Supervised Learning with Parameterized Quantum Circuits

Philip K. Sørli Niane
Department of Physics, University of Oslo, Norway
Github address: Link here

June 29, 2021

Abstract

In this article supervised learning with parameterized quantum circuits(PQCs) were used to separate samples from the iris dataset[1] and classifying them within two classes. Two different ansatzes were tested, with various amounts of parameters in the PQCs. The optimization process where computed classically based on the output of the PQC. The chosen optimization algorithm was the gradient descent.

The results from using supervised learning with PQCs revealed an accuracy of 0.92 on the validation set, using the least complex ansatz with 26 free parameters, while using the second ansatz which was more entangled and complex with 30 free parameters resulted in an accuracy of 0.88 on the same set. The first ansatz had a loss of 0.37, while the second circuit had a loss of 0.27 which shows that the second ansatz were more certain in its true predictions and less certain in its wrong predictions compared to the first ansatz.

Contents

1 Introduction **3**

2 Theory **3**
2.1 Basics of Quantum computing 3
2.2 Basis 4
2.3 Quantum circuits 4
2.3.1 Quantum Logical Gates [4] 5
2.4 Parameterized Quantum Circuits 7
2.4.1 Encoders 7
2.4.2 Ansatzes 8
2.4.3 Entanglement of the qubits 9
2.5 Optimizing the Variational Parameters 9
2.5.1 Gradient Descent 9
2.5.2 Binary Cross Entropy 10
2.5.3 Parameter Shift-Rule 10
2.6 The Iris Dataset 11

3 Method **11**
3.1 The Program Flow 11
3.2 Implementation of the quantum circuit 11
3.3 Parallelization of the Code 12

4 Results **12**
4.1 Initialization of the Variational Parameters 12
4.2 Investigating the Learning Rate and Number of Variational Parameters 14
4.3 Results from Predicting the Iris Dataset 16

5 Discussion **17**
5.1 Initialisation of the variational parameters 17
5.2 Optimal number of variational parameters and learning rate . . . 18
5.3 Predictions on the Iris Dataset 18

6 Conclusion **19**
6.1 Prospects for the future 19

A Appendix **21**
A.1 Source code 21
A.2 Differentiation of the Binary Cross Entropy function 21

1 Introduction

Quantum computing is getting an increasing recognition all over the world, due to the fact that quantum computers have the possibility of changing the world of mathematical computations within almost all fields of science as we know. Due to the increasing acknowledgment of quantum computers, it is only natural to try to combine the field with one of the most popular fields within classical computing, machine learning.

Combining classical- and quantum computing making a hybrid machine learning model sounds like a good idea at first thinking one could achieve the best of both worlds, but can this hybrid approach challenge the spectacular advancements in machine learning that have been constructed with just classical computing?

In this article parameterized quantum circuits containing arbitrary amounts of variational parameters will be used to classify the iris dataset[1]. Two different ansatzes will be investigated as the building blocks of the circuits, and the well known optimization method gradient descent will be used to optimize the variational parameters.

The article is built up by starting off with some basics of quantum computing including having a look at some quantum gates and explanation of quantum circuits. The article then follows by having a look at parameterized quantum circuits, and how these works. In addition a couple of ansatzes will be proposed. The way to optimize the parameters of the PQCs will then be explained. The datasets will be discussed which is later followed by the details of the implementation. Then comes the analysis part unveiling the results and discussion of them, naturally followed by a conclusion summing up the article.

2 Theory

2.1 Basics of Quantum computing

Quantum computing uses the occurrences of quantum mechanics to process and manipulate information. Quantum computing has endless of possibilities to speed up computations when solving certain problems. But to understand the beauty of machine learning within quantum computing, one needs to know the basics of quantum computing first, here comes a short summary with the most important key elements to grasp the article. For a deeper dive into the theory on quantum computing, it would be worth starting here [2].

The key element when comparing classical computers with quantum computers is the comparison between classical bits and qubits. A classical computer uses bits to handle and manipulate information, where each bit can be either 0 or 1. A quantum computer on the other hand uses qubits. Qubits have the possibility of taking the form of superpositions between the original bits of 0 and 1. A qubit

can be a superposition of two states, $\alpha|0\rangle + \beta|1\rangle$ where $|\alpha|^2 + |\beta|^2 = 1$ which opens up the possibility of having multiple states at once.

2.2 Basis

All operations on qubits can be expressed mathematically. Therefore defining a basis of two possible qubit states as orthogonal vectors are only natural. The two states a qubit can have when being observed are $|0\rangle = [1, 0]^T$ and $|1\rangle = [0, 1]^T$. Where $\langle 1|0\rangle = 0$ and $\langle 0|0\rangle = 1$ due to basis states being orthogonal. The basis states expands into $|00\rangle = [1, 0, 0, 0]^T$, $|01\rangle = [0, 1, 0, 0]^T$, $|10\rangle = [0, 0, 1, 0]^T$ and $|11\rangle = [0, 0, 0, 1]^T$ when having two qubits, which naturally increases the vector size the same pattern when adding more qubits. Linear combinations of the different states are also possible, making superpositions as follows:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{1}$$

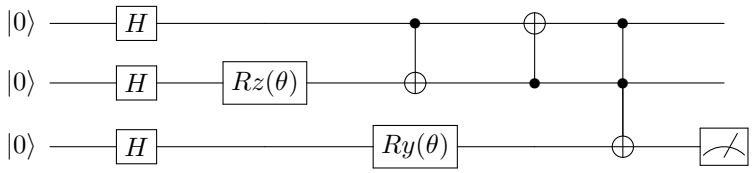
Where $|\alpha|^2 + |\beta|^2 = 1$.

