An Evolutionary Algorithm Design for Pauli-based Quantum Kernel Classification

Yozef Tjandra[1,*,†], Hendrik Santoso Sugiarto[1,†]

[1]Calvin Institute of Technology, Calvin Tower RMCI, Jl. Industri Blok B14, 10610, Jakarta, Indonesia

Abstract

Recently, converting classical data into quantum information brought considerably potential applications in improving machine learning tasks. Particularly, a quantum feature map could provide a promising alternative kernel to enhance a Support Vector Classifier (SVC). While there are very few existing guiding principles to design a high performing feature map, a quantum circuit family called the Pauli feature map is arguably known to behave well. The family is characterized by the occurrence of Pauli gates on the quantum circuits, while it still has several tunable parameters whose optimal values are sensitive to the nature of the datasets. In this work, we present an automatic generation of such feature map using the Genetic Algorithm (GA), aiming to maximize the accuracy of the model while keeping the circuit as simple as possible. We applied the approach to both synthetic and real datasets. The resulting classification metrics and best circuits are discussed in comparison with several classical and quantum kernel baselines. In general, the GA-generated feature maps perform better than other baselines. Moreover, the results show that the evolutionary circuits tend to differ among various datasets, which signify the usability of this generic scheme to determine the best customized quantum feature map for a specific dataset.

Keywords

Quantum Machine Learning, Quantum Circuit Optimization, Genetic Algorithm, Support Vector Machine,

1. Introduction

The progress of quantum information technologies opens a new computational capability beyond classical computation. An important area that can be benefited by quantum computation is artificial intelligence (AI), with main focus on machine learning algorithms[1, 2, 3]. Many studies have been conducted to replicate various classical machine learning models in the quantum computation framework. Several supervised classification models have quantum counterparts, for instances quantum KNN[4], quantum decision tree[5] and quantum NN[6]. Whereas, there exist also some quantum algorithms for unsupervised techniques, such as quantum PCA[7] and quantum clustering[8]. As for more advanced models, certain quantum versions also exist, for example quantum CNN[9] and quantum GAN[10].

Most contemporary machine learning models face challenges of requiring a huge computational resource due to their need for a large amount of data and complex model architectures. In this context, quantum machine learning seeks to enhance the capabilities of machine learning algorithms by using the computational advantages of quantum computing that are hard to simulate classically. Under this situation, an approach called the variational method emerges, in which a set of parameters are applied in a quantum circuit whose values are to be optimized in order to perform classification[11, 12].

On the other hand, there is also another approach that is closely related to the concept of kernel method, which had been very successful in classical machine learning[13]. This approach works particulary well when combined with the well-established support vector machine (SVM)[14] commonly known as the kernel trick. In the quantum version of kernel, each data point is transformed into Hilbert space in the hope that the targeted space prospectively provides more expressive power to help the SVM defines the linear decision boundaries[15, 16].

It is theoretically shown that an optimal quantum variational classifier is essentially a specific form of quantum kernel method-based classifier[17]. Therefore, instead of finding the best parametrized circuit, one could also design a particular kernel to achieve the same result. This motivates us to focus on the quantum kernel method that is applied to the Support Vector Classifier (SVC). Our work will mainly address the strategy to automatically generate an optimal quantum kernel for SVC.

To build a quantum kernel is essentially to encode classical data ($\vec{x}$) into a useful quantum information ($|\psi(\vec{x})\rangle$) that can be processed meaningfully by a quantum computer. While this is notoriously difficult, a common approach is to design a specific quantum circuit known as the quantum feature map. The most common quantum feature map circuit for the quantum kernel SVM is to use

Joint Workshops at 49th International Conference on Very Large Data Bases (VLDBW'23) — International Workshop on Quantum Data Science and Management (QDSM'23), August 28 - September 1, 2023, Vancouver, Canada

*Corresponding author.

†These authors contributed equally.

yozef.tjandra@calvin.ac.id (Y. Tjandra);

hendrik.sugiarto@calvin.ac.id (H. S. Sugiarto)

0000-0001-8756-9176 (Y. Tjandra); 0000-0002-1831-9808 (H. S. Sugiarto)

the ZZ feature map[15]. This map contributes to some considerably good results[18, 19, 20]. However, there is no rigorous explanation of why such map is ideal for the domain-specific works, implying that there might be another better unexplored feature map option. Instead of finding the optimal feature map of all possible quantum circuits, we try to tackle a more modest optimization problem: to find the best Pauli feature map for specific data (since ZZ feature map is a special case of Pauli feature map family). Nevertheless, searching the optimal Pauli feature map is also considerably challenging. Because, as the number of data features grows, the combinatorial possibilities of Pauli feature maps also grow exponentially. Therefore, we need an efficient mechanism to find the optimal Pauli feature map for the data.

To find the optimal Pauli feature map, we propose a genetic algorithm design with specific encoding that is compatible with Pauli feature map family. Genetic algorithm is a well-established meta-heuristic optimization for searching optimal solutions. This method is very versatile and can be adapted to various types of complex problem, including quantum circuit optimization.

The objectives of this work are threefold: (i) to implement the genetic algorithm strategy for automatic generation of the best Pauli feature map for specific datasets, (ii) to investigate whether customized forms of Pauli feature map produce a better classifier than popular classical and quantum counterparts (iii) to understand if there is a consistent pattern among best circuit configurations in varying natures of datasets.

