Cohort 5 Task 2 Submission: Encoding and Classifier

Quantum Open Source Foundation QC Mentorship Program

Francesco Scala

February 27, 2022

1 Introduction

Hi! This repository contains my submission to ***qosf QC Mentorship program***. I dealt with Task 2 that is the following:

- Encoding the following files in a quantum circuit mock_train_set.csv and mock_test_set.csv in at least two different ways (these could basis, angle, amplitude, kernel or random encoding).
- Design a variational quantum circuit for each of the encodings, uses the column 4 as the target, this is a binary class 0 and 1.
- You must use the data from column 0 to column 3 for your proposed classifier.
- Consider the ansatz you are going to design as a layer and find out how many layers are necessary to reach the best performance.

Analyze and discuss the results.

Feel free to use existing frameworks (*e.g.* PennyLane, Qiskit) for creating and training the circuits.

This PennyLane demo can be useful: Training a quantum circuit with Pytorch,
This Quantum Tensorflow tutorial can be useful: Training a quantum circuit with Tensorflow.

For the variational circuit, you can try any circuit you want. You can start from one with a layer of RX, RZ and $CNOTs$.

1.1 Required packages

In order to properly execute everything `Pennylane 0.21.0` and `Matplotlib 3.4.3` are needed.

1.2 Repository contents

The repository contains the data files `mock_train_set.csv` and `mock_test_set.csv`, the Python scripts `main.py` that performs the main computation and `cfr_plots.py` which gives some plots useful for comparing the different training processes. Things works properly when one first run the `main.py` file and then the `cfr_plots.py` file. Anyway, all the info collected during training and testing were already saved and are made available in the repository (`*.npy`), so one can directly run the `cfr_plots.py` file. In addition, there is also the *Images* folder containing the all the obtained plots. The complete result analysis is reported in `qosf_c5_task2.pdf` while the full cohort 5 description of the tasks is provided in the `Cohort_5_Screening_Tasks.pdf`.

1.3 Summary of the approach

After having read the data files, we encode the data features with different encoding techniques and we apply a variational circuit to classify the data items. Since, for sake of generality, we initialize the parameters of the classifier at random we will repeat the training procedure multiple times and in the end we will look at the average trend. We also repeat the same procedure for different numbers of ansatz layers in the variational circuit.

2 Encodings

2.1 Angle encoding

Angle embedding encodes N features into the rotation angles of n qubits, where $N \leq n$. In general, if there are fewer features than rotations, the remaining rotation gates are not applied. In the case considered, we use for 4 qubits to encode the four features. We renormalize the features so as to have values in $(0, 2\pi]$.

The rotations can be chosen as either RX, RY or RZ gates.

2.2 Amplitude encoding

Amplitude embedding encodes 2^n features into the amplitude vector of n qubits. To represent a valid quantum state vector, the L2-norm of features must be one. In the case considered, we only needed 2 qubits to encode the four features. This will result in a substantial speedup in terms of training time.

3 Principal functions in `main.py`

In order to read the train and test data, we at first define the `ParseFile` which returns the dataset with data features as a np.array and the correct labels as a np.array. Then we have `construct_circuit` that is a function encoding the features with the specified embedding and then applies the variational ansatz. For what concerns the variational ansatz we used two different ansatzes depending on the encoding.

For the Amplitude Embedding we used as a classifier a variational circuit only made of RY rotations and one $CNOT$ gate. The circuit is shown in Fig. 1.

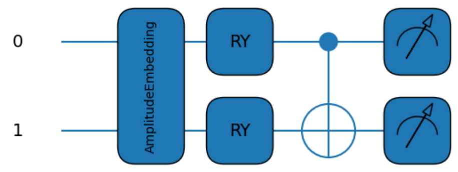


Figure 1: Quantum circuit with amplitude encoding and the corresponding variational ansatz.

Since in the case of Angle Embedding the previous ansatz seemed to struggle in classyfing correctly, we tried to increase its complexity by adding to the RY rotations also a layer of RZ rotations. Then three $CNOT$ gates are applied linearly. It should be noticed that by combining RY and RZ the whole Bloch sphere will be spanned for each qubit. The circuit is shown in Fig. 2.