Multiple qubits can be expressed as tensor products of the qubits, the following way:

$$|\psi_1\rangle \otimes |\psi_2\rangle ... \otimes |\psi_n\rangle = |\psi_1\psi_2...\psi_n\rangle \tag{2}$$

2.3 Quantum circuits

Quantum circuits consists of quantum wires transferring information. The information is then manipulated by letting some quantum logical gate also called a quantum gate, act on the current state. A quantum circuit could for instance look like the following example:



Each horizontal line is called a quantum wire, and each of these represents time evolving from left to right. Each quantum wire represents a qubit which can be altered and manipulated. The start point $|0\rangle$ represents the initial state of the qubit, while the end of the qubit represents the quantum state after the quantum gates have acted on the quantum states. When going from left to right, the quantum states encounters some quantum gates represented by boxes containing a letter or just some circles and dots, which unveils the type of gate it is. These quantum gates are the operators which alter the quantum states. Vertical lines between quantum wires shows the presence of controlled gates between the two connected wires. Some quantum gates will be discussed in further details in the next subsection. The last part of the third qubit in the

drawn circuit, reveals that the measurement of a qubit will happen here. The results from the measurement then reveals which state the last qubit is in. By measuring the same qubit multiple times gives the probability of having the state in either state $|0\rangle$ or $|1\rangle$ [3].

2.3.1 Quantum Logical Gates [4]

Quantum logical gates or quantum gates which they also are called are different ways of manipulating the information in quantum circuits. Quantum gates are expressed mathematically by matrices. There are infinite many matrices, which makes it possible to manipulate quantum states infinite different ways in theory. Some quantum states are more popular than others, lets have a look at some of the most frequently used ones.

The **Pauli gates** are often used within quantum circuits. The Pauli X-gate or NOT gate which it is also named can be thought of as a rotation around the x axis π radians. The NOT gate are represented by the following matrix and bracket:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = |0\rangle\langle 1| + |1\rangle\langle 0| \tag{3}$$

Which interchanges the labels between the two qubit states the following way:

$$X|0\rangle = |1\rangle \tag{4}$$

The Pauli Y-gate can be thought of as a rotation around the y axis π radians. The Y-gate are represented by the following matrix and bracket:

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} = -i|0\rangle\langle 1| + i|1\rangle\langle 0| \tag{5}$$

Which maps $|0\rangle$ to $i|1\rangle$ and $|1\rangle$ to $-i|0\rangle$. the following way:

$$Y|0\rangle = i|1\rangle \tag{6}$$
$$Y|1\rangle = -i|0\rangle \tag{7}$$

The Pauli Z-gate works the same way as the other Pauli gates. The Z-gate are represented as a rotation around the z axis π radians. The Z-gate are represented by the following matrix and bracket:

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = |0\rangle\langle 0| - |1\rangle\langle 1| \tag{8}$$

Which maps $|1\rangle$ to $-|1\rangle$ and works as an identity operator on $|0\rangle$ states the following way:

$$Z|0\rangle = |0\rangle \tag{9}$$
$$Z|1\rangle = -|1\rangle \tag{10}$$

But since there exists operators that rotates the quantum states around certain axis by π radians, there must also exist rotational operators that rotate the qubits by an arbitrary amount of radians. In fact the **rotational operator gates** does this exactly. By inserting a parameter θ one can rotate the quantum states around the x, y, or z axis by that many radians. The rotational operators are given by the following matrices:

$$R_x(\theta) = \exp(-iX\theta/2) = \begin{bmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{bmatrix} \tag{11}$$

$$R_y(\theta) = \exp(-iY\theta/2) = \begin{bmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{bmatrix} \tag{12}$$

$$R_z(\theta) = \exp(-iZ\theta/2) = \begin{bmatrix} \exp(-i\theta/2) & 0 \\ 0 & \exp(i\theta/2) \end{bmatrix} \tag{13}$$

Where the subscript indicates which axis the rotation is happening along.

The **Hadamard gate** also called a H-gate are one of the most common quantum gates used. The H-gate are represented by the following matrix and bracket:

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}}\langle 0| + \frac{|0\rangle - |1\rangle}{\sqrt{2}}\langle 1| \tag{14}$$

Which maps $|0\rangle$ to $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|1\rangle$ to $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$. The following way:

$$H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \tag{15}$$

$$H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} \tag{16}$$

Controlled quantum gates are quantum gates that acts on two or more qubits when the criteria of another quantum state are satisfied. The quantum state that needs to satisfy the condition is the controlled gate. An example of a popular controlled gate are the well known CNOT which is a controlled NOT gate. The CNOT on two qubits controls if the first qubit is in state $|1\rangle$. If the controlled qubit possess the state, the NOT gate will act on the second qubit mapping $|0\rangle$ to $|1\rangle$ and opposite. If the controlled qubit does not contain the criteria state, the CNOT will not act on the state, and instead act as an identity operator. Controlled gates are expressed in quantum circuit diagrams as vertical lines between two or more qubits as in figure 1, where the dark dots are the control qubits and the other are the gate that potentially acts on the state.

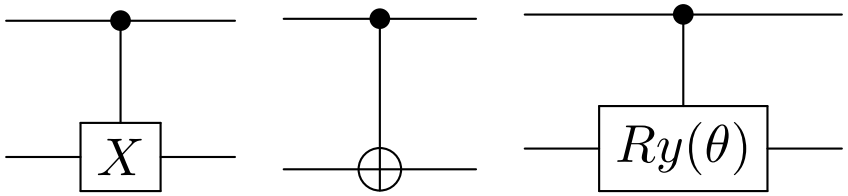


Figure 1: Some controlled quantum gates. a) A controlled NOT gate, called a CNOT. b) Also a CNOT but a slightly different visual appearence but does the same as a controlled NOT gate. c) A controlled Ry gate.

2.4 Parameterized Quantum Circuits

Parameterized quantum circuits also called PQCs are quantum circuits consisting of one or more quantum gates that are dependent of some variational parameter. Most often multiple quantum gates that depends of some parameter(s) is put together into a circuit which is used as an ansatz. The ansatzes used in this article will be revealed in section 2.4.2.

