1 Results and analysis

1.1 Iris dataset

As stated in section **??**, we will investigate only two of the three classes in the Iris dataset - which allows us to only make measurements on one of the qubits to determine the class of the sample. Using a train-test-split of 80/20, with two thirds of the dataset gives us 80 samples, and subsequently 80 different quantum ciruits that we will evaluate, and take measurements on, to make predictions.

1.2 Training the model circuit

Using the quantum circuit described in section **??**, with 13 learnable parameters (3 sequences, and 1 bias term) we encode our dataset and train the model using a standard gradient descent method (see section **??**), we are ready to evaluate the performance of our model.
Running for four different learning rates, and 35 epochs (with 10000 shots per epoch), resetting the parameters to the same starting point before each run, produces the following graphs:

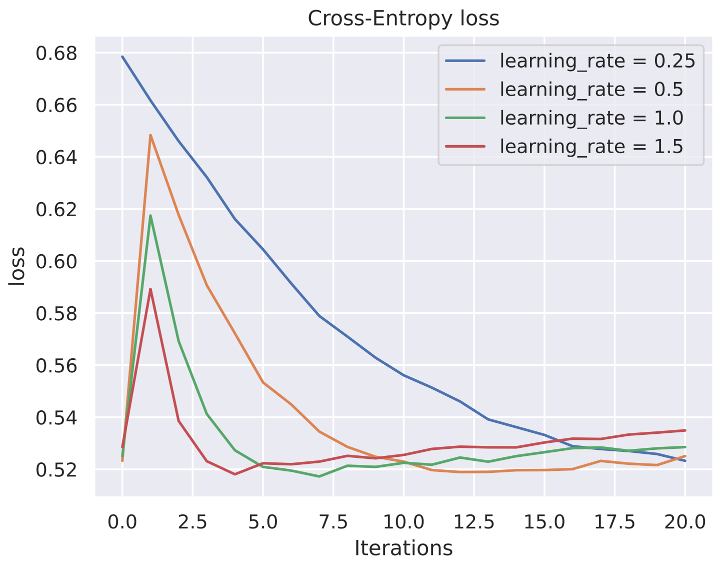

Figure 1: Cross entropy as a function of training iterations for different learning rates

From the graph in figure (**??**) we can see that all 4 choices of learning rates converge to the same minima, $\approx \text{CE} = 0.52$, but at different rates. The higher learning rates converge quicker, which can be intepreted as them taking "larger steps" in parameter space, and thus finding a minima much quicker, which is consistent with the results in the graph.
It seems also that the algorithm diverges slightly for the higher learning rates, which could be caused by the fact that the optimization "jumps" in and out of the local minima that the algorithm converges to when the learning rate is larger. This is a common problem with gradient descent, and is usually solved by decreasing the learning rate as the algorithm converges (using more sophisticated, momentum-based gradient descent methods, like ADAM). One interesting observation is the fact that for $l = 0.25$, it seems that the algorithm starts at a much higher initial loss - and this is most likely due to the randomness of the predictions, as we only ran 10 000 shots per epoch, we cannot guarantee that we predict the true state of the system.
A more rigorous study would employ a much larger number of shots, and also run the optimization multiple times to attempt to minimize the probability that the initial state is mis-read.

A different, and maybe even more informative measure for model performance, is the check the prediction accuracy for the four different circuit setups. In figure (**??**) we again identify the fault that arises from the randomness of the measurements, as the initial accuracy scores are somewhat different, even for the same initial parameters. What we do see, however, is that the larger learning rates again converge