The rest of this paper is organized as follows. In Section 2, several related existing works are reviewed and compared. Section 3 describes detailed explanation of our proposed method to generate the Pauli feature map using the evolutionary strategy. The datasets are described in Section 4 while the results and further discussions are presented in Section 5. Finally, Section 6 concludes the paper along with some future outlooks.

2. Related Works

This work is a part of an optimization problem in which the specific quantum circuit is optimized on the basis of several criteria. The specific meta-heuristic optimization frameworks that we employ is the genetic algorithm that mimics the natural selection process to obtain the fittest individual within generations representing the best solution [21]. Genetic algorithms are used in extremely diverse areas of classical optimizations. To name some instances, it is applicable for path planning of a sensor-based robot[22], job scheduling[23], and image processing[24]. In the field of machine learning, the genetic algorithm is also commonly utilized to find the best neural network architecture, for example one could review its ubiquitous applications in[25].

Recently, genetic algorithms are also used to optimize various quantum circuits for various purposes. The application of genetic algorithm to automatically generate a quantum feature map enhancing a SVC is first formulated in 2021[26]. In the following year, the same authors extend their method with some classical dimensionality reduction strategies in order to accommodate more complex datasets[27]. In their work, the evolutionary design could freely choose the quantum gates structure as well as the entanglement scheme, whereas we rather constrain our design so that the resulting circuit is always under the Pauli feature map family. Although our design seems to be less general, it is able to accommodate any number of qubits. It also serves the main research purpose with the best configuration, that is to understand the performance of quantum kernels within the underlying family of Pauli feature maps.

Later on, many other developments and variations on this topic emerges. Apart from the SVC, the genetic algorithm is also applicable for designing the best circuit configuration in the context of Parametric Quantum Circuit (PQC)[28]. Moreover, a comparison between the SVC enhanced by the evolutionary kernel and the variational ansatz method is also explored[29]. Besides the genetic algorithm, several other methods to automatically generate the quantum circuit to enhance a classification task also exist. In 2022, a Sequential Model-based Optimization (SMBO) using Parzen estimator to find the best ansatz circuit is proposed [30]. In a very recent paper, a Bayesian approach to adaptively construct the feature map for preparing an SVM task is presented[31].

3. Methods

3.1. Quantum Kernel Classification

Classification is a type of machine learning framework to classify data based on labeled training examples. Labeled training data consists of input data with associated output labels or categories, and the algorithm learns to identify patterns and relationships between the input data and their corresponding labels. Once trained, the algorithm can then be used to classify new unlabeled data into one of the predefined categories.

A nonlinear classifier is needed when the data is too complex and cannot be separated by a linear boundary decision. One of the most common nonlinear classifier is a kernel SVM[14]. Kernel SVM is capable of separating nonlinear data by mapping the original data using a complex function into a higher-dimensional feature space to enable better linear separation between different classes. Specifically, a nonlinear kernel term is inserted into standard SVM prediction. The kernelized

binary prediction for data test $\vec{x}'$ depends on the sign of $\sum_{m=1}^{M} c_m y_m K(\vec{x}_m, \vec{x}') + b$, where c_m and b represent the trainable parameters, y_m is the data label, and $K(\vec{x}_m, \vec{x}')$ is the kernel between data training $\vec{x}_m$ and data test $\vec{x}'$.

In quantum kernel classification, the kernel function is implemented using a quantum circuit which is capable of transforming low dimensional classical data into high dimensional quantum states (i.e. Hilbert space), allowing for the exploration of quantum interactions between data points. These quantum feature maps can be tailored to capture specific patterns in the data, enhancing the classification performance. The key idea behind quantum kernel is to compute the inner product between pairs of quantum information efficiently. The inner products can be used to obtain the feature kernel, $K(\vec{x}, \vec{z}) = \left|\langle \psi(\vec{x}) \mid \psi(\vec{z}) \rangle\right|^2$. By performing quantum inner product, the computational advantages of quantum computing such as exponential speedup can be acquired[18].

3.2. Pauli Feature Maps

In traditional machine learning, feature maps are used to transform raw input data into a higher dimensional feature space. Similarly, quantum feature maps are used to transform classical data into a quantum state that can be processed by a quantum computer. In a quantum feature map, the input data is transformed using a quantum gates operation to produce a new quantum state vector that contains higher-order correlations between the original data points ($|\psi(\vec{x})\rangle = V_\Phi(\vec{x})|0\rangle^{\otimes N}$). The quantum feature maps are able to efficiently generate complex transformation that are computationally hard to construct using classical method. Moreover, the base quantum circuit operation can also be repeated multiple times to construct more complex feature maps.

Quantum feature maps have been shown to be effective in a variety of machine learning tasks, and ongoing research is exploring new types of quantum feature maps and their applications in practical machine learning problems[16, 18, 26]. ZZ feature map is the most common proposed feature maps for kernel support vector machine problem because of its simplicity and experimental performance[15]. ZZ feature map itself is a second order Pauli feature maps. In general, any Pauli feature maps that transform input data with N features $\vec{x} \in \mathbf{R}^N$ into quantum information in N qubits $|\psi(\vec{x})\rangle$ can be described as unitary operator below:

$$V_{\Phi(\vec{x})} = \underbrace{U_{\Phi(\vec{x})} H^{\otimes N} \cdots U_{\Phi(\vec{x})} H^{\otimes N}}_{\text{\# repetitions}} \tag{1}$$

$U_{\Phi(\vec{x})}$ represents the Pauli expansion matrix:

$$U_{\Phi(\vec{x})} = \exp\left(i \sum_{S \subseteq [N]} \alpha \phi_S(\vec{x}) \prod_{j \in S} P_j \right). \tag{2}$$

with data mapping:

$$\phi_S(\vec{x}) = \begin{cases} x_j & \text{if } S = \{x_j\} \\ \prod_{j \in S} (\pi - x_j) & \text{if } |S| > 1 \end{cases} \tag{3}$$

where S is some n-subset of the N feature indices representing the connections between different qubits, $P_j \in \{I, X, Y, Z\}$ represents the Pauli matrices and α is a variable to adjust the magnitude of Pauli rotation gates.