We considered as output of the function the sum of the probabilities of the 2nd half of the computational base states.

To train our classifier we defined a dedicated function `train_classifier` that progressively updates the variational ansatz parameters to act as a classifier given the training set, the correct training labels, the initial rotation angles and the number of layers of the ansatz. In addition, one can also evaluate the model performances during the training by passing test set and labels. As a cost function we use the cross entropy calculated by giving in input the array containing all the output of the computation and the array of correct labels. During the training, we also display the the accuracy of the classification performed with the parameters at each step. One data item is considered to be of class 1 if the output of the `construct_circuit` function is greater than 0.5, otherwise it is considered as belonging at class 0. As an optimizer we chose Adagrad, that is a gradient-descent optimizer with past-gradient-dependent learning rate in each dimension, meaning that it adjusts the learning rate for each parameter x_i in the parameter vector x based on past gradients. We set the the stepsize at 0.25 and we repeat the training process for 30 epochs (this was fixed to have manageable training times).

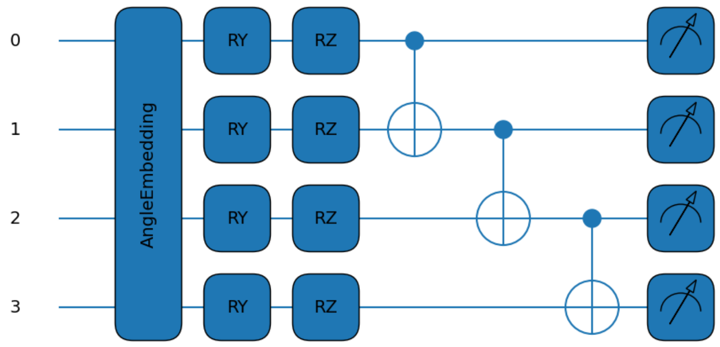


Figure 2: Quantum circuit with angle encoding and the corresponding variational ansatz.

Last but not least, we implemented the function that calculates the average trend of train and test accuracy, plotting them if `plot=True`. The function reads the train and test `.csv` files and after having randomly initialized the parameters of the classifier it calls the `train_classifier` function. Here we set a random seed in order to have reproducibility of the data. This is repeated 10 times. In addition, here we keep track of the training time in order to compare the time requirements of each approach.

4 Result analysis

Since we want to find the best possible model to classify the given dataset, we will implement variational circuits with anstaz layers number ranging from 1 to 3.

4.1 Angle encoding

The angle embedding seems to struggle learn something from the training set. As a matter of fact, as already mentioned in Sec. 2.1, we at first increased the complexity of the ansatz by adding also RZ rotations so as to try to increase the classification performances. Nevertheless this appears to not be enough to allow the model to learn something from the training set since the training accuracy is always about 50% with very little standard deviation even for different depths of the variational classifier and for different rotations in the encoding, as we can see from Figs. 3, 4, 5. In contrast the test accuracy has always a *huge* standard deviation. The optimization algorithm, probably, is not able to find the global minimum of the cost function due to the large parameter space to explore. This is strictly linked to the chosen embedding that requires one qubit per each feature to encode[1].

The only case in which there seems to be a little sign of learning, highlighted by increasing accuracy for both training and test, is the one with RZ rotations and only 1 layer in the classifier[2]. More in detail, if one increases the number of training epochs they will see that in fact there is real learning even tough the performances are not very satisfying[3] (see Fig. 6[4]).

Fig. 7 groups together all the test accuracy trends in the case of angle encoding to allow the comparison. This highlights that there is no angle encoding configuration in which the classifier efficiently predict the data labels. The best performances within 30 epochs are reached by the encoding with RX and RZ rotations both with one layer. It is to be noticed that while we said that in the case of RZ rotations there is learning, with RX from Fig. 3 we can see that the training accuracy never changes, meaning that there is no real learning.

[1]As a matter of fact, if one slightly modifies the code to use IQP embedding they will see that the same problem arises. We tried it, but we didn't include it in the report because it doesn't gives any extra information.
[2]Confirming our hypotesis that the parameