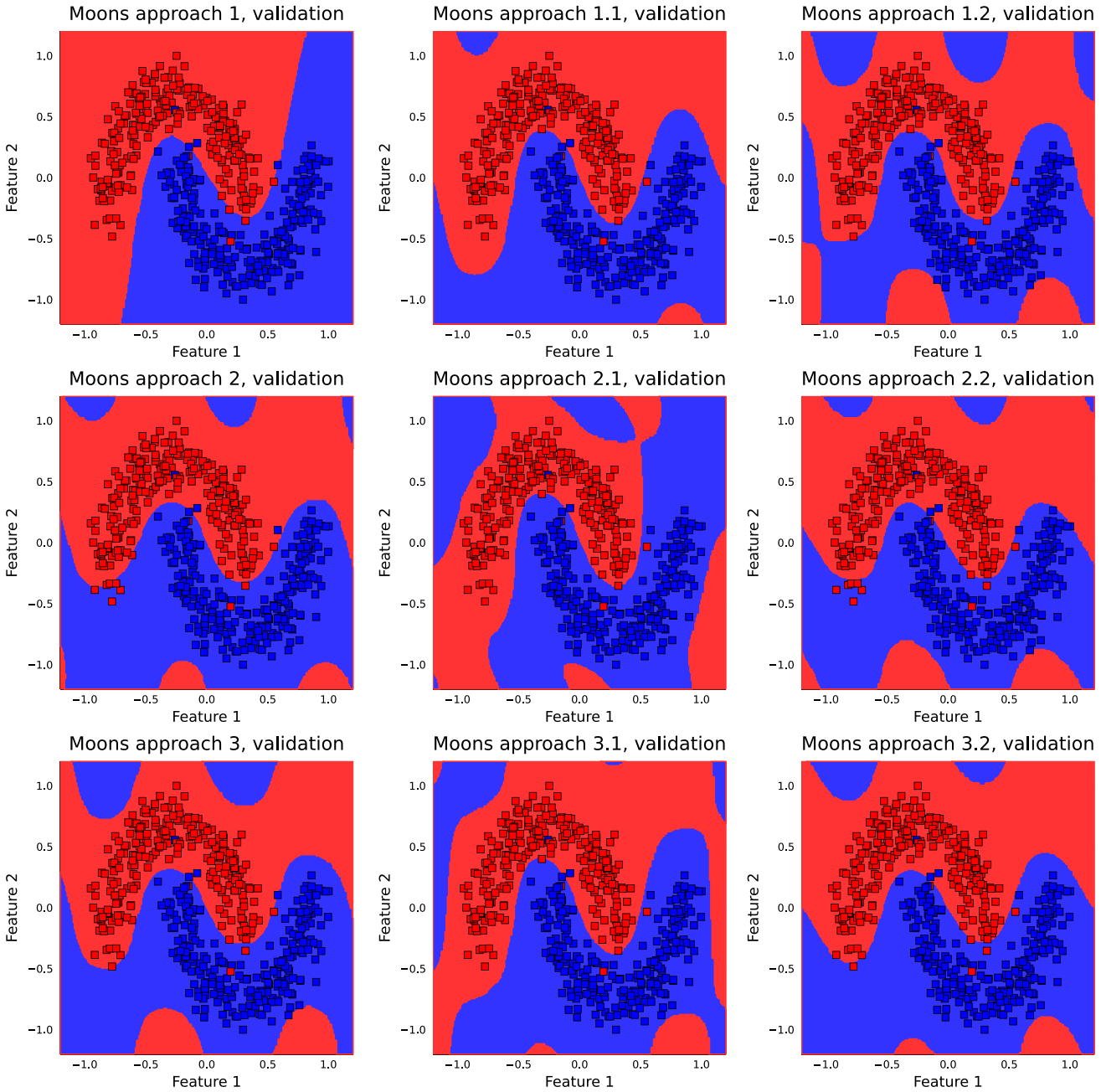


Fig. 32 Decision boundaries for each approach on the Moons validation data

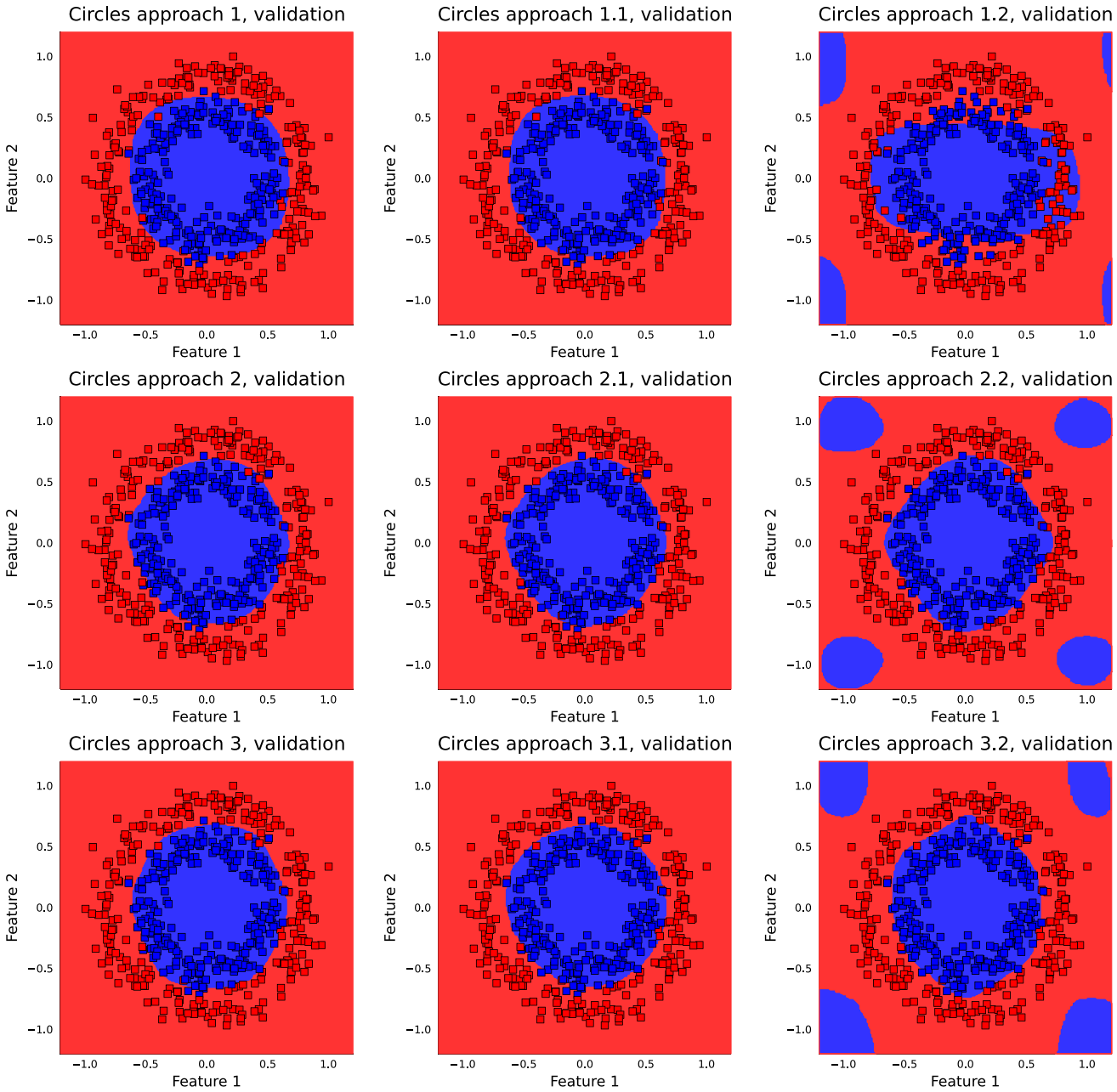


Fig. 33 Decision boundaries for each approach on the Circles validation data

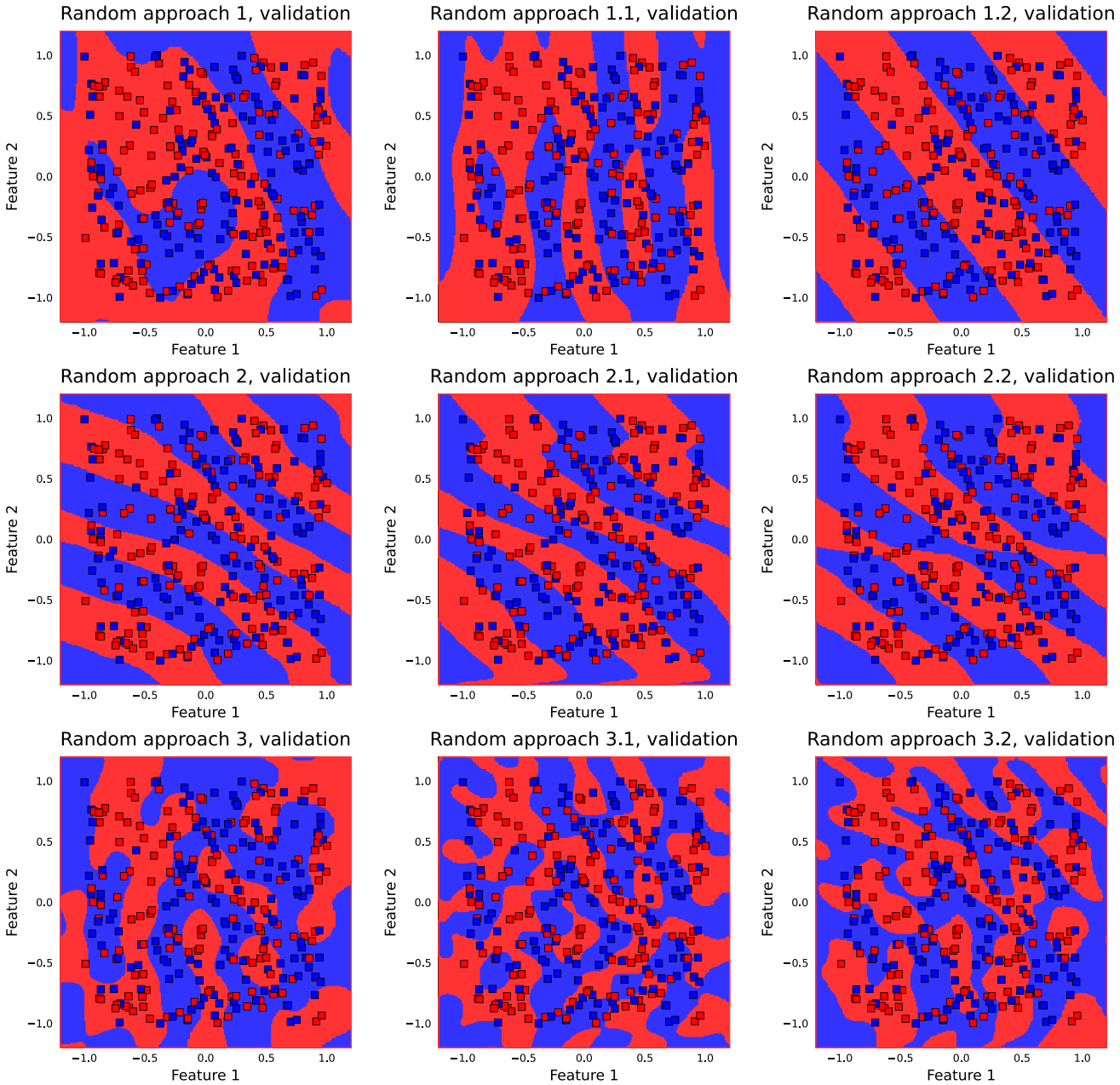


Fig. 34 Decision boundaries for each approach on the Random validation data. In the case of the Random dataset, the validation data does not consist of unseen randomly generated points as would be the case in other datasets. Under the assumption that the models cannot generalize to new pseudorandom data, the validation dataset instead shows the concatenation of the training and testing points to give an idea of the different approaches' capacity for memorizing a random assignment of labels to random points

Author Contributions RPJ wrote the main manuscript text and prepared figures. All authors reviewed the manuscript.

