Quantum Variational Multi-Class Classifier for the Iris Data Set

Ilya Piatrenka[0000−0003−0917−4798] and Marian Rusek[0000−0002−9978−7530]

Institute of Information Technology
Warsaw University of Life Sciences — SGGW
ul. Nowoursynowska 159, 02-776 Warsaw, Poland
marian_rusek@sggw.edu.pl

Abstract. Recent advances in machine learning on quantum computers have been made possible mainly by two discoveries. Mapping the features into exponentially large Hilbert spaces makes them linearly separable — quantum circuits perform linear operations only. The parameter-shift rule allows for easy computation of objective function gradients on quantum hardware — a classical optimizer can then be used to find its minimum. This allows us to build a binary variational quantum classifier that shows some advantages over the classical one. In this paper we extend this idea to building a multi-class classifier and apply it to real data. A systematic study involving several feature maps and classical optimizers as well as different repetitions of the parametrized circuits is presented. The accuracy of the model is compared both on a simulated environment and on a real IBM quantum computer.

Keywords: Quantum Computing · Hybrid Classical-Quantum Algorithms · Variational Quantum Classifier · Artificial Intelligence · Machine Learning.

1 Introduction

Classical machine learning techniques have made great strides in the past decade, enabled in large part by the availability of sufficiently powerful hardware. For example, the success of neural networks based on deep learning was possible in part because of powerful parallel hardware consisting of clusters of graphical processors [30]. Maybe the existence of quantum hardware might enable further advances in the field, making the development of new machine learning algorithms possible [8]. However, ideal fault-tolerant quantum computers [27] are still not available today. What we have are Noisy Intermediate Scale Quantum computers (NISQ) [5].

There are two major paradigms in the quantum computer world: quantum annealing and gate-based computation. Both allow for solving optimization problems by finding a minimum of certain objective functions. Thus, they can both potentially be used for machine learning tasks. D-Wave Systems is a major vendor of superconducting quantum annealing machines. Their recent machines

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08760-8_21

offer up to 5640 qubits [7, 6]. A more general gate-based architecture potentially allows for execution of arbitrary algorithms not restricted to optimization problems. Recently IBM unveiled a 127 qubit gate-based machine operating on transmons[1]. Alternative physical realizations of gate-based quantum computers involve photonic systems and ion-traps[19]. Gate-based quantum circuits can be created using several open source software tools [10]. In this paper, a gate-based IBM 5 qubit transmon computer is used and programmed in Python with Qiskit library to solve a supervised machine learning problem.

To mitigate the influence of decoherence errors in NISQ computers, quantum circuits with limited depth are needed. This can be achieved when the work of a quantum computer is supplemented by a classical one. This yields hybrid quantum-classical computations. In a variational setting, the quantum circuit $U(\boldsymbol{\theta})$ is parametrized by a set of numbers $\boldsymbol{\theta}$. At each step of the algorithm, a cost function $C(\boldsymbol{\theta})$ is evaluated based on multiple measurements of the quantum circuit for a given input state $|\psi_0\rangle$. Then, an optimization algorithm running on a classical computer is used to update the parameters $\boldsymbol{\theta}$ to minimize the cost function $C(\boldsymbol{\theta})$. The final state $U(\boldsymbol{\theta}_{\text{opt}})|\psi_0\rangle = |\psi(\boldsymbol{\theta}_{\text{opt}})\rangle$ is a superposition that contains the solution to the problem with a high probability amplitude. Variational algorithms were introduced in 2014, with the variational eigensolver [21].

Another well known example of such a variational approach is the Quantum Approximate Optimization Algorithm (QAOA) [9]. It allows us to find approximate solutions to the Quadratic Unconstrained Binary Optimization (QUBO) problem. The final state $|\psi(\boldsymbol{\theta}_{\text{opt}})\rangle$ measured in the computational basis gives with high probability the bit string corresponding to the solution of the specific problem (e.g., MaxCut). The influence of noise on the performance of this algorithm on IBM quantum computers was studied in [2].

The interplay between quantum computing and machine learning has attracted a lot of attention in recent years. The use of an exponentially large feature space of dimension 2^n where n is the number of qubits that is only efficiently accessible on a quantum computer provides a possible path to quantum advantage. In [12] two algorithms have been proposed that process classical data and use the quantum state Hilbert space as the feature space. Both algorithms solve a problem of supervised learning: the construction of a binary classifier. One method, the quantum variational classifier, uses a variational quantum circuit with a classical optimizer to classify the data. The other method, a quantum kernel estimator, estimates the kernel function on the quantum computer and optimizes a classical Support Vector Machine (SVM). In this paper we extend the first method to encompass the multi-class classification case.