In this paper, we will optimize the combination of Pauli sequence, number of repetitions, entanglement type, and α. These parameters are arguments on the PauliFeatureMap class in Qiskit Python package[32]. The Pauli sequence represents the possible choices for P_j. The number of repetitions represents how many times the Pauli expansion circuit is repeated. The entanglement type represents the entanglement structure among qubits. In general, every qubit can be either entangled or not with other qubits in various different graph structure. To simplify the situation, we use only 4 possible schemes: full, circular, linear, and reverse linear. In full entanglement, all pairs of qubits are connected, while on all other schemes, only consecutive qubits are entangled. The difference is that in circular scheme, the last qubit is connected to the first one, while the other two schemes does not allow this. The distinction between the rest is that the entanglement of the reverse linear is in the opposite direction of that of the linear scheme. Originally, α is continuous variables. However, in our scheme, α is constrained into 16 possible values only: $\left\{\frac{\ell\pi}{16} : \ell = 1, \ldots, 16\right\}$, to simplify the combinatorial search space.

In this context, the ZZ feature map is only a special case of Pauli feature maps with Pauli sequence $[Z, ZZ]$, no repetition, full entanglement, and $\alpha = 1$. This quantum gates configuration can also be associated with Ising interaction[15]. For 2 features (2 qubits), the Pauli expansion matrix of ZZ feature map can be written as:

$$U_{\phi(\vec{x})} = \exp\Big(i x_0 Z_0 + i x_1 Z_1 + i(\pi - x_0)(\pi - x_1) Z_0 Z_1 \Big). \tag{4}$$

The first two terms are equivalent with R_Z rotation gate on each individual qubit. Specifically, $\exp(i x_0 Z_0) = R_Z(2x_0)$ and $\exp(i x_1 Z_1) = R_Z(2x_1)$. Furthermore, the tensor product, $\exp(i(\pi - x_0)(\pi - x_1) Z_0 Z_1)$ is equivalent with entangled gates: $CX \cdot (I \otimes R_Z(2(\pi - x_0)(\pi - x_1))) \cdot CX$.

3.3. Genetic Algorithm

In this work, we use a metaheuristic approach, namely the Genetic Algorithm (GA) to determine the best feature map to assist the SVC. For a classic overview of the algorithm, one can consult [21]. The method is well known for tackling the local minima problem by employing random exploitation and exploration within the search space.

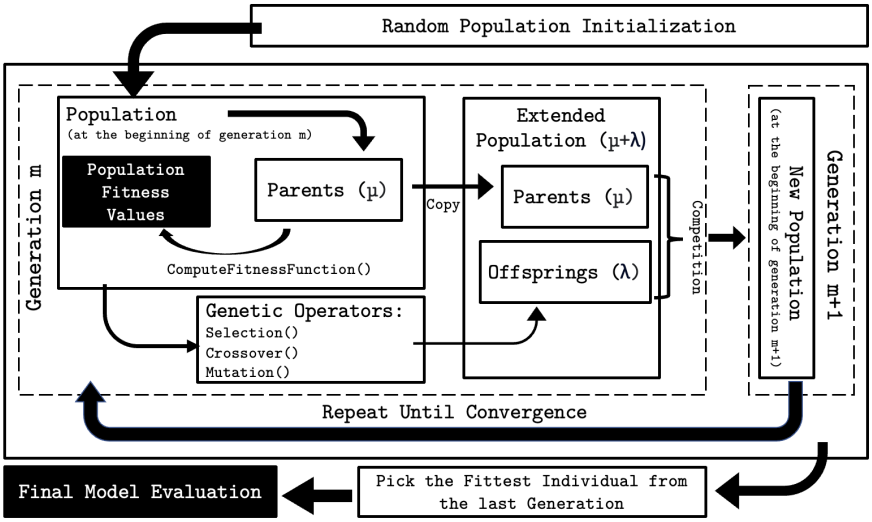

Figure 1: Overview of Genetic Algorithm with $(\mu + \lambda)$ strategy. The processes described by the black-colored boxes in the chart are the only parts involving training and evaluation of Quantum SVC model.

In particular, GA will automatically tune the quantum circuits configuration under the Pauli feature-maps family to achieve the best quantum kernel. Whereas, the SVC uses this kernel and internally optimises its parameters.

We provide an overview of the GA design with $(\mu + \lambda)$ strategy, summarized in Figure 1, as follows. Initially, the algorithm generates a population of random individuals of size μ, encoded by a 4-ary string each of which represents a certain instance of the Pauli quantum feature map as a candidate for solution. For a certain number of generations, the algorithm performs the fitness values computation as well as with certain probabilities, some genetic operators to each individual to filter out the fittest individual representing the best solution for the optimization problem. The more fitness value of an individual has, the more chance of survival it gains within generations. The genetic operators include the followings: selection operator to perform exploitation while crossover and mutation operators to perform exploration. These operators are applied in order to produce λ number of new offsprings. These processes are repeated for a certain (fixed) number of generations that is chosen as a hyperparameter of the GA. Finally, the fittest individual of the last generation is taken to be the final solution and an additional evaluation is performed using a predetermined testcase to obtain the overall score of GA solution.