Funding Open access funding provided by University of KwaZulu-Natal. We would like to thank the National Institute of Theoretical and Computational Sciences (NITheCS) and the Center for Artificial Intelligence Research (CAIR) for funding this research.

Availability of Data and Materials All code and data associated with the current submission is available at https://github.com/RowPJ/hybrid-genetic-optimisation-for-quantum-feature-map-design.

Declarations

Conflict of Interest Francesco Petruccione the Chair of Scientific Board and Co-Founder of QUNOVA computing. The authors declare no other conflict of interest.

References

Alloghani M, Al-Jumeily Obe D, Mustafina J et al (2020) A systematic review on supervised and unsupervised machine learning algorithms for data science, pp 3–21. https://doi.org/10.1007/978-3-030-22475-2_1

Altares-López S, Ribeiro A, García-Ripoll JJ (2021) Automatic design of quantum feature maps. Quantum Sci Tech 6(4):045,015. https://doi.org/10.1088/2058-9565/ac1ab1, https://doi.org/10.1088/2058-9565/ac1ab1

Bautu A, Bautu E (2007) Quantum circuit design by means of genetic programming

Bezanson J, Edelman A, Karpinski S et al (2017) Julia: a fresh approach to numerical computing. SIAM Rev 59(1):65–98. https://doi.org/10.1137/141000671, https://epubs.siam.org/doi/10.1137/141000671

Blank J, Deb K (2020) pymoo: multi-objective optimization in python. IEEE Access 8:89,497–89,509

Boser B, Guyon I, Vapnik V (1996) A training algorithm for optimal margin classifiers. In: Proceedings of the 5th annual ACM workshop on computational learning theory, pp 144–152

Chahar V, Katoch S, Chauhan S (2021) A review on genetic algorithm: past, present, and future. Multimed Tools App 80. https://doi.org/10.1007/s11042-020-10139-6

Chen BS, Chern JL (2022) Generating quantum feature maps for SVM classifier. https://doi.org/10.48550/ARXIV.2207.11449, https://arxiv.org/abs/2207.11449

Cristianini N, Shawe-Taylor J, Elisseeff A et al (2001) On kernel-target alignment. In: Dietterich T, Becker S, Ghahramani Z (eds) Advances in neural information processing systems, vol 14. MIT Press. https://proceedings.neurips.cc/paper/2001/file/1f71e393b3809197ed66df836fe833e5-Paper.pdf

Deb K, Pratap A, Agarwal S et al (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197. https://doi.org/10.1109/4235.996017

Dua D, Graff C (2017) UCI machine learning repository. http://archive.ics.uci.edu/ml

Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT press

Grover LK (1996) A fast quantum mechanical algorithm for database search. In: Proceedings of the twenty-eighth annual ACM symposium on theory of computing. Association for computing machinery, New York, USA, STOC '96, pp 212–219, https://doi.org/10.1145/237814.237866, https://doi.org/10.1145/237814.237866

Horak K, Sablatnig R (2019) Deep learning concepts and datasets for image recognition: overview 2019. p 100. https://doi.org/10.1117/12.2539806

Hubregtsen T, Wierichs D, Gil-Fuster E et al (2022) Training quantum embedding kernels on near-term quantum computers. Phys Rev A 106(042):431. https://doi.org/10.1103/PhysRevA.106.042431, https://link.aps.org/doi/10.1103/PhysRevA.106.042431

Jacot A, Gabriel F, Hongler C (2018) Neural tangent kernel: convergence and generalization in neural networks. In: Proceedings of the 32nd international conference on neural information processing systems. Curran Associates Inc., Red Hook, NY, USA, NIPS'18, pp 8580–8589

Johnson SG (2011) The NLopt nonlinear-optimization package. http://ab-initio.mit.edu/nlopt

Kandala A, Mezzacapo A, Temme K et al (2017) Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. Nature 549(7671):242–246. https://doi.org/10.1038/nature23879, https://doi.org/10.1038/nature23879