In [1] the effective dimension measure based on the Fisher information has been proposed. It can be used to assess the ability of a machine learning to train. It was found that a class of quantum neural networks is able to achieve a considerably higher capacity and faster training ability than comparable classical feedforward neural networks. A higher capacity is captured by a higher effective

[1] https://research.ibm.com/blog/127-qubit-quantum-processor-eagle

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08760-8_21

dimension, whereas faster training implies that a model will reach a lower loss value than another comparable model for a fixed number of training iterations. This suggests an advantage for quantum machine learning, which was demonstrated both numerically and on real quantum hardware using the Iris Data Set [3]. In this paper we use this data set not only to check the model training speed but also its accuracy.

This paper consists of 5 main sections. In Section 2 the details of a variational hybrid quantum-classical model for multi-class classification are described. In Section 3, the results from training this model on the training subset of the Iris Data Set are presented and its accuracy is tested on test subset. Both a classical Qiskit simulator as well as a real IBM quantum computer are used. In Section 4 we discuss these results. In Section 5 we finish with some conclusions.

2 Model

2.1 Parametrized quantum circuits

Parameterized quantum circuits, where the gates are defined by tunable parameters, are fundamental building blocks of near-term quantum machine learning algorithms. They are used for two things:

- To encode the data, where the parameters are determined by the data being encoded.
- As a quantum model, where the parameters are determined by an optimization process.

As all quantum gates used in a quantum circuit are unitary, a parameterized circuit itself can be described as a unitary operation on qubits.

How do we choose parameterized quantum circuits that are good candidates for quantum machine learning applications? First, they need them to generalize well. This means that the circuit should be able to map to a significant subset of the states within the output Hilbert space. To avoid being easy to simulate on a classical computer, the circuit should also entangle qubits. In [28], the authors propose the measures of expressibility and entangling capability to discriminate between different parameterized quantum circuits. A strong correlation between classification accuracy and expressibility, and a weak correlation between classification accuracy and entangling capability has been found [13]. Note, that the use of parametrized quantum circuits with a high expressibility in supervised machine learning scenarios may lead to overfitting [24].

In the current era of near term quantum computing, there is limited error correction or mitigation and limited qubit connectivity. To accommodate these device constraints, in [14] a class of hardware efficient parameterized circuits was introduced. They are built from layers consisting of one two-qubit entangling gate and up to three single-qubit gates. Also, a particular qubit connection topology is used. These results have been extended in [12] where the authors introduced a general parameterized circuit, which includes layers of Hadamard

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08760-8_21

gates interleaved with entangling blocks, and rotation gates. This unitary was chosen because it is classically difficult to compute, but tractable on near term quantum hardware. It can be used to encode data.

Data encoding Data representation is crucial for the success of machine learning models. For classical machine learning, the problem is how to represent the data numerically, so that it can be best processed by a classical machine learning algorithm. For quantum machine learning, this question is similar, but more fundamental: how to represent and efficiently input the data into a quantum system, so that it can be processed by a quantum machine learning algorithm. This is usually referred to as data encoding and is a critical part of quantum machine learning algorithms that directly affect their power.

One of the simplest data encoding methods is angle encoding. In this case, the number of features is equal to the number of qubits and the features are encoded into the rotation angles of qubits. This method only encodes one datapoint at a time. It does, however, only use a constant depth quantum circuit, making it amenable to current quantum hardware. For a general unitary operator from [12] this method corresponds to the parameters $k = 1, P_0 = Z$. In Qiskit the corresponding function is called `ZFeatureMap`. For $k = 2, P_0 = Z, P_1 = ZZ$ we get the Qiskit `ZZFeatureMap` circuit, which contains layers of Hadamard gates interleaved with qubit rotations and entangling blocks. These data encoding circuits were used in [1]. These authors call them "easy quantum model" and "quantum neural network" respectively.

In this paper, in addition to the feature map circuits described above, a slightly more sophisticated circuit is used. It is depicted in Fig. 1 and differs from the `ZZFeatureMap` by adding qubit rotations along Y axis. Using notation from [12] it corresponds to the parameters $k = 3, P_0 = Z, P_1 = Y, P_2 = ZZ$ and is generated using the Qiskit `PauliFeatureMap` function. It only encodes a datapoint $\boldsymbol{x}$ of 3 features, despite having 9 parameterized gates.

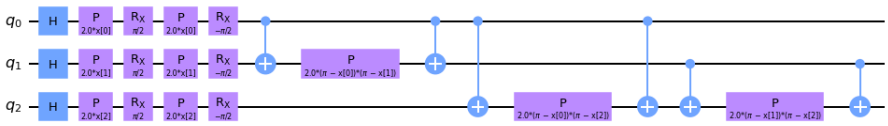

Fig. 1. `PauliFeatureMap` data encoding circuit for 3 qubits. H, P, and R_X are Hadamard, phase shift, and X rotation gates; x is the feature vector.