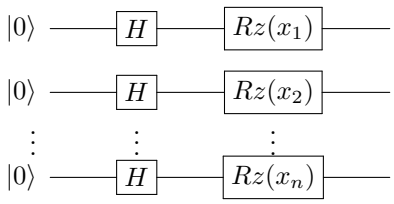
So how is PQCs connected to supervised learning? Shortly summarised, supervised learning aims to learn trends from data that can be generalized to predict other unseen data. In search of the perfect generalisation of data, the machine learning models need to adjust to training data by optimizing the parameters decreasing the loss which is computed from the prediction estimates and the true labels.

Using this same approach by having the PQC which is dependent on some parameters, predict an output and optimize the parameters used in the circuit gives a machine learning approach to PQCs. To read more about machine learning and PQCs feel free to have a look here [5].

2.4.1 Encoders

Encoding the information that is going to be manipulated in quantum circuits is the first part of the quantum circuits and therefore crucial. There are multiple ways of encoding the data e.g. basis encoding, amplitude encoding, qsample encoding etc. Different encoding methods can be studied in [6].

A way of encoding the data which will also be done in this article, is by having every qubit start off with a Hadamard gate splitting the quantum state into a superposition which is then acted on by a $Rz(x)$ gate. The clue on encoding classical data into quantum states lays here, because the input x will be the classical feature data. The encoding part of the ansatzes used be as follows:

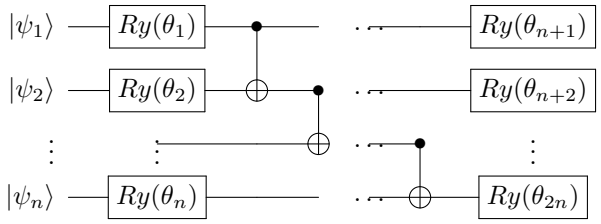


Where n is the number of qubits. By rewriting each feature into a $Rz(x)$ quantum gate which is dealt its own qubit, the needed qubits is therefore equal to the number of features, excluding possible qubits used for measuring.

2.4.2 Ansatzes

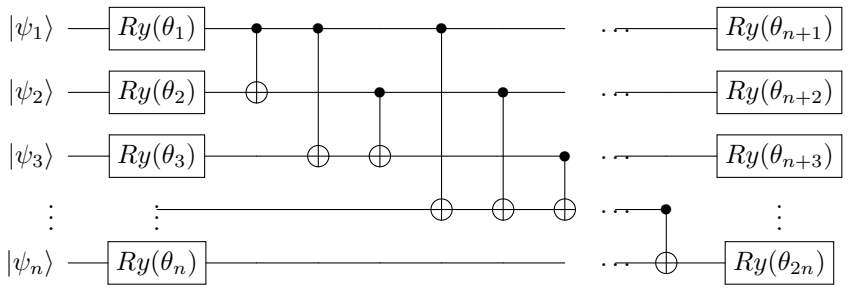
Ansatzes used in quantum computations are parts of quantum circuits which is assumed to mimic the behaviour of some function. Making the ansatz more complex opens up the possibility of reaching more parts of the Hilbert space where all quantum states resides. Making really large and complex quantum circuits is difficult due to quantum states being quiet unstable over time, which is why ansatzes must not be made too complex. In this article, two ansatzes will evaluated.

The first ansatz will be referred to as "ansatz 1" throughout the article, and consists of rotational operators $Ry(\theta)$ along each qubit, which is then followed by some entangling between the qubits using controlled not gates between every pairwise qubit as follows.



Where $\boldsymbol{\theta}$ is the variational parameters that needs to be optimized and n is the number of qubits. When more variational parameters are used, the ansatz adds more $Ry(\theta)$ gates following the same pattern.

The second ansatz will be referred to as "ansatz 2" throughout the article, and consists of the rotational operator $Ry(\theta)$ along each qubit as in ansatz 1, but the entanglement in ansatz 2 is a bit more complicated. The $Ry(\theta)$ operators are followed by controlled gates controlling all qubits above them as follows:



Where $\boldsymbol{\theta}$ is the variational parameters that needs to be optimized and n is the number of qubits. When more variational parameters are used, the ansatz adds more $Ry(\theta)$ gates following the same pattern. Notice that the input states of the circuits are $|\psi_x\rangle$ since the input of the ansatz part of the circuit is the already altered quantum states from the encoder.

2.4.3 Entanglement of the qubits

The last part of the PQCs is the entangling part which binds all qubits together. For ansatz 1 the entangling part is a controlled not gate controlling all other qubits except the target qubit which is the last qubit. A measurement is then performed on the last qubit, giving the probability $|c_1|^2$ of having the qubit in state $|1\rangle, which can be used as a classification prediction between class |0\rangle and class |1\rangle$.

For ansatz 2, the entanglement is just a continuation of the controlled not gates from the same ansatz. Which means that the middle part from the ansatz is composed to the end of the quantum circuit. A measurement is then performed on the last qubit, giving the probability $|c_1|^2$ of having the qubit in state $|1\rangle$, which the same way as for ansatz 1 can be used as a classification mesurement.

2.5 Optimizing the Variational Parameters

So how is it possible to optimize the variational parameters in PQCs? As mentioned in section 2.4 optimizing the parameters is done by minimizing the loss. The loss is computed by a so called loss function. Then by using an optimization method it is possible to change the variational parameters in a way that minimizes the loss.

2.5.1 Gradient Descent

A clever way of finding the best variational parameters $\boldsymbol{\theta}$ where $\boldsymbol{\theta}$ is a vector containing the different parameters, is by using the gradient descent method. The gradient descent method is a method used for minimizing a function, by moving in the opposite direction of the gradient.

The gradient descent method is quiet straight forward. An initial $\boldsymbol{\theta}$, also known as a guess, is needed to start of the method and compute the next $\boldsymbol{\theta}$. The gradient descent formula is the following [7]:

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n - \gamma \frac{\partial L}{d\boldsymbol{\theta}} \tag{17}$$

Where γ is the learning rate and L is the loss function.