Khurana D, Koli A, Khatter K et al (2022) Natural language processing: state of the art, current trends and challenges. Multimedia Tools and Applications. https://doi.org/10.1007/s11042-022-13428-4

Liu Y, Arunachalam S, Temme K (2021) A rigorous and robust quantum speed-up in supervised machine learning. Nature Phys 17(9):1013–1017

Lukac M, Perkowski M (2002) Evolving quantum circuits using genetic algorithm. In: Proceedings 2002 NASA/DoD conference on evolvable hardware, pp 177–185. https://doi.org/10.1109/EH.2002.1029883

Luo XZ, Liu JG, Zhang P et al (2020) Yao.jl: extensible, efficient framework for quantum algorithm design. Quantum 4:341. https://doi.org/10.22331/q-2020-10-11-341, https://doi.org/10.22331/q-2020-10-11-341

Myszczynska MA, Ojamies PN, Lacoste AMB et al (2020) Applications of machine learning to diagnosis and treatment of neurodegenerative diseases. Nat Rev Neurol 16(8):440–456. https://doi.org/10.1038/s41582-020-0377-8, https://doi.org/10.1038/s41582-020-0377-8

Ostaszewski M, Grant E, Benedetti M (2021) Structure optimization for parameterized quantum circuits. Quantum 5:391. https://doi.org/10.22331/q-2021-01-28-391, https://doi.org/10.22331/q-2021-01-28-391

Pedregosa F, Varoquaux G, Gramfort A et al (2011) Scikit-learn: machine learning in Python. J Mach Learn Res 12:2825–2830

Pellow-Jarman R (2022) Hybrid genetic optimisation for quantum feature map design. https://github.com/RowPJ/hybrid-genetic-optimisation-for-quantum-feature-map-design

Powell MJD (1994) A direct search optimization method that models the objective and constraint functions by linear interpolation, Springer Netherlands, Dordrecht, pp 51–67. https://doi.org/10.1007/978-94-015-8330-5_4, https://doi.org/10.1007/978-94-015-8330-5_4

Rasconi R, Oddi A (2019) An innovative genetic algorithm for the quantum circuit compilation problem. In: Proceedings of the thirty-third AAAI conference on artificial intelligence and thirty-first innovative applications of artificial intelligence conference and ninth AAAI symposium on educational advances in artificial intelligence. AAAI Press, AAAI'19/IAAI'19/EAAI'19. https://doi.org/10.1609/aaai.v33i01.33017707, https://doi.org/10.1609/aaai.v33i01.33017707

Rebentrost P, Mohseni M, Lloyd S (2013) Quantum support vector machine for big data classification. Phys Rev Lett 113. https://doi.org/10.1103/PhysRevLett.113.130503

Schuld M (2021) Quantum machine learning models are kernel methods

Schuld M, Killoran N (2019) Quantum machine learning in feature Hilbert spaces. Phys Rev Lett 122(040):504. https://doi.org/10.1103/PhysRevLett.122.040504, https://link.aps.org/doi/10.1103/PhysRevLett.122.040504

Shende V, Bullock S, Markov I (2006) Synthesis of quantum-logic circuits. IEEE Trans Comput Aid Design Integr Circ Syst 25(6):1000–1010. https://doi.org/10.1109/tcad.2005.855930, https://doi.org/10.1109/tcad.2005.855930

Shor PW (1997) Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J Comput 26(5):1484–1509. https://doi.org/10.1137/s0097539795293172, http://dx.doi.org/10.1137/S009753979529317

Suzuki Y, Yano H, Gao Q et al (2020) Analysis and synthesis of feature map for kernel-based quantum classifier. Quantum Mach Intell 2(1). https://doi.org/10.1007/s42484-020-00020-y, http://dx.doi.org/10.1007/s42484-020-00020-y

Vapnik V (1998) Statistical Learning Theory. Wiley-Interscience

Vapnik V, Chervonenkis A (2015) On the uniform convergence of relative frequencies of events to their probabilities, pp 11–30. https://doi.org/10.1007/978-3-319-21852-6_3