Quantum model The Qiskit `RealAmplitudes` circuit is a heuristic trial wave function used as Ansatz in chemistry applications or quantum model classification circuits in machine learning. The circuit consists of alternating layers of

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08760-8_21

R_Y rotations and $CNOT$ entanglements. We use it in our paper with different numbers of repetitions. The number of variational parameters θ_j is equal to $(r+1)n$, where r is the number of repetitions of this circuit, and n is the number of qubits. This circuit can be regarded as a special case of the Qiskit `TwoLocal` circuit. The prepared trial quantum states will only have real amplitudes.

2.2 Quantum variational classification

Quantum supervised machine learning is the task of learning the parameters $\boldsymbol{\theta}$ of a variational circuit that maps an input feature vector $\boldsymbol{x}$ to an output $\hat{y}$ based on example training input-output pairs $(\boldsymbol{x}, y)$. The accuracy of the model can then be checked using a set of testing examples. In the case of a classification problem that asks us to assign data into specific categories, the outputs $\hat{y}$ are category numbers.

Label assignment Such a variational quantum classifier algorithm was introduced by multiple groups in 2018 [26]. For a binary classification problem, with binary output labels, the measured expectation value of $Z^{\otimes n}$ can be interpreted as the output of a classifier. The average is calculated over several shots of the algorithm. It can be shown, that this is equivalent to measuring the average value of Z acting on one qubit only.

Let us now extend this procedure to a multi-class classification. For a given feature vector $\boldsymbol{x}$ and variational parameters θ the output of the parmetrized circuit is measured in the computational basis. Thus the output of each shot of the algorithm is a bitstring $\boldsymbol{b}$, where $b_i \in \{0, 1\}$. The predicted category number can then be calculated as:

$$\hat{y} = \sum_{i=0}^{n-1} b_i \pmod{K} \tag{1}$$

where K is the number of categories. Note, that for two categories $K = 2$, Eq. (1) is just a parity function. Thus, in this case, the classification result is equivalent to the previously described one.

After measuring the same circuit again, a different set of bits $\boldsymbol{b}$ is obtained. Thus, the output from multiple shots of the experiment are the probabilities p_k of belonging the feature vector $\boldsymbol{x}$ to different categories $k = 0, \ldots K-1$.

Cost function To compare the different parameters $\boldsymbol{\theta}$, we need to score them according to some criteria. We call the function that scores our parameters the "cost" or "loss" function, as bad results are more costly.

The combination of softmax with cross-entropy is a standard choice for a cost function to train neural network classifiers [22]. It measures the cross-entropy between the given true label y and the output of the neural network $\hat{y}$. The network's parameters are then adjusted to reduce the cross-entropy via back-propagation.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08760-8_21

Cross-entropy measures the difference between two probability distributions q_k and p_k:

$$-\sum_{k=0}^{K-1} q_k \log_2 p_k \tag{2}$$

In our case, the target distribution $q_k = \delta_{ky}$ is just a delta function. Therefore, Eq. (2) becomes:

$$-\log_2 p_y \tag{3}$$

where p_k is the probability of measuring label k. In order to obtain the cost function, we sum expressions from Eq. (3) for all points from the training set. This calculation is done by Qiskit `OneHotObjectiveFunction`.

Similarly, the authors [1] also use a cross-entropy cost function for a binary classification problem.

2.3 Training parameterized quantum circuits

Like classical models, we can train parameterized quantum circuit models to perform data classification tasks. The task of supervised learning can be mathematically expressed as the minimization of the objective function, with respect to the parameter vector $\boldsymbol{\theta}$. In the training phase, the quantum computer calculates the predicted labels. The classical computer compares them to the provided labels and estimates the success of our predictions using the objective function. Based on this cost, the classical computer chooses another value for θ using a classical optimization algorithm. This new value is then used to run a new quantum circuit, and the process is repeated until the objective function stabilizes at a minimum. Note, that finding a global minimum is not an easy task, as the loss landscape can be quite complicated [18].

There are many different types of algorithms that we can use to optimise the parameters of a variational circuit: gradient-based, evolutionary, and gradient-free methods. In this paper, we will be using gradient-based methods. For circuit-based gradients, there's a very nice theoretical result — the parameter shift rule [25]. It gives a very easy formula for calculating gradients on the quantum circuit itself. It is very similar to the equation for finite difference gradients except for the difference is not infinitesimally small.

In vanilla gradients, the Euclidean distance between the points is used, which doesn't take the loss landscape into account. With Quantum Natural Gradients [29], a distance that depends on the model based on Quantum Fisher Information is used instead. It allows to transform the steepest descent in the Euclidean parameter space to the steepest descent in the model space and thus approaches the target faster than vanilla gradient descent. However, this comes at the cost of needing to evaluate many more quantum circuits.