3.3.1. Genetic Encoding

To represent a quantum Pauli Feature Map, there are 4 key elements to encode: the sequence of Pauli strings, the number of repetitions, the type of entanglement, and the Pauli rotation factor. All these are represented using a 4-ary string of length 25 whose alphabet is taken from $\{0, 1, 2, 3\}$. The detailed explanation of the encoding is provided in Figure 2. Note that the design of this Pauli feature map encoding is applicable for any number of features that the map would receive. However, if the number of features is $N < 4$, then the Pauli strings are restricted to have length of at most N by manipulating the encoding rule for the string length control digits.

We list down some examples to help the reader understand the genetic encoding as provided in Table 1. We put some additional remarks on the examples as follows. The Pauli sequence of the first example is explained in the example box of Figure 2. As the 0-th digit of the second example is 0, it means that the Pauli sequence only has length 1 and thus the substring $a_6 a_7 \dots a_{20}$ would not contribute anything to describe the feature map. The similar fashion also appears when the first digit of a block of Pauli string control has value less than 3. For instance, consider the substring $a_{16} a_{17} \dots a_{20}$ in the first example which controls the last Pauli string of the 4-length Pauli sequence. The value $a_{16} = 1$ tells that the Pauli string only has length 2 implying that the values of a_{19} and a_{20} play no role in describing the feature map.

3.3.2. Fitness Values

As the GA encoding is able to expressively represent various deep and highly entangled circuits, some of them might provide a considerably good quality of classification performance. However, at certain point, an extremely complex circuit would not be practical to deal

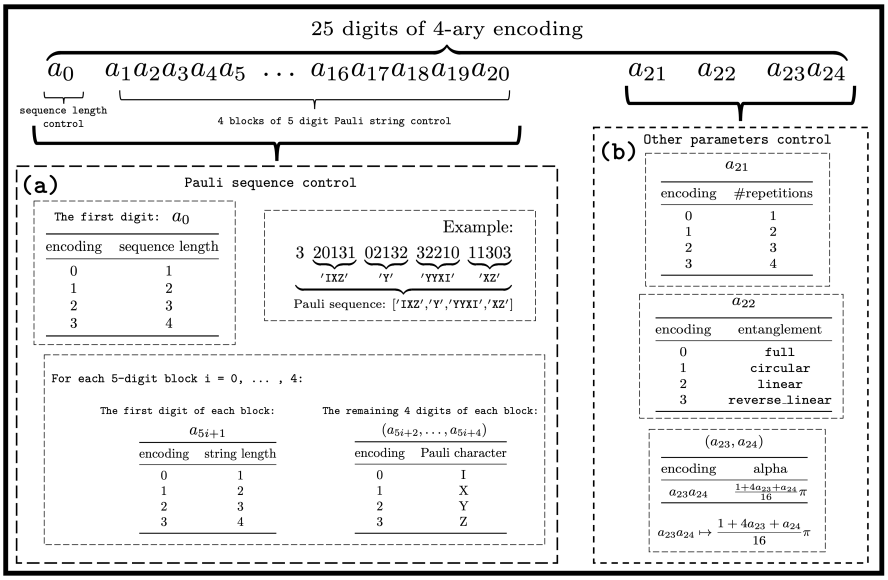

Figure 2: Description of Pauli Feature Map Encoding parameterized by a 4-ary string of length 25. (a) The first 21 digit control the Pauli sequence where the first digit a_0 defines the length of the sequence and the next $5(a_0 + 1)$ digits describes the $(a_0 + 1)$ Pauli strings in $(a_0 + 1)$ blocks of 5-digit strings. (b) The last 4 digits of the encoding describe three other related parameters.

Table 1
Some Genetic Encoding Examples

no	genetic code	Pauli Sequence	#repetitions	entanglement	alpha
1	3 20131 02132 32210 11303 0 1 03	['IXZ', 'Y', 'YYXI', XZ']	1	circular	$\frac{4\pi}{16}$
2	1 03221 21203 23301 11022 2 0 20	['Z', 'XYI']	3	full	$\frac{9\pi}{16}$

with the error correction of the near-term quantum computers. Hence, in this work, the design of the fitness function would balance between the good classification performance and the complexity of the circuit.

Most of the fitness function design in this work is taken from [29]. First, the complexity of a circuit is defined as

$$\text{Complexity} = N_{\text{RGate}} + 2N_{\text{HGate}} + 5N_{\text{CNOTGate}}, \quad (5)$$

where $N_{\text{RGate}}, N_{\text{HGate}}, N_{\text{CNOTGate}}$ are respectively the number of R_P, H, and CNOT gates in the circuit. Hence, the more gates a circuit has, its complexity increases.

Since the optimization's objectives are to maximize the accuracy of the SVC and to minimize the complexity of the circuit, the fitness value is defined as follow with w is some positive tunable constant to control the balance between both objectives, determined experimentally.

$$\text{Fitness} = \text{Complexity} + \frac{w}{\text{Accuracy}}. \quad (6)$$

3.3.3. Genetic Operators

Figure 3 provide general explanation of how the genetic operators work. Under the scheme of k-tournament selection, at rounds, k random individuals in the population are selected to join the tournament at which the one having the greatest fitness value wins the tournament and thus proceed to the next genetic operators. Particularly, this work uses $k = 5$. Next, the parents selected by the tournaments are randomly paired to undergo the 2-point crossover scheme with some hyperparameter probability. For the mutation, we use a slight modification of the bitflip scheme: once a 'bit' is decided to mutate, it will randomly choose a new character from the 4-ary alphabet uniformly.