2.5.2 Binary Cross Entropy

To optimize the parameters, there is a need for a function which shows how good or bad a prediction is. This function is called the loss function. There are several possible loss functions to use when determining the loss, but in this article binary cross entropy will be used as an estimate of the loss. The binary cross entropy function often called log loss, goes as follows:

$$L = -\frac{1}{N}\sum_{i=1}^{N} y_i \log f(x_i;\boldsymbol{\theta}) + (1-y_i)\log\left(1 - f(x_i;\boldsymbol{\theta})\right) \tag{18}$$

Where N is the number of samples computing the loss of, y_i is the true label of the i'th sample, and $f(x_i;\boldsymbol{\theta})$ is the prediction of the i'th sample. To see more on binary cross entropy, please see [8].

To optimize the loss function, the gradient of the loss needs to be found, the derivative of binary cross entropy with respect to some variational parameter $\boldsymbol{\theta}$ is written as follows, the derivation can be seen in the appendix:

$$\frac{\partial}{\partial\theta_k} L = \sum_{i=1}^{n} \frac{f(x_i;\boldsymbol{\theta}) - y_i}{f(x_i;\boldsymbol{\theta})(1 - f(x_i;\boldsymbol{\theta}))} \frac{\partial}{\partial\boldsymbol{\theta}_k} f(x_i;\boldsymbol{\theta}) \tag{19}$$

Where the values are the same as in equation (18), and θ_k is the k'th variational parameter. When using PQCs the gradient is not as easy accessed, so $\frac{\partial}{\partial\boldsymbol{\theta}_k} f(x_i;\boldsymbol{\theta})$ can be computed by using the shift rule which is explained in the next section.

2.5.3 Parameter Shift-Rule

The parameter shift rule is a way of extracting the gradient of quantum circuits when using quantum gates that contains varational parameters. The parameter shift rule is pretty straight forward. The gradient can be estimated by evaluating the circuit with the parameter that is differentiated with respect to, shifted $\frac{\pi}{2}$ to the right, then evaluating the same circuit again, but this time the same parameter is shifted to the left. Then the gradient can be estimated by having the first prediction subtracted by the second prediction and divided in half. The

formula goes as follows:

$$\begin{aligned}\frac{\partial f(x_i;\theta_1,\theta_2,\ldots,\theta_k)}{\partial\theta_j} &= \frac{f(x_i;\theta_1,\theta_2,\ldots,\theta_j+\pi/2,\ldots,\theta_k)}{2} \\ &- \frac{f(x_i;\theta_1,\theta_2,\ldots,\theta_j-\pi/2,\ldots,\theta_k)}{2}\end{aligned} \tag{20}$$

To read about the parameter shift-rule in a more detailed manner, please have a look here [9].

2.6 The Iris Dataset

The iris dataset is a well known datset containing different types of information for the plant iris. The different types of irises is Iris Setosa, Versicolour and Iris Virginica, but in this article only the former two will be used. The different information about each iris contained in the dataset is sepal length, sepal width, petal length and petal width. There is 50 samples withing each class.

3 Method

3.1 The Program Flow

The flow of the program on supervised learning with quanutm computers goes as follows. The implementation starts of by determining the desired parameters, like number of quantum gates and the wanted ansatz. Number of epochs and minibatches is also defined, but due to the dataset being small, a batchsize of 1 is used throughout the project, which means that the parameters will update after each sample. Further on, the dataset is defined, which is scaled and split into a train and a validation set of 75% and 25% respectively. The validation set will serve as a test set due to the small dataset available.

After all parameters and data are prepared and organized, it is time to start the training. The samples in addition to the variational parameters are fed into the quantum circuit computing an output between 0 and 1. The output is then brought further into the gradient descent method along with the sample and its true label. Gradient descent is then performed on each of the variational parameters trying to make the prediction even more accurate. The optimized parameters is then returned and used on the next sample. This goes on until each of the samples have been through the process, this equals to 1 epoch. This process is then repeated n more times, where n is the number of epochs, on search for the optimal parameters.

3.2 Implementation of the quantum circuit

Since the project implements quantum circuits, it is worth mentioning the convenient library qiskit. Qiskit is a library that is used to simulate quantum

computations on a local classical computer, but also has the ability to run that same code at a real time quantum computer prototype.

In this project, qiskit was used to simulate a noisy quantum circuit simulator, which took the input of a sample, where each feature was processed by a qubit. The information was then brought further into the quantum circuit which was altered by the parameterized quantum gates, the qubits were also entangled, to make the qubits affect eachother. When the information reached the end of the quantum circuit, the last qubit was measured which served as the prediction.

3.3 Parallelization of the Code

To save some time when searching for the optimal parameters, the code were parallelized using the fork() command. Fork() duplicates the code of the main process, also called a parent process, unto another core which then the process is called a child process. The parent process can fork multiple child processes unto the other cores having them all run in parallel. Changing the parameters after each fork, opens up the possibility of doing multiple simulations at the same time which ensures that the results can be accumulated even faster, when investigating the machine learning model with different initial values and parameters.