Simultaneous Perturbation Stochastic Approximation (SPSA) [11] is an optimization technique where to reduce the number of evaluations we randomly sample from the gradient. Since we don't care about the exact values but only

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08760-8_21

about convergence, an unbiased sampling works on average quite well. In practise, while the exact gradient follows a smooth path to the minimum, SPSA will jump around due to the random sampling, but eventually it will converge, given the same boundary conditions as the gradient.

In this paper, in addition to SPSA, Constrained Optimization by Linear Approximation (COBYLA) [23], and Sequential Least SQuares Programming (SLSQP) [15] optimization algorithms are used.

3 Results

In this section we use the data from the Iris Data Set [3]. It contains 3 classes: *Iris-setosa*, *Iris-versicolor*, and *Iris-virginica*; 150 samples (50 for each class), and 4 features: *petal width*, *petal length*, *sepal width*, and *sepal length*. This data set was split into the training and test sets using `shuffle` and `train_test_split` functions from the `scikit` library. The test set constitutes 30% of the total data set, i.e, it contains 45 samples.

To obtain the results presented below, Python version 3.9.9 was used together with libraries listed in Table 1.

Table 1. Libraries used and their versions.

Library	Version
Qiskit	0.31.0
Qiskit-Machine-Learning	0.2.1
Qiskit-Terra	0.18.3
Qiskit-Aer	0.9.1
NumPy	1.21.1

3.1 Qiskit Simulation

As described in Section 2.1 we have considered circuits with 3 types of data encoding layers: `ZFeatureMap`, `ZZFeatureMap`, and `PauliFeatureMap`. One parametrized circuit `RealAmplitudes` was used. Both the data encoding and parametrized circuits were repeated 1, 2, and 4 times. This gives the total number of 27 variational classifier circuits. Each circuit was run and measured 1024 times.

Training phase In the training phase 3 classical optimizers from Section 2.3: SPSA, SLSQP, and COBYLA were used to tune the variational parameters on the Qiskit simulator. They were limited to 200 learning iterations each, but SLSQP always finished its work before 100 iterations. Sometimes, COBYLA or SPSA also finish their work earlier.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08760-8_21

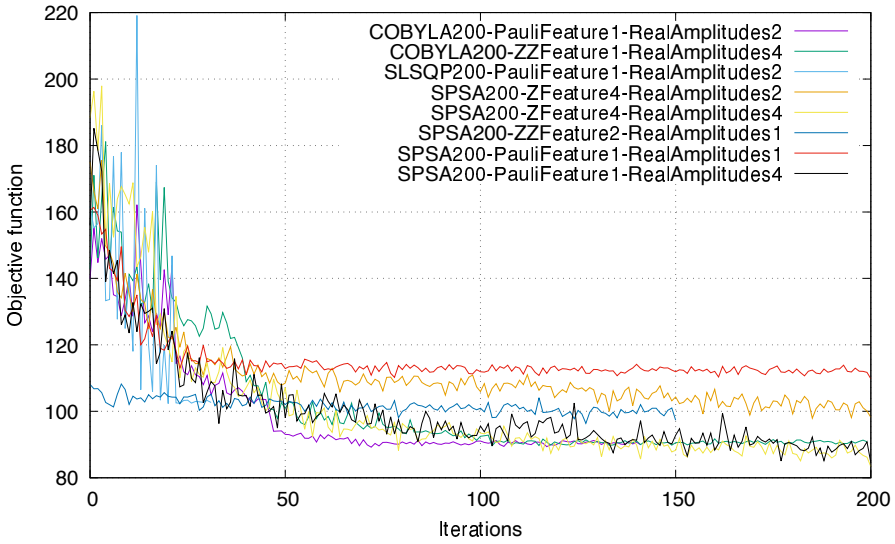

Fig. 2. Objective function value versus number of iterations as calculated by the Qiskit simulator. The cases ploted are marked in bold in Table 2.