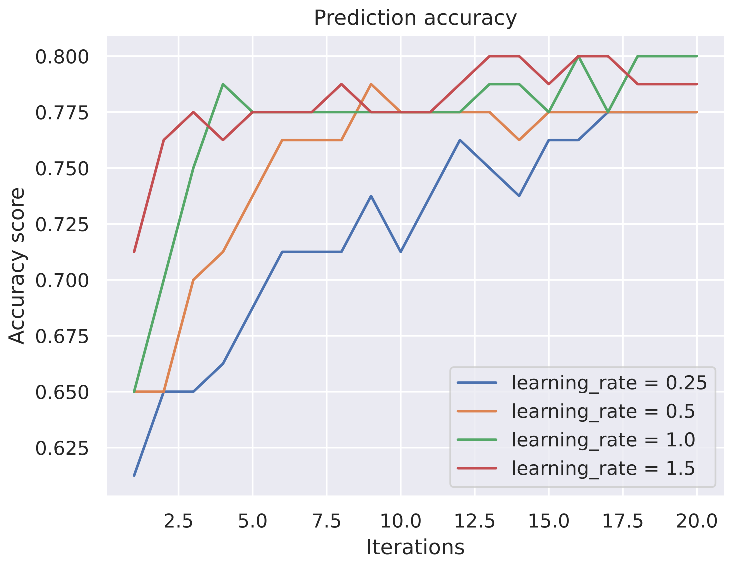

Figure 2: Prediction accuracy as a function of training iterations for different learning rates

faster to a high accuracy score of $\approx 80\%$, while the smallest learning rate $= 0.5$ take more iterations to converge, and to a lower accuracy. Some of the reason for the varying levels of accuracy is due to us having to ”round” the predictions when measuring the accuracy scores. Since the predicted measurements will be averages over 10 000 shots, they will not be integer numbers - maybe 8000 shots are evaluated to 0, still 2000 are perdricted 1. Naturally, there is the possibility to be ”lucky” that your floating point predictions minimize the loss-function, while the rounded predictions yield a ”low” accuracy score. More shots per epoch would yield a more accurate prediction, but also increase the computational cost of the model - which is why we've chosen to use 10 000 shots per epoch.

1.3 Data encoding

Using the encoding described in section **??**, we allow the model to properly entangle differently for each sample - yielding a more diverse starting point for the ansatz to classify. We could've added even more rotation gates, by also looking at the difference between sample points - to make the qubit-states more dependent on all the feature values in a sample - but as we can see from figure (**??**), the entanglement introduced from the CNOT gates, both in the encoding and the ansatz, is sufficient for the model to accurately classify the samples. More complex encoding could potentially yield more rapid convergence in our model circuit, or even improve on the accuracy scores shown in figure (**??**).

To see the importance of a good encoding, we will attempt to run the model circuit with two different encodings:

- A simple encoding, with only one rotation gate per qubit
- A more complex encoding, with a rotation gate per qubit, and CNOT gates to entangle them

Both encoding still make use of a Hadamard gate to create superpositons - and the circuit is ran with a learning rate $= 1.0$, and an ansatz that applies one rotation gate per qubit, and also 3 CNOT gates between the pairs. Running for 10 epochs to visualize the necessity for entanglement yields the results shown in figure (**??**)

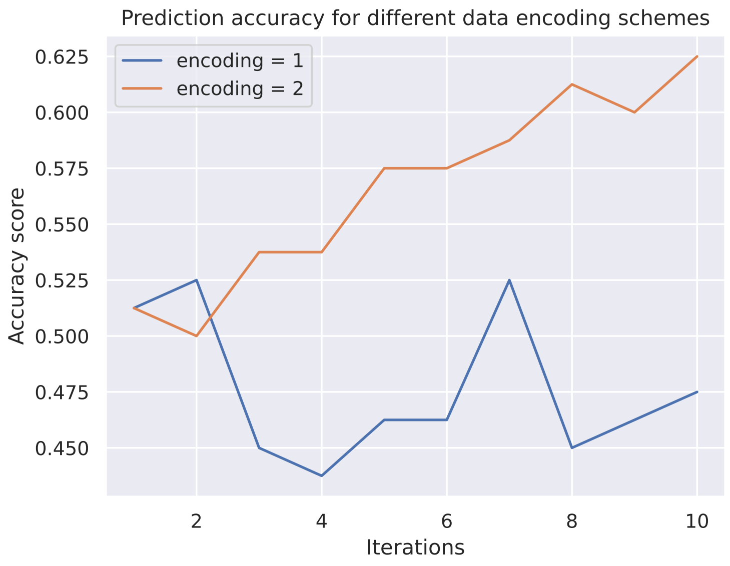

Figure 3: Prediction accuracy as a function of epochs (iterations) for different encodings

where we see that the non-entangled encoding performs much worse than the entangled encoding, which is to be expected. It diverges to a low accuracy score of ≈ 0.5, which is the same as qualified guessing. Checking the loss function, we get the following: This graph gives us a good indication that the