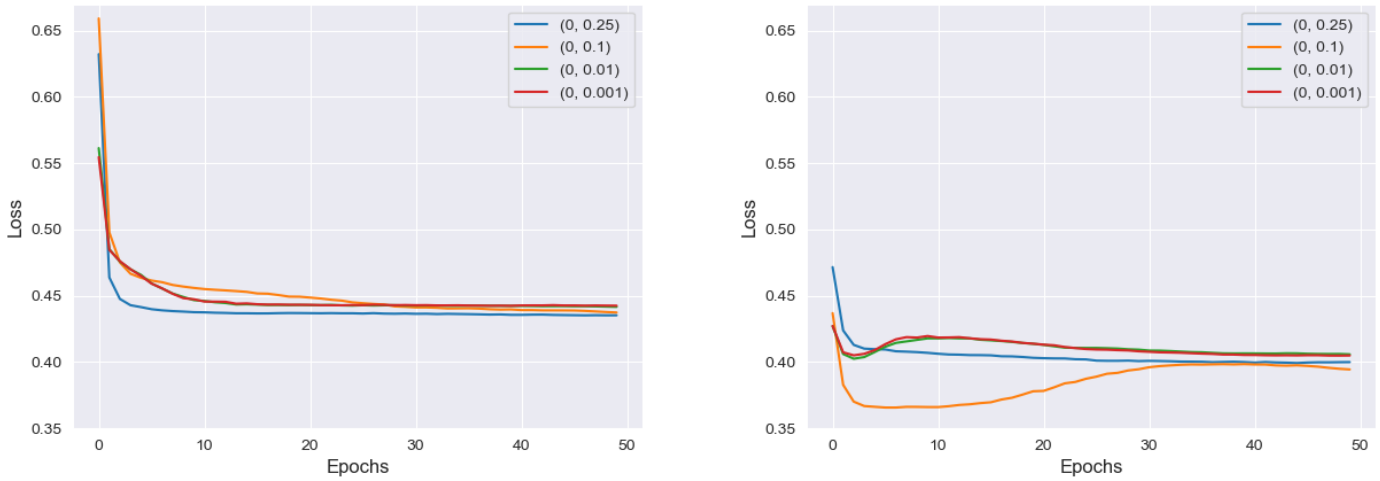
4 Results

Then it is time to have a look at the results, of the simulations.

4.1 Initialization of the Variational Parameters

There is lots of different- ways and configurations to initialize the variational parameters before starting the program. Due to the algorithm and the gradient descent naturally needing some initial guesses for the parameters, this was investigated.

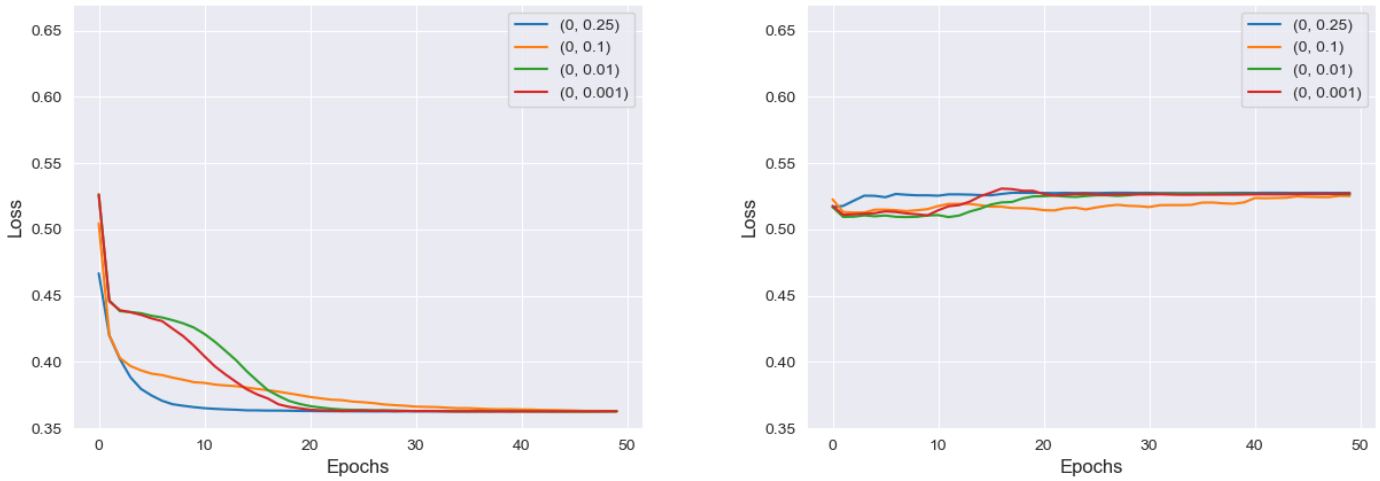
The program were ran with different ways of initializing the variational parameters. The results from initializing the parameters by using a uniform distribution can be seen in figure 2.



(a) (b)

Figure 2: Loss plotted as a function of epochs with 10 variational parameters and learning rate of 0.01 using ansatz 1. The variational parameters has a uniform initialization with random values from the interval which is described in the figures. a) Training data b) Validation data

The results from initializing the parameters by using a normal distribution can be seen in figure 3.



(a) (b)

Figure 3: Loss plotted as a function of epochs with 10 variational parameters and learning rate of 0.01 using ansatz 1. The variational parameters has a normal initialization with random values from the interval which is described in the figures. a) Training data b) Validation data

The chosen distribution for the rest of the computations were chosen to be a uniform distribution from the interval of $(0, 0.1)$ due to reasons explained in section 5.1

4.2 Investigating the Learning Rate and Number of Variational Parameters

The machine learning model was trained with different amounts of variational parameters in combination of different learning rates. The results from using ansatz 1 can be seen in figure 4