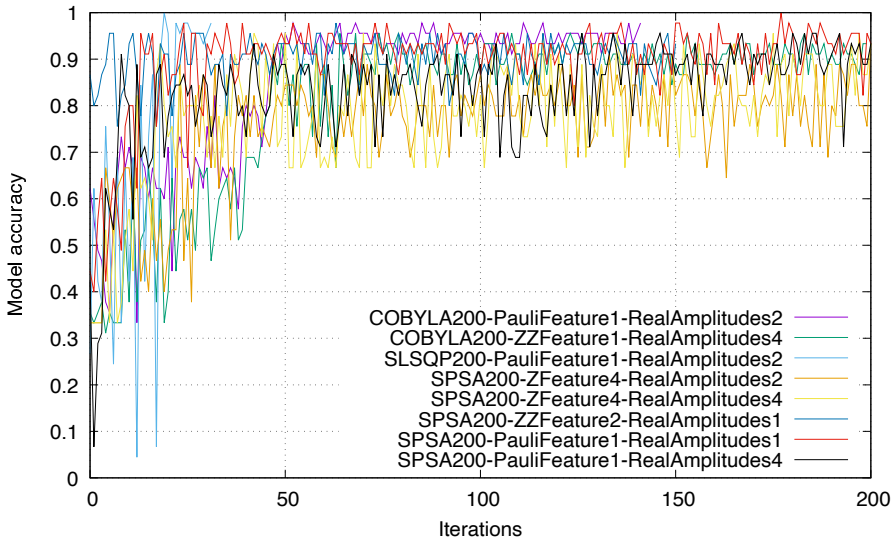

Fig. 3. Model accuracy versus number of iterations as calculated by the Qiskit simulator. The cases ploted are marked in bold in Table 2.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08760-8_21

Example results for selected cases (that also performed well on a real quantum computer) are shown in Fig. 2. The fastest case to stabilize is *SLSQP200-PauliFeature1-RealAmplitudes2* (blue line), which reached its target value of the objective function already after 31 iterations. Next come *COBYLA200-PauliFeature1-RealAmplitudes2* (violet line) and *SPSA200-ZZFeature2-RealAmplitudes1* (dark blue line), where 141 and 151 iterations were needed respectively. The result of *COBYLA200-PauliFeature1-RealAmplitudes2* (violet line) is among one of the lowest, was reached very fast, and is very stable with the number of iterations.

In general, if the loss landscape is fairly flat, it can be difficult for the optimization method to determine which direction to search. This situation is called a barren plateau [20]. For all 81 cases considered here, this phenomenon was not observed. The optimization algorithms were always able to stabilize at some minimum of the objective loss function.

Accuracy check Model accuracy is defined as the ratio of the correctly predicted classes to the number of points in the test set. The predicted class k is the one observed with the highest probability p_k in 1024 repetitions of the experiment. The results of its calculation on the test set are shown in Fig. 3. We see that the accuracy of the models optimized by SPSA oscillates a lot with the algorithm iteration number. The results of SLSPQ and COBYLA are by far more stable. The cases *COBYLA200-PauliFeature1-RealAmplitudes2* (violet line) and *SLSQP200-PauliFeature1-RealAmplitudes2* (blue line) seem to provide the highest and most stable accuracy. All the models shown here are able to score above 90%, therefore no overfitting is observed.

3.2 IBM Quito Quantum Computer

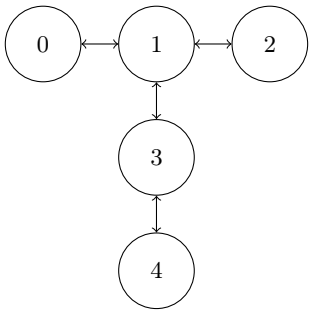

Fig. 4. Qubit connection topology of the IBM Quito quantum computer.

In this section we repeat the experiments on a real 5 qubit IBM Quito quantum computer. Its qubit connection topology is presented in Fig. 4. We need

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08760-8_21

only 4 of them. Note that both the data encoding circuits and the parametrized circuit from Section 2.1 use an all-to-all qubit entanglement scheme (except for the `ZFeatureMap` where no entanglement is used). Therefore, physical qubits 0–3 are the optimal choice to map the 4 logical qubits of our circuits. By choosing this subset, each pair of qubits will be separated by at most one additional qubit. On the other hand, if qubits 1–4 were chosen, then some pairs of qubits would be separated by two additional qubits.

We have sorted the table of 81 results from Section 3.1 by descending accuracy. The first 32 cases have been taken and launched on qubits 0–3 of the IBM Quito quantum computer. For variational paramaters the values computed during Qiskt simulation were substituted. No training on the quantum hardware was performed due to limited computer availability. The results of this experiment are presented in Table 2. Only 8 models scored accuracy above 60%. They are marked in bold (these cases were used in Figs. 2 and 3). For some models, the accuracy dropped below 33.3%, i.e., below random class assignment. Models with the lowest numbers of $CNOT$ gates generally achieve the best results: *SPSA200-ZFeature4-RealAmplitudes4*, 24 gates, accuracy 73,3%; *SPSA200-PauliFeature1-RealAmplitudes1*, 18 gates, accuracy 71,1%; *SPSA200-ZFeature4-RealAmplitudes2*, 12 gates; accuracy 68,9%. This will be discussed in the next section.

Note, that the choice of an classical optimizer is very important for the quantum algorithm performance. The accuracy of *SPSA200-ZFeature4-RealAmplitudes2* dropped to 33.3% but *SPSA200-ZFeature4-RealAmplitudes4* performed much better — in this case the accuracy dropped only to 68.9%.