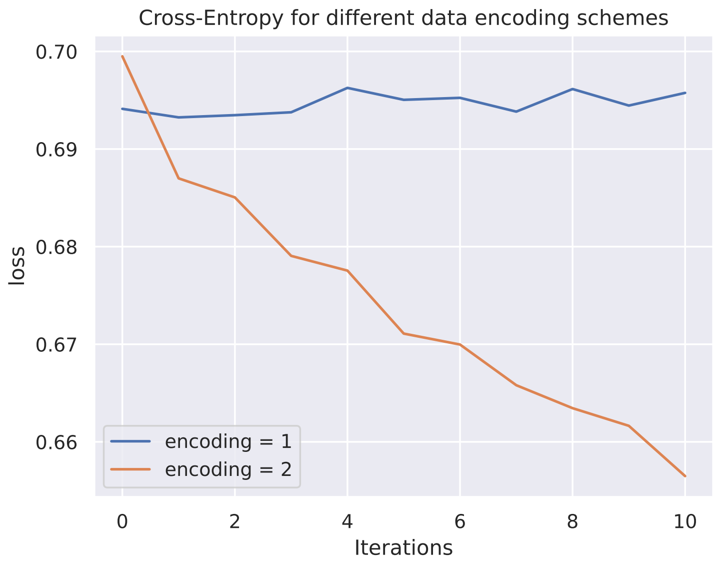

Figure 4: Cross-entropy loss as a function of epochs (iterations) for different encodings

entangled encoding performs at a higher level - and that we need the entanglement to properly classify the samples. Without the entanglement introduced in the encoding, the model can not properly mimic the patterns in the dataset, and it does not converge whatsoever.

1.4 Variational ansatz

Our choice of ansatz, with multiple CNOT gates and rotation gates, allowed our model to capture the underlying structure of the dataset, accurately predicting the training set. Evaluating our model circuit with the optimized parameters found when running the scheme that produced the plots in figure (**??**), yielded the following scores

- Test accuracy: 0.52
- Test loss: 0.79

We could see that our model potentially did overfit the data, as the test accuracy is not on par with the training accuracy - indicating a low level of generalization in our model circuit. The accuracy is no better than guessing, so our model clearly failed at predicting on this unseen test-data. This is something we could expect, given how simple the circuit is, and more complex datasets would require even more complex ansatzes to achieve similar accuracy scores. This is reason for concern if we'd like to apply such a quantum machine learning circuit on real life classification tasks - but as a proof of concept, we can be satisfied with convergence, and a decent accuracy score on the training data. This shows the potential of quantum machine learning, but also highlights the necessity for continued research in the field - to develop more complex models that can generalize better to unseen data.
Given a learning rate of $l = 1.0$, which from figure (**??**) and (**??**) seems to be an optimal choice of hyperparameter, we can test different variants of the variational ansatz to see how the model complexity affets the accuracy scores. This has been done for 3 different variants:

- A simple ansatz, with only one set of rotation gates, and no CNOT gates
- A more complex ansatz, with one set of rotation gates, and some CNOT gates
- A full ansatz, with multiple sequences of rotation gates, and CNOT gates between each pair of qubits, and a bias term.