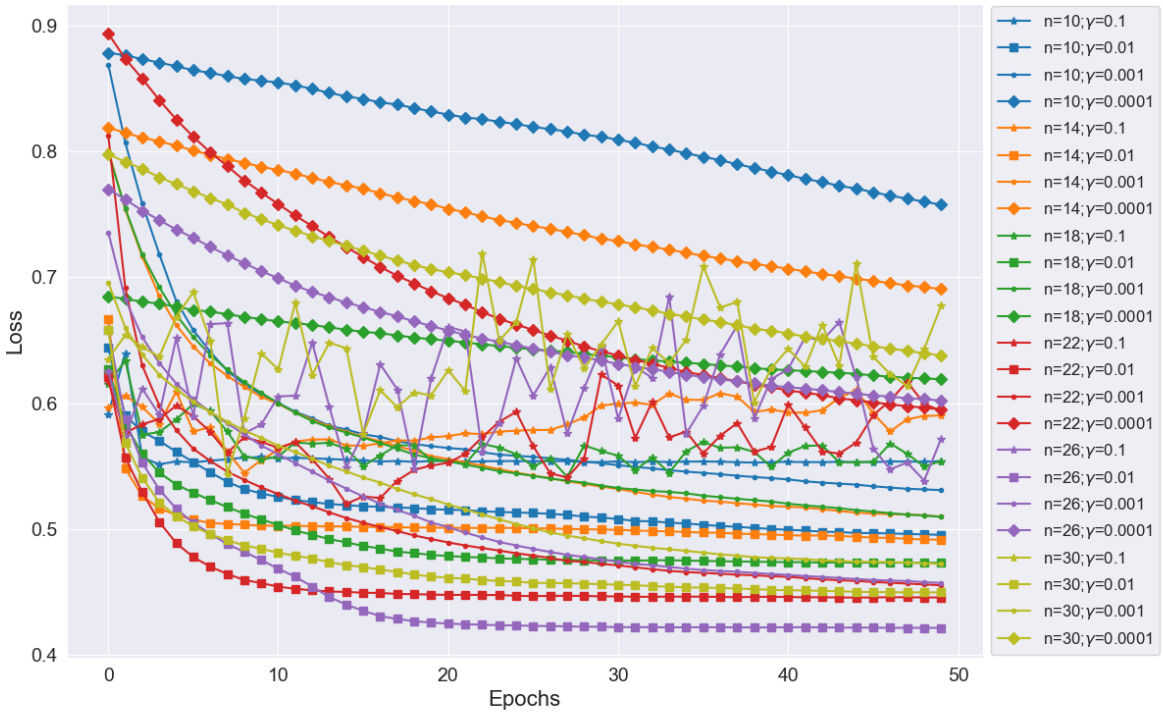


Figure 4: Loss as a function of epoch of the training data, with n number of parameters and learning rate γ. Ansatz 1 was used with uniform distribution of the initial parameters from the interval of $(0, 0.1)$

The same results from using ansatz 2 can be seen in figure 5.

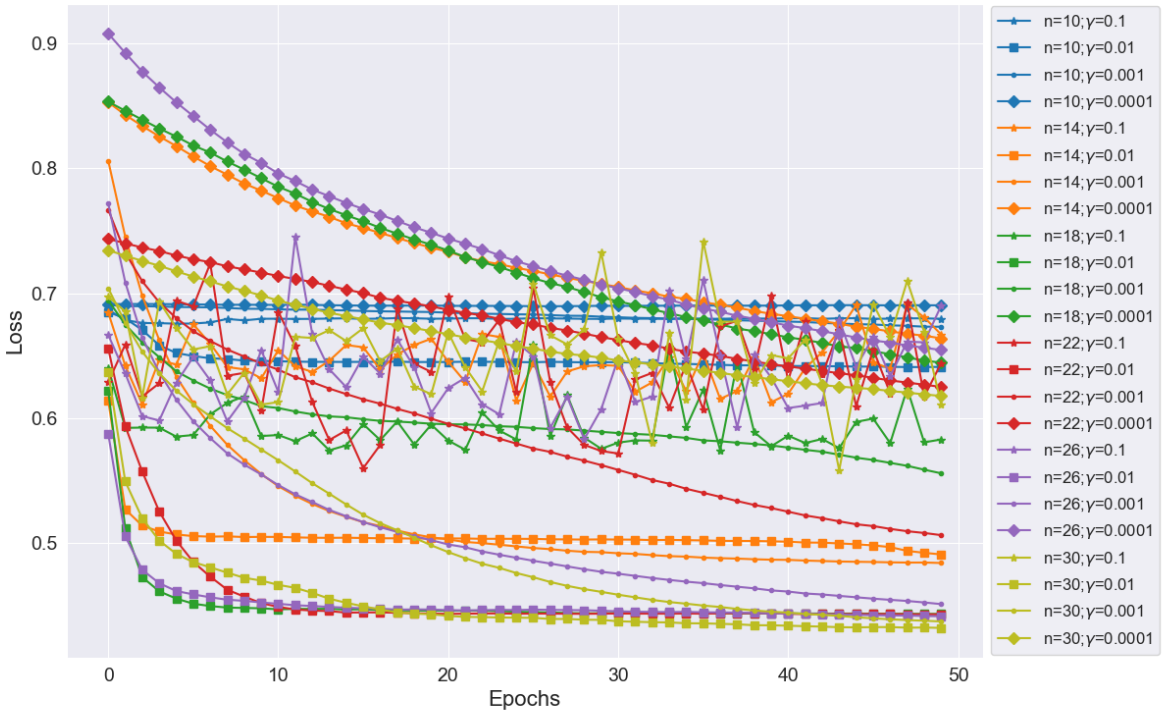


Figure 5: Loss as a function of epoch of the training data, with n number of parameters and learning rate γ. Ansatz 2 was used with uniform distribution of the initial parameters from the interval of $(0, 0.1)$

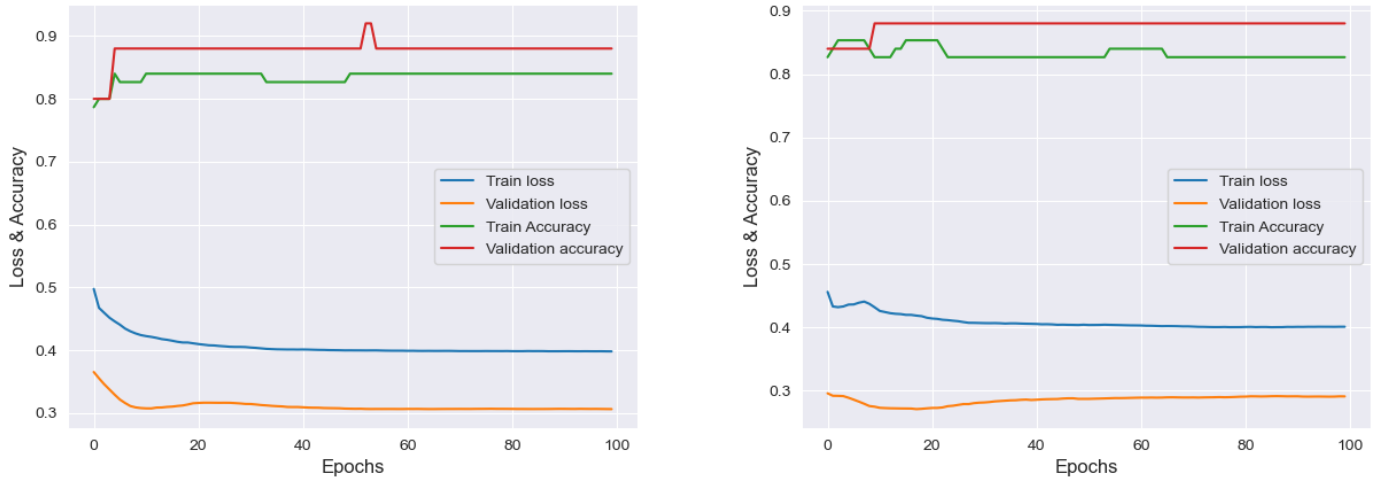
The optimal parameters chosen for both ansatzes can be seen in table 1. The reason for the choices of parameters can be seen in section 5.2.