4 Discussion

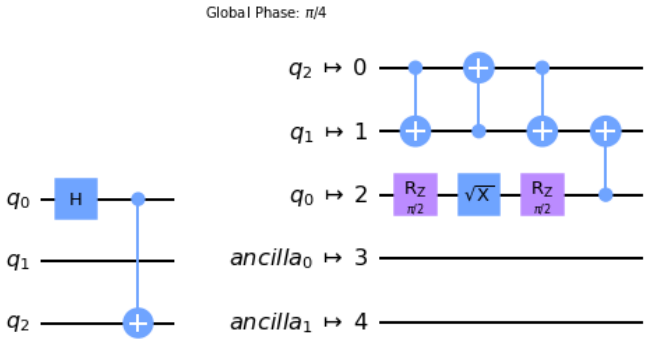

Fig. 5. Simple quantum circuit for Bell state production before (left) and after (right) transpilation.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08760-8_21

Table 2. Model accuracy comparison between Qiskit simulation and execution on a real quantum computer.

Variational quantum circuit and optimization algorithm used for its training	Model accuracy on a Qiskit simulator	Model accuracy on a real quantum computer	Number of $CNOT$ gates before transpilation	Number of $CNOT$ gates after transpilation	Number of single qubit gates
COBYLA200-PauliFeature1-RealAmplitudes2	97.7%	**66.6%**	24	60	170
SLSQP200-ZZFeature4-RealAmplitudes1	97.7%	31.1%	54	135	184
SLSQP200-PauliFeature1-RealAmplitudes2	97.7%	**66.6%**	24	60	170
SLSQP200-PauliFeature1-RealAmplitudes4	97.7%	48.9%	36	88	226
SLSQP200-PauliFeature2-RealAmplitudes1	97.7%	57.8%	30	71	196
COBYLA200-ZZFeature2-RealAmplitudes4	95.5%	48.9%	48	129	220
COBYLA200-PauliFeature4-RealAmplitudes2	95.5%	33.3%	60	144	368
SPSA200-ZZFeature1-RealAmplitudes4	95.5%	55.6%	36	91	190
SPSA200-ZZFeature2-RealAmplitudes4	95.5%	37.8%	48	129	220
SPSA200-ZZFeature4-RealAmplitudes4	95.5%	35.6%	72	180	280
SLSQP200-ZZFeature2-RealAmplitudes2	95.5%	48.9%	36	101	164
SLSQP200-ZZFeature4-RealAmplitudes2	95.5%	33.3%	60	152	224
SLSQP200-ZZFeature2-RealAmplitudes4	95.5%	33.3%	48	129	220
SLSQP200-ZZFeature4-RealAmplitudes4	95.5%	35.6%	72	180	280
SLSQP200-PauliFeature4-RealAmplitudes2	95.5%	33.3%	60	144	368
SLSQP200-PauliFeature4-RealAmplitudes4	95.5%	33.3%	72	172	424
COBYLA200-ZFeature4-RealAmplitudes4	93.3%	33.3%	24	63	256
COBYLA200-ZZFeature1-RealAmplitudes4	93.3%	**66.7%**	36	91	190
COBYLA200-PauliFeature2-RealAmplitudes4	93.3%	40.0%	48	116	292
COBYLA200-PauliFeature4-RealAmplitudes4	93.3%	33.3%	72	172	424
SPSA200-ZFeature4-RealAmplitudes2	93.3%	**68.9%**	12	35	200
SPSA200-ZFeature4-RealAmplitudes4	93.3%	**73.3%**	24	63	256
SPSA200-ZZFeature2-RealAmplitudes1	93.3%	**62.2%**	30	84	124
SPSA200-ZZFeature4-RealAmplitudes1	93.3%	31.1%	54	135	184
SPSA200-PauliFeature1-RealAmplitudes1	93.3%	**71.1%**	18	43	130
SPSA200-PauliFeature4-RealAmplitudes1	93.3%	42.2%	54	127	328
SPSA200-PauliFeature4-RealAmplitudes2	93.3%	42.2%	60	144	368
SPSA200-PauliFeature1-RealAmplitudes4	93.3%	**68.9%**	36	88	226
SPSA200-PauliFeature4-RealAmplitudes4	93.3%	28.9%	72	172	424
SLSQP200-ZZFeature1-RealAmplitudes4	93.3%	33.3%	36	91	190
SLSQP200-PauliFeature2-RealAmplitudes2	93.3%	26.7%	36	88	236
SLSQP200-PauliFeature2-RealAmplitudes4	93.3%	33.3%	48	116	292
SLSQP200-PauliFeature4-RealAmplitudes1	93.3%	35.5%	54	127	328

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08760-8_21

On a real quantum computer, not every pair of qubits is directly connected. Therefore, executing a two-qubit gate on them leads to the inclusion of additional helper gates to the circuit during the transpilation phase. For example, consider qubits 0 and 2 from Fig. 4. Let us launch on them a simple circuit for Bell state production that consists of one Hadamard gate H and one $CNOT$ gate. It is shown on the left of Fig. 5. During the transpilation phase, this single $CNOT$ gate is changed to four $CNOT$ gates. The result is presented on the right side of Fig. 5. The Hadamard gate H has been changed to 3 single qubit gates: $R_Z(\pi/2)$, $\sqrt{X}$, and $R_Z(\pi/2)$ because the IBM Quito quantum computer has no direct implementation for it. Notice the change of the qubit on which the single qubit gate operates — this is due to the phase-kickback effect [17].