4. Data

4.1. Dataset

4.1.1. Synthetic Dataset

We synthetically generate 3 datasets using the Scikitlearn library[33]: the blobs, the noisy circles, and the noisy moons datasets (Figure 4). These datasets are configured to have binary labels, 2 features and 200 instances.

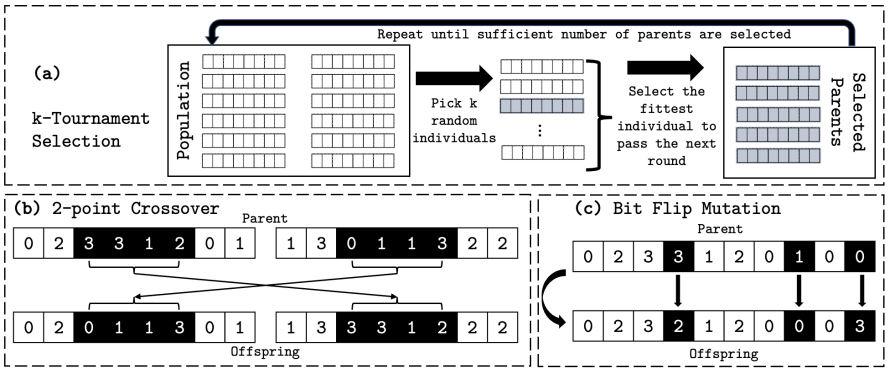

Figure 3: Workflow of the genetic operators: (a) *k*-Tournament Selection. (b) 2-point crossover (c) Bitflip mutation.

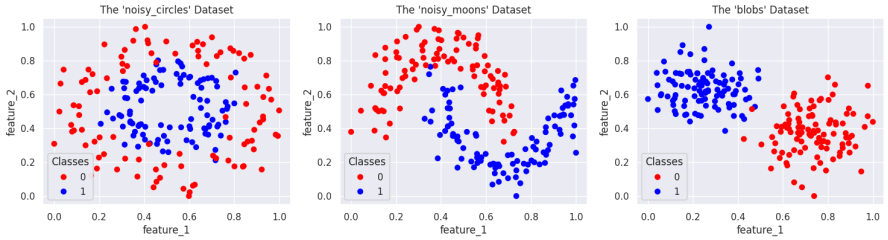

Figure 4: Three Synthetic Datasets for Classification.

Table 2
Real Dataset Specification

Dataset	iris	blood	irish	veteran
source	[35]	[36]	[37]	[38]
# features	4	4	5	7
# instances	150	748	500	137
# labels	3	2	2	2
include categorical data?	No	No	Yes	Yes

4.1.2. Real Dataset

To ensure that the method also works well on real data, we perform the same benchmarking on several well-known datasets commonly used to evaluate classification models. Table 2 provides some description of each dataset used in this works. All of these datasets are obtainable via OpenML Python Package[34].

4.2. Data Preprocessing & Validation

To set up the training, we normalize the data instances and split them into 80% training set and 20% testing set. To avoid the overfitting problem, we divide the training set into 4 groups to perform the 4-fold cross-validation. In each group, the model is trained and tested to obtain two classification evaluation metrics: accuracy and f1-score (macro-averaged). We include the f1-score since it represents the balance between the other two well-known metric scores, precision, and recall. The average scores among those of the 4-fold groups will contribute to the fitness values computed during the genetic algorithm. By having the 4-fold scheme, the feature map obtained during the training phase would not overly suit to a very specific group of training set while behave very poorly on another group. Finally, once the best feature map is decided by the GA, we employ it to perform the last training using all the 80% train set. The overall evaluation score of the model is then determined by the last 20% test set which has not been seen at all by the model during the evolutionary training.

5. Results and Discussions

We apply the GA to generate quantum feature map to both synthetic and real datasets in order to review its effectiveness. All performance metrics presented in this section are the ones obtained on the test dataset whose instances are not included during the training.

The GA hyperparameters used in this work are presented in Table 3. These values are generally inherited

Table 3
GA hyperparameters used in the optimization process

Hyperparameter	Value
Population Size (μ)	100
Offspring Size (λ)	60
Number of Generation	80
Crossover Probability	0.3
Individual Mutation Probability	0.7
Bitflip Mutation Probability	0.25

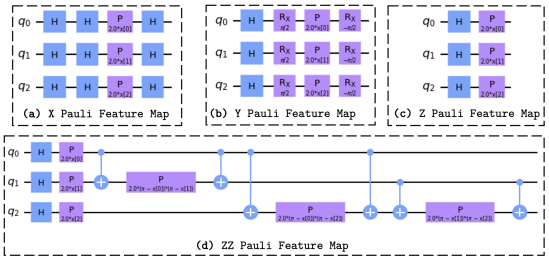

Figure 5: 4 Predetermined Quantum Feature Maps Baselines.

from [26] while also experimentally fine-tuned. We remark that the relatively low number of generations chosen in this work is determined based on experiments, to avoid the GA to overfit the training data too much. This is because as the number of generation goes up, while the accuracy of the training set rises, at a certain point, the performance on the test set starts to worsen. The construction of the quantum circuit in this paper is implemented using the Qiskit framework[32] (IBM Python library for quantum computing) on classical computers.