	Number of parameters	Learning rate	Distribution type	Distribution interval
Ansatz 1	26	0.01	Uniform	(0, 0.01)
Ansatz 2	30	0.01	Uniform	(0, 0.01)

Table 1: The optimal parameters chosen for each ansatz

4.3 Results from Predicting the Iris Dataset

The results from using PQCs to predict the iris dataset can be seen in figure 6



(a) (b)

Figure 6: Loss and accuracy as a function of epoch for both the training- and validation data when applying the PQCs with the optimal parameters from table 1. a) Ansatz 1. b) Ansatz 2

The results from classifying the iris dataset can be seen in table 2.

	Val. acc.	Epoch-acc.	Val. loss	Epoch-Loss	Acc. from classical ML [10]
Ansatz 1	0.92	[52,53]	0.37	99	1
Ansatz 2	0.88	[9, 99]	0.27	17	1

Table 2: Results from classifying the iris dataset. Val. is the validation set, acc. is the accuracy, Epoch-acc is the corresponding epoch with the highest accuracy achieved, and the same for epoch-loss.

5 Discussion

5.1 Initialisation of the variational parameters

By having a look at figure 2 it is quiet clear that all uniform initializations converges nicely towards some loss value. The two initialization values that converges towards the lowest training loss is the initialization interval of $(0, 0.25)$ and $(0, 0.1)$. Then having a look at the validation loss in the other sub figure reveals that the initialisation of $(0, 0.1)$ gives the lowest validation loss, therefore the best choice of uniform distribution of initial parameters is from the interval

of $(0, 0.1)$. It is worth mentioning that the reason the validation set has a lower loss on the validation set than the training set is probably due to the validation set being too small consisting of only 25 samples.

Having a look at figure 3, it is shown that the training loss is about 0.1 lower after convergence compared to the uniform distribution. The validation loss in the normal distribution case is converging towards approximately the same loss. Because of fast convergence rate for the training loss, the optimal normal distribution seems to be from the interval of $(0, 0.1)$ with the center in the middle. Comparing the the optimal uniform distribution to the optimal normal distribution, makes the choice of distribution end up as an uniform distribution in $(0, 0.1)$ becuase it has a lower validation loss.

5.2 Optimal number of variational parameters and learning rate

Determining the parameters when using ansatz 1, figure 4 shows that the quantum circuit giving the lowest training loss was the circuit consisting of 26 parameters, and a learning rate of 0.01. The convergence rate was also quiet fast having a stable loss from around 20 epochs and so on.

Figure 5 on the other hand, showing the results from using ansatz 2, shows that the quantum circuit giving the lowest training loss was the circuit consisting of 30 parameters and a learning rate of 0.01 with quiet steady convergence.

For both ansatzes the cirquits used with learning rates of 0.1 was a bit too high, making the loss quiet unstable, which makes sense due to the parameters changing quiet a lot each step taken with the gradient descent method. A too small learning rate of 0.0001 on the other hand had a quiet slow converging rate, 50 epochs not being enough.

It is easy to see from table 1, that all but the number of variational parameters are the same for the two ansatzes. The reason for this is probably due to ansatz 2 being a bit more complicated with controlled not gates entangling the qubits to all qubits above the targeted one. This also makes sense when comparing the loss of circuits containing 10 parameters. The 10-parameter circuits using ansatz 1, managed to learn some trends and reduce its losses some, while the 10-parameter circuits using ansatz 2 had a hard time learning trends with the training loss being approximately constant.

5.3 Predictions on the Iris Dataset

Table 2 and figure 6 shows that the best accuracy when using ansatz 1 on the validation set was achieved at epoch 52 and 53, which have an accuracy of 0.92. When using ansatz 2 the best accuracy with a score of 0.88 was achieved between epoch 9 and 99, which means that the convergence towards a steady accuracy was quiet fast. Ansatz 1 has a bit better accuracy than ansatz 2, but a higher minimum loss. This indicates that ansatz 2 were more certain of its true

predictions and less certain of its false prediction compared to ansatz 1. Both ansatzes were getting somewhat close to the state of the art classical machine learning approaches which has an accuracy of 1, missing 8 and 12 % for ansatz-1 and 2 respectively to achieve classical machine learning results.

The accuracy is really steady for both ansatzes after a couple of epochs, this might be due too gradient descent having a bit of a hard time optimizing the parameters when the gradient becomes small, which is only natural remembering that there are more than 20 parameters in the quantum circuits to be optimised each batch. This would probably be easier for an optimization method that contains some momentum.

It is worth mentioning that the validation set has a lower loss and higher accuracy for both ansatzes which has a tendency to happen when the validation set is small. It is also worth mentioning that the accuracy starts quiet high in both cases, which might be due to the initialisation of the parameters being quiet good, in addition to the fact that small datasets only need a bit of luck to achieve high accuracies.