At the moment the experiment from Section 3.2 was performed, the average error of the $CNOT$ gate of the IBM Quito computer was $1.052e-2$ — a detailed error map for this this machine is given on the IBM Quantum Computing web page[2]. The circuits that performed best in Table 2 all have about 60 or less $CNOT$ gates after transpilation.

5 Conclusions

The subject of this study was to implement and study a variational quantum program for supervised machine learning. Multi-class classification of the Iris Data Set was performed without resorting to multiple binary classifications and One-vs-Rest or One-vs-One strategies. Model accuracy of different data encoding circuits and repetitions of the parametrized circuit was analyzed both on a simulator and on a real quantum computer. It seems that one repetition of `PauliFeatureMap` with Z, Y, and ZZ gates; and two repetitions of the `RealAmplitudes` is the candidate that is stable in training and gives good classification accuracy. This circuit consists of 60 $CNOT$ gates after transpilation on the IBM Quito computer and has 170 single qubit gates. 4 features of the data set are mapped onto a 16 dimensional Hilbert space. Encoding circuit parameters depend not only on the features themselves but also on their products. The parametrized circuit (ansatz) has the total number of 12 variational parameters that are trained using a classical optimizer. The best results seem to come from COBYLA and SLSQP. In future research we plan to implement classical artificial intelligence that would be responsible for an optimal circuit choice. There are already some first steps in this direction [16].

In [4] the authors proposed another hybrid classical-quantum variational approach for multi-class classification. Their quantum multi-class classifier is designed with multiple layers of entangled rotational gates on data qubits and ancilla qubits with adjustable parameters. The class number is predicted by measuring the ancilla qubits. Similarly to the `ZFeatureMap` data encoding uses only single-qubit rotations. Their circuit is created using PennyLane and run on the classical simulator. By training it on the Iris Data Set these authors were

[2] https://quantum-computing.ibm.com/services?services=systems

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08760-8_21

able to reach the accuracy of 92.10%. This is lower than the accuracy of all cases from Table 2 where the top accuracy od 97.7% was reached without using any ancilla qubits.

A large discrepancy between the accuracy of the model created on the Qiskit simulator and a real IBM Quito quantum computer has been found. It is due to gate errors and peculiar qubit connection topology. This accuracy drop persists even when the training is done on the quantum computer itself. We were quite surprised by these results because all IBM quantum computers with 7 or more qubits available today have a connection topology consisting of T-like structures similar to that from Fig. 4 and have similar error gates for the $CNOT$ gates. They also utilize the same gate set presented in Sec. 4. In [1] there was absolutely no problem in training a variational binary classifier on the Iris Data Set. The IBM Montreal 27 qubit quantum computer reached lower value of the cost function faster that a classical simulator (model accuracy was not calculated). The possible explanation is that for more qubits available the transpiler was able to choose physical qubits with lower $CNOT$ gate errors. Unfortunately we did't have access to that machine to check this. Another explanation is that these authors used linear qubit connectivity instead of all-to-all. We use the latter one similarly to [4]. In the future, fault tolerant quantum computers or computers with all-to-all qubit connection topology should eliminate these problems.

The code used in this paper as well as some additional results are available from a publicly accessible GitHub repository[3].

References

1. Abbas, A., Sutter, D., Zoufal, C., Lucchi, A., Figalli, A., Woerner, S.: The power of quantum neural networks. Nature Computational Science **1**(6), 403–409 (2021)
2. Alam, M., Ash-Saki, A., Ghosh, S.: Analysis of quantum approximate optimization algorithm under realistic noise in superconducting qubits. arXiv preprint arXiv:1907.09631 (2019)
3. Anderson, E.: The species problem in iris. Annals of the Missouri Botanical Garden **23**(3), 457–509 (1936)
4. Chalumuri, A., Kune, R., Manoj, B.: A hybrid classical-quantum approach for multi-class classification. Quantum Information Processing **20**(3), 1–19 (2021)
5. Córcoles, A.D., Kandala, A., Javadi-Abhari, A., McClure, D.T., Cross, A.W., Temme, K., Nation, P.D., Steffen, M., Gambetta, J.M.: Challenges and opportunities of near-term quantum computing systems. arXiv preprint arXiv:1910.02894 (2019)
6. Dattani, N., Chancellor, N.: Embedding quadratization gadgets on chimera and pegasus graphs. arXiv preprint arXiv:1901.07676 (2019)
7. Dattani, N., Szalay, S., Chancellor, N.: Pegasus: The second connectivity graph for large-scale quantum annealing hardware. arXiv preprint arXiv:1901.07636 (2019)
8. Dunjko, V., Taylor, J.M., Briegel, H.J.: Quantum-enhanced machine learning. Physical review letters **117**(13), 130501 (2016)
9. Farhi, E., Goldstone, J., Gutmann, S.: A quantum approximate optimization algorithm. arXiv preprint arXiv:1411.4028 (2014)