The results are shown in figure (**??**)

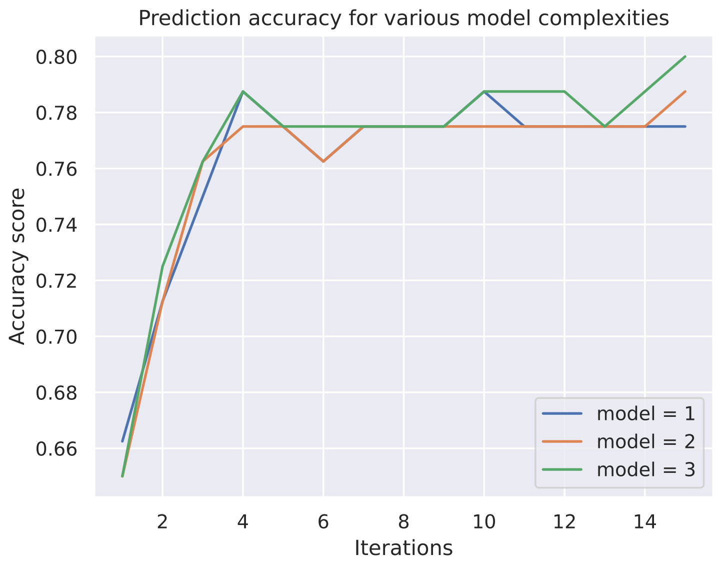

Figure 5: Prediction accuracy as a function of ansatz complexity

Interestingly, all 3 levels of complexity perform at the same level, accuracy wise. This indicates that we may not need a very complex ansatz to capture the patterns in the Iris dataset, and could make do with a very simple model ansatz. Another point this could show, is that our encoding is very efficient, and our good encoding provides a great initial starting point for our optimizatoin algorithm, allowing the model to accurately predict with few learnable parameters. We also check the loss function, to see if the model complexity affects the convergence rate of the model, presented in figure (**??**)

Figure (**??**) further backs our initial discovery that the model complexity does not necessarily need to be very high, and a very simple ansatz can capture the structure in the data. We still do not achieve a higher accuracy than 80%, with the loss function plateuing at $CE \approx 0.5$, which is an indicator that the model lands in a local minima. Another possibility is that the model suffers from "barren plateus", regions in the parameter space where the gradients are vanishingly small (or zero), without the model having reached a minima. The barren plateu problem is a common problem in quantum computing, and

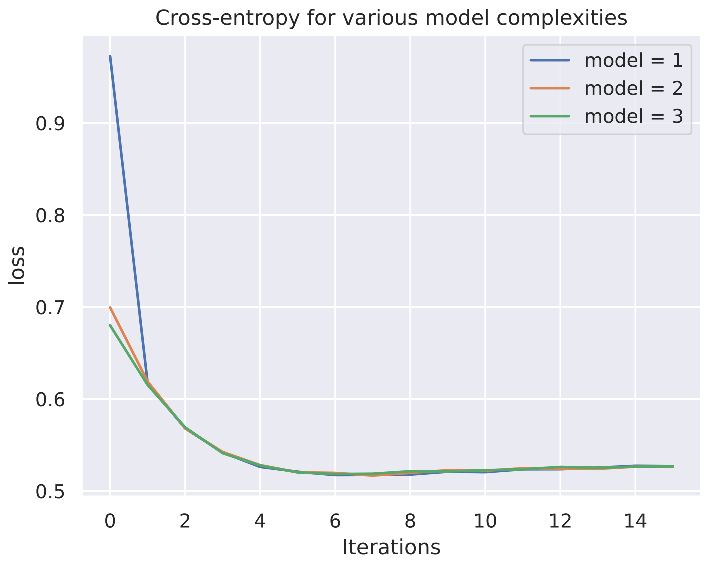

Figure 6: Cross entropy as a function of ansatz complexity

a common solution is to use non-gradient based methods. As a sanity check, we ran one model, with learning rate = 1.0, with the more complex model, using the COBYLA optimizer, as explained in section **??**. Doing so yields the following

- Accuracy score: 0.78
- Loss function: 0.49

which is strikingly similar to the results achieved using our gradient descent method. Thus we can assume that the barren plateu problem is not the main issue with our model, but rather that the model is stuck in a local minima.