6 Conclusion

After exploring hybrid versions of machine learning methods, connecting classical computations and PQCs shows that there is a lot of potential within the machine learning world of quantum computing. The results did reach classical machine learning state of the art approaches but showed satisfactory results, and might also reach the classification levels of classical machine learning with some finetuning.

In this article the PQCs were used to classify the iris dataset using the optimal parameters found after going for a search for the best learning rate, number of parameters and distributions possible.

6.1 Prospects for the future

The main prospects for the future would quiet naturally be to find some more optimal ansatzes. Personally I think that spending some time finding a better suited ansatz would give a great boost of the performance, probably pushing the accuracy of the iris dataset further into the 90%'s.

Testing the machine learning models on larger and more advanced datasets would also be quiet rewarding, seeing how the model performs.

Another interesting prospect for the future would be to have a look at how large depth a quantum circuit can manage and how many variational parameters can be used before the cirquit becomes too unstable. Having a more clear limit on how deep a circuit can be, would probably make the search for optimal ansatzes more progressive.

References

[1] D. Dua and C. Graff, *UCI machine learning repository*, 2017. [Online]. Available: http://archive.ics.uci.edu/ml.

[2] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, 10th. USA: Cambridge University Press, 2011, ISBN: 1107002176.

[3] D. Marciano, *Quantum computing for everyone - part iii: Quantum circuits and openqasm, code project*, [Online; accessed 30-March-2021], 2018. [Online]. Available: %5Curl%7Bhttps://www.codeproject.com/Articles/1182208/Quantum-Computing-for-Everyone-Part-III-Quantum-Ci%7D.

[4] A. Asfaw, L. Bello, Y. Ben-Haim, M. Bozzo-Rey, S. Bravyi, N. Bronn, L. Capelluto, A. C. Vazquez, J. Ceroni, R. Chen, A. Frisch, J. Gambetta, S. Garion, L. Gil, S. D. L. P. Gonzalez, F. Harkins, T. Imamichi, H. Kang, A. h. Karamlou, R. Loredo, D. McKay, A. Mezzacapo, Z. Minev, R. Movassagh, G. Nannicni, P. Nation, A. Phan, M. Pistoia, A. Rattew, J. Schaefer, J. Shabani, J. Smolin, J. Stenger, K. Temme, M. Tod, S. Wood, and J. Wootton. (2020). "Learn quantum computation using qiskit," [Online]. Available: http://community.qiskit.org/textbook.

[5] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, "Parameterized quantum circuits as machine learning models," *Quantum Science and Technology*, vol. 4, no. 4, p. 043 001, Nov. 2019. DOI: 10.1088/2058-9565/ab4eb5. [Online]. Available: https://doi.org/10.1088/2058-9565/ab4eb5.

[6] M. Schuld and F. Petruccione, *Supervised Learning with Quantum Computers*, 1st. Springer Publishing Company, Incorporated, 2018, ISBN: 3319964232.

[7] Wikipedia contributors, *Gradient descent — Wikipedia, the free encyclopedia*, https://en.wikipedia.org/w/index.php?title=Gradient_descent&oldid=1008553674, [Online; accessed 28-February-2021], 2021.

[8] D. Godoy, *Understanding binary cross-entropy / log loss: A visual explanation*, https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a, [Online; accessed 28-June-2021], 2018.

[9] G. E. Crooks, *Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition*, 2019. arXiv: 1905.13311 [quant-ph].

[10] P. Rathi, *100% accuracy (classification) in iris dataset*, https://www.kaggle.com/prakharrathi25/100-accuracy-classification-in-iris-dataset, [Online; accessed 29-June-2021], 2019.

A Appendix

A.1 Source code

All the source code is located in
`https://github.com/philipkarim/Supervised-Learning-with-Parameterized-Quantum-Circuits`.

A.2 Differentiation of the Binary Cross Entropy function

The binary cross entropy function is given as the following:

$$L = -\frac{1}{N}\sum_{i=1}^{N} y_i \log f(x_i;\boldsymbol{\theta}) + (1-y_i)\log\left(1-f(x_i;\boldsymbol{\theta})\right) \tag{21}$$

Then by focusing on the one sample having N=1, and set $f = f(x_i;\boldsymbol{\theta}$ and $y = y_i$ gives the following:

$$L = -y\log f - (1-y)\log\left(1-f\right) \tag{22}$$

Then differentiating the loss with respect to θ_k gives the following:

$$\frac{\partial L}{\partial \theta_k} = \frac{\partial}{\partial \theta_k}(-y\log f - (1-y)\log\left(1-f\right)) \tag{23}$$

Then differentiating the loss with respect to θ_k using the chain rule, and remembering that f is actually $f(x_i;\boldsymbol{\theta})$ gives the following expression that needs to be solved:

$$\frac{\partial L}{\partial \theta_k} = \frac{\partial L}{\partial f}\frac{\partial f}{\partial \theta_k} \tag{24}$$

Computing $\frac{\partial L}{\partial f}$ gives:

$$\frac{\partial L}{\partial f} = -\frac{y}{f} + \frac{1-y}{1-f} \tag{25}$$

Which can be written as

$$\frac{\partial L}{\partial f} = \frac{f-y}{f(1-f)} \tag{26}$$

Throwing this into equation (25) and generalise the fomrula for multiple samples in addition to writing $f(x_i;\boldsymbol{\theta}) = f_i$ gives the derivative of the loss:

$$\frac{\partial}{\partial \theta_k} L = \sum_{i=1}^{n} \frac{f(x_i;\boldsymbol{\theta}) - y_i}{f(x_i;\boldsymbol{\theta})(1-f(x_i;\boldsymbol{\theta}))}\frac{\partial}{\partial \boldsymbol{\theta}_k} f(x_i;\boldsymbol{\theta}) \tag{27}$$

Where $\frac{\partial}{\partial \boldsymbol{\theta}_k} f(x_i;\boldsymbol{\theta})$ is computed using the parameter shift function in equation (20).