[3] https://github.com/odisei369/quantum_learning_iris

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08760-8_21

10. Fingerhuth, M., Babej, T., Wittek, P.: Open source software in quantum computing. PloS one **13**(12), e0208561 (2018)
11. Gacon, J., Zoufal, C., Carleo, G., Woerner, S.: Simultaneous perturbation stochastic approximation of the quantum fisher information. Quantum **5**, 567 (2021)
12. Havlíček, V., Córcoles, A.D., Temme, K., Harrow, A.W., Kandala, A., Chow, J.M., Gambetta, J.M.: Supervised learning with quantum-enhanced feature spaces. Nature **567**(7747), 209–212 (2019)
13. Hubregtsen, T., Pichlmeier, J., Bertels, K.: Evaluation of parameterized quantum circuits: on the design, and the relation between classification accuracy, expressibility and entangling capability. arXiv preprint arXiv:2003.09887 (2020)
14. Kandala, A., Mezzacapo, A., Temme, K., Takita, M., Brink, M., Chow, J.M., Gambetta, J.M.: Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. Nature **549**(7671), 242–246 (2017)
15. Kraft, D., et al.: A software package for sequential quadratic programming (1988)
16. Kuo, E.J., Fang, Y.L.L., Chen, S.Y.C.: Quantum architecture search via deep reinforcement learning. arXiv preprint arXiv:2104.07715 (2021)
17. Lee, C.M., Selby, J.H.: Generalised phase kick-back: the structure of computational algorithms from physical principles. New Journal of Physics **18**(3), 033023 (2016)
18. Li, H., Xu, Z., Taylor, G., Studer, C., Goldstein, T.: Visualizing the loss landscape of neural nets. Advances in neural information processing systems **31** (2018)
19. Linke, N.M., Maslov, D., Roetteler, M., Debnath, S., Figgatt, C., Landsman, K.A., Wright, K., Monroe, C.: Experimental comparison of two quantum computing architectures. Proceedings of the National Academy of Sciences **114**(13), 3305–3310 (2017)
20. McClean, J.R., Boixo, S., Smelyanskiy, V.N., Babbush, R., Neven, H.: Barren plateaus in quantum neural network training landscapes. Nature communications **9**(1), 1–6 (2018)
21. Peruzzo, A., McClean, J., Shadbolt, P., Yung, M.H., Zhou, X.Q., Love, P.J., Aspuru-Guzik, A., O'brien, J.L.: A variational eigenvalue solver on a photonic quantum processor. Nature communications **5**(1), 1–7 (2014)
22. Qin, Z., Kim, D., Gedeon, T.: Rethinking softmax with cross-entropy: Neural network classifier as mutual information estimator. arXiv preprint arXiv:1911.10688 (2019)
23. Rios, L.M., Sahinidis, N.V.: Derivative-free optimization: a review of algorithms and comparison of software implementations. Journal of Global Optimization **56**(3), 1247–1293 (2013)
24. Salman, S., Liu, X.: Overfitting mechanism and avoidance in deep neural networks. arXiv preprint arXiv:1901.06566 (2019)
25. Schuld, M., Bergholm, V., Gogolin, C., Izaac, J., Killoran, N.: Evaluating analytic gradients on quantum hardware. Physical Review A **99**(3), 032331 (2019)
26. Schuld, M., Petruccione, F.: Supervised learning with quantum computers, vol. 17. Springer (2018)
27. Shor, P.W.: Fault-tolerant quantum computation. In: Proceedings of 37th Conference on Foundations of Computer Science. pp. 56–65. IEEE (1996)
28. Sim, S., Johnson, P.D., Aspuru-Guzik, A.: Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. Advanced Quantum Technologies **2**(12), 1900070 (2019)
29. Stokes, J., Izaac, J., Killoran, N., Carleo, G.: Quantum natural gradient. Quantum **4**, 269 (2020)
30. Wang, Y.E., Wei, G.Y., Brooks, D.: Benchmarking tpu, gpu, and cpu platforms for deep learning. arXiv preprint arXiv:1907.10701 (2019)

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08760-8_21