As baselines, we pick three classical and four quantum kernels to assist a support vector machine model. The classical include the linear, polynomial, and radial based kernels. For the quantum kernel baselines, we use four circuits based on Pauli feature map families: the X, Y, Z, and ZZ feature maps. One can observe some instances of these circuits on small number of qubits in Figure 5. We then compare their performances to the feature map automatically obtained by the genetic algorithm.

5.1. Classification on Synthetic Datasets

We present the classification results on the synthetic datasets in Table 4. The best metric score for each dataset are written in bold while the best scores among each baseline group are also emphasized in italic. The Pauli feature map circuits produced by the GA best individual for each dataset are provided in Figure 6. The detailed description of each feature map is accessible in Table 5.

While the blobs dataset seems not challenging enough to discriminate between the classical and quantum feature maps, the other two datasets are able to show better performances of the GA-generated feature map over the other baselines. It is reasonable since the blobs dataset is basically linearly separable, while the other two have non linear decision boundaries. For the noisy-moons dataset, the quantum baselines perform worse than the classical ones, in particular the RBF kernel could do a perfect task. Note that the same quality also achievable by the quantum feature map provided by the GA. Meanwhile, the quantum kernels are also generally outperformed by the classical kernels for noisy-circles. However, the GA-generated quantum feature map provides a better performance than the best classical baseline. It shows that ZZ feature map is too complex for simple data.

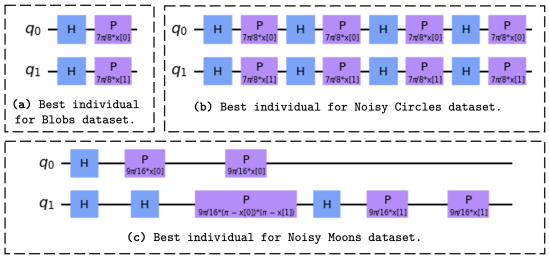

Figure 6: The Pauli feature map circuits produced by the best individual of the GA for the synthetic datasets. Detailed parameters of each circuit can be observed in Table 5.

The best Pauli circuits for the synthetic datasets are mostly dominated by the Z expansion sub-circuit in their components, while there is no Y expansion sub-circuit appears on the circuits. This is also consistent with the fact that the Y Pauli feature map performs very poorly on these 3 datasets compared with other baselines. However, this might not be a general fact since we will see later on the next subsection that the Y expansion sub-circuit would play a significant role in a better classification performance on certain datasets.

5.2. Classification on Real Datasets

For real datasets, the classification results can be reviewed in Table 6. Similarly, bold scores denote the overall best metric scores for each dataset while the italic represents the best among each baseline group. The Pauli feature map circuits encoded by the best individual of the GA for each dataset are presented in Figure 7 with each detailed description presented in Table 5.

From the metrics presented there, it is evident that the model enhanced by the GA-generated quantum feature maps generally works the best among all other baseline models. While on iris dataset, the best performance of each group of baselines is no worse than the GA-generated one, on the other three datasets, the quantum-GA kernel properly dominates all the baseline kernels.

Table 4
Classification Evaluation on Synthetic Datasets (using Accuracy and F1 Score)

	Metric Score	ACC			F1		
Type	Model \Dataset	blobs	noisy_circles	noisy_moons	blobs	noisy_circles	noisy_moons
classical	SVC linear	***1.0000***	0.3500	0.9500	***1.0000***	0.3484	0.9500
	SVC poly	***1.0000***	0.8000	0.9250	***1.0000***	0.7980	0.9249
	SVC rbf	***1.0000***	*0.9000*	***1.0000***	***1.0000***	*0.9000*	***1.0000***
quantum	QSVC X	***1.0000***	0.7500	0.9250	***1.0000***	0.7396	*0.9249*
	QSVC Y	0.4750	0.4750	0.4750	0.3220	0.3220	0.3220
	QSVC Z	***1.0000***	*0.8750*	*0.9250*	***1.0000***	*0.8743*	*0.9249*
	QSVC ZZ	***1.0000***	0.8000	*0.9000*	***1.0000***	0.7954	0.9000
quantum_GA	GA-QSVC	***1.0000***	***0.9500***	***1.0000***	***1.0000***	***0.9499***	***1.0000***

Table 5
Best Individuals Circuit Parameters for all Datasets

group	dataset	pauli_sequence	num_reps	entanglement	alpha
Synthetic	noisy_circles	['Z']	4	circular	2.748894
Synthetic	noisy_moons	['XI', 'Z', 'Z']	2	reverse_linear	1.767146
Synthetic	blobs	['Z']	1	circular	1.374447
Real Data	iris	['Z', 'IIIZ']	2	circular	1.767146
Real Data	blood	['YI', 'IZ']	1	circular	1.963495
Real Data	irish	['Z', 'Y', 'Z', 'Z']	4	circular	0.196350
Real Data	veteran	['IY', 'YI', 'IX']	1	reverse_linear	0.981748

We observe that the existence of categorical features on a dataset could give a significant discrimination between the classical and quantum feature maps. On the all-numerical-valued features such as blood and iris datasets, it seems that the quantum baselines do not significantly outperform the classical kernels while the one assisted by GA could slightly perform better. On the other hand, when there are categorical features on the dataset, which are the Irish and Veteran datasets, the quantum-GA feature maps give a significantly better result than all classical baselines. In particular, the occurrence of the Y expansion sub-circuit plays a considerable role to achieve a better score, for example in veteran dataset.

From these results, apparently there is no definitive generic choice of Pauli sequence which could fit all types of dataset. There are better alternatives of Pauli sequence combination rather than ZZ feature map. While at a glance, the choice of Z expansion sub-circuit may seem to be ubiquitous on the best choice of many datasets, it is not an absolute fact. For example, the best circuit of the veteran dataset possesses no Z expansion sub-circuit at all. Hence, the automatic nature of this evolutionary

Table 6
Classification Evaluation on Some Real Datasets (using Accuracy and F1 Score)

	Metric Score	ACC				F1			
Type	Model \Dataset	blood	iris	irish	veteran	blood	iris	irish	veteran
classical	SVC linear	0.7333	***0.9667***	*0.8298*	*0.7143*	0.4673	***0.9585***	*0.8295*	*0.6190*
	SVC poly	*0.7400*	0.9333	0.7553	*0.7143*	0.4491	0.9190	0.7551	0.5130
	SVC rbf	*0.7400*	***0.9667***	0.7660	0.6786	*0.4706*	***0.9585***	0.7621	0.4043
quantum	QSVC X	*0.7400*	0.9333	0.7660	0.6786	*0.4491*	0.9190	0.7621	0.4043
	QSVC Y	0.7333	0.2000	0.5426	***0.7500***	0.4231	0.1111	0.3517	*0.4286*
	QSVC Z	*0.7400*	***0.9667***	0.7660	0.6786	*0.4491*	***0.9585***	0.7621	0.4043
	QSVC ZZ	0.7333	0.8333	*0.9043*	0.7143	0.4463	0.8216	*0.9042*	0.4167
quantum_GA	GA-QSVC	***0.7600***	***0.9667***	***0.9149***	***0.7500***	***0.6153***	***0.9585***	***0.9149***	***0.6040***

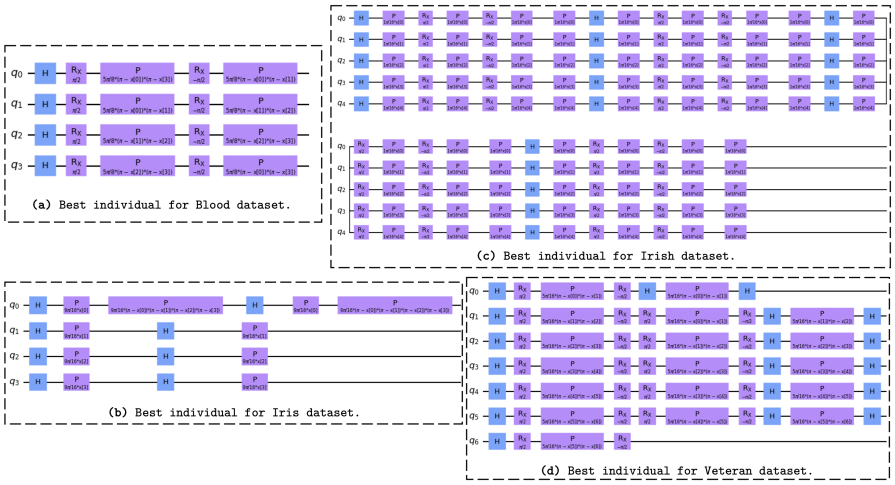

Figure 7: The Pauli feature map circuits produced by the best individual of the GA realization for the real datasets. Detailed parameters of each circuit can be observed in Table 5

design of feature map could provide a generic way to obtain the best circuit that suits the specific dataset without falling into overfitting issue. Moreover, we can observe that all best circuits has no entanglement and also there are diverse range of alpha and repetitions parameters. This result is consistent with some other works [26, 39].

6. Conclusion

In this work, we have explored the design of quantum feature maps optimization for the quantum kernel classification using the evolutionary algorithm framework. In all choices of datasets, the evolutionary design has the best accuracy. In fact, the GA-quantum feature map tends to perform significantly better when the dataset has categorical features, which is not a natural habitat of the classical kernels. Overall, the best feature map obtained by the evolutionary algorithm tends not to converge into specific characteristics. This marks the significance of this method, which is able to find diverse designs of feature map customized to the nature of the dataset.

Several possible improvements are left for future works. Further investigation is needed on how the quantum kernel particularly tends to perform better in datasets with categorical features. Moreover, although currently the entanglement type is only encoded using 1 character among the 4-ary genetic code, some modifications may contribute significantly to the complexity of the circuit. For the next work, it is possible to refine the style of the entanglement into many other options so that the small changes of the digits would not lead to huge difference in circuit complexity. Furthermore, this scheme can also be implemented in real quantum computers environment when the technology is ready. Finally, this scheme of evolutionary feature map generation can also be applied in other quantum machine learning optimizations.

Acknowledgments

This work was funded by PT Lancs Arche Consumma (MOU.003/CIT/XI/2022) and PT Astra International TBK - TSO (MOU.002/CIT/XI/2022).

References

[1] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, S. Lloyd, Quantum machine learning, Nature 549 (2017) 195–202.
[2] M. Schuld, I. Sinayskiy, F. Petruccione, An introduction to quantum machine learning, Contemporary Physics 56 (2015) 172–185.
[3] Y. Zhang, Q. Ni, Recent advances in quantum machine learning, Quantum Engineering 2 (2020) e34.
[4] Y. Dang, N. Jiang, H. Hu, Z. Ji, W. Zhang, Image classification based on quantum k-nearest-neighbor algorithm, Quantum Inf. Process. 17 (2018) 1–18.
[5] S. Lu, S. L. Braunstein, Quantum decision tree classifier, Quantum Inf. Process. 13 (2014) 757–770.
[6] E. Farhi, H. Neven, Classification with quantum neural networks on near term processors, arXiv preprint arXiv:1802.06002 (2018).
[7] S. Lloyd, M. Mohseni, P. Rebentrost, Quantum principal component analysis, Nat. Phys. 10 (2014) 631–633.

[8] I. Kerenidis, J. Landman, A. Luongo, A. Prakash, q-means: A quantum algorithm for unsupervised machine learning, Advances in neural information processing systems 32 (2019).

[9] I. Cong, S. Choi, M. D. Lukin, Quantum convolutional neural networks, Nat. Phys. 15 (2019) 1273–1278.

[10] C. Zoufal, A. Lucchi, S. Woerner, Quantum generative adversarial networks for learning and loading random distributions, npj Quantum Inf. 5 (2019) 103.

[11] M. Schuld, A. Bocharov, K. M. Svore, N. Wiebe, Circuit-centric quantum classifiers, Physical Review A 101 (2020) 032308.

[12] K. Mitarai, M. Negoro, M. Kitagawa, K. Fujii, Quantum circuit learning, Phys. Rev. A 98 (2018) 032309.

[13] S. Theodoridis, K. Koutroumbas, Pattern recognition, Elsevier, 2006.

[14] C. Cortes, V. N. Vapnik, Support-vector networks, Machine Learning 20 (2004) 273–297.

[15] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, J. M. Gambetta, Supervised learning with quantum-enhanced feature spaces, Nature 567 (2019) 209–212.

[16] M. Schuld, N. Killoran, Quantum machine learning in feature hilbert spaces, Physical review letters 122 (2019) 040504.

[17] M. Schuld, Supervised quantum machine learning models are kernel methods, arXiv preprint arXiv:2101.11020 (2021).

[18] Y. Suzuki, H. Yano, Q. Gao, S. Uno, T. Tanaka, M. Akiyama, N. Yamamoto, Analysis and synthesis of feature map for kernel-based quantum classifier, Quantum Machine Intelligence 2 (2020) 1–9.

[19] J. Mancilla, C. Pere, A preprocessing perspective for quantum machine learning classification advantage in finance using nisq algorithms, Entropy 24 (2022) 1656.

[20] M. Grossi, N. Ibrahim, V. Radescu, R. Loredo, K. Voigt, C. Von Altrock, A. Rudnik, Mixed quantum–classical method for fraud detection with quantum feature selection, IEEE Transactions on Quantum Engineering 3 (2022) 1–12.

[21] L. Davis, Handbook of genetic algorithms (1991).

[22] G. Yasuda, H. Takai, Sensor-based path planning and intelligent steering control of nonholonomic mobile robots, in: IECON'01. 27th Annual Conference of the IEEE Industrial Electronics Society (Cat. No. 37243), volume 1, IEEE, 2001, pp. 317–322.

[23] A. Madureira, C. Ramos, S. do Carmo Silva, A coordination mechanism for real world scheduling problems using genetic algorithms, in: Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600), volume 1, IEEE, 2002, pp. 175–180.

[24] M. Gong, Y.-H. Yang, Multi-resolution stereo matching using genetic algorithm, in: Proceedings IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV 2001), IEEE, 2001, pp. 21–29.

[25] H. Chiroma, S. Abdulkareem, A. Abubakar, T. Herawan, Neural networks optimization through genetic algorithm searches: a review, Appl. Math. Inf. Sci 11 (2017) 1543–1564.

[26] S. Altares-López, A. Ribeiro, J. J. García-Ripoll, Automatic design of quantum feature maps, Quantum Science and Technology 6 (2021) 045015.

[27] S. Altares-López, J. J. García-Ripoll, A. Ribeiro, Autoqml: Automatic generation and training of robust quantum-inspired classifiers by using genetic algorithms on grayscale images, arXiv preprint arXiv:2208.13246 (2022).

[28] M. Incudini, F. Martini, A. Di Pierro, Structure learning of quantum embeddings, arXiv preprint arXiv:2209.11144 (2022).

[29] B.-S. Chen, J.-L. Chern, Genetically autogenerated quantum feature maps, arXiv preprint arXiv:2207.11449 (2022).

[30] N. Nguyen, K.-C. Chen, Quantum embedding search for quantum machine learning, IEEE Access 10 (2022) 41444–41456.

[31] E. Torabian, R. V. Krems, Compositional optimization of quantum circuits for quantum kernels of support vector machines, Physical Review Research 5 (2023) 013211.

[32] Qiskit contributors, Qiskit: An open-source framework for quantum computing, 2023. doi:10.5281/zenodo.2573505.

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Mach Learn Res. 12 (2011) 2825–2830.

[34] M. Feurer, J. N. Van Rijn, A. Kadra, P. Gijsbers, N. Mallik, S. Ravi, A. Müller, J. Vanschoren, F. Hutter, Openml-python: an extensible python api for openml, Mach Learn Res. 22 (2021) 4573–4577.

[35] R. A. Fisher, The use of multiple measurements in taxonomic problems, Ann. Eugen. 7 (1936) 179–188.

[36] I.-C. Yeh, K.-J. Yang, T.-M. Ting, Knowledge discovery on rfm model using bernoulli sequence, Expert Systems with Applications 36 (2009) 5866–5871.

[37] V. Greaney, T. Kellaghan, Equality of Opportunity in Irish Schools: a longitudinal study of 500 students, Educational Company, 1984.

[38] J. D. Kalbfleisch, R. L. Prentice, The statistical analysis of failure time data, John Wiley & Sons, 2011.

[39] A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster, J. I. Latorre, Data re-uploading for a universal quantum classifier, Quantum 4 (2020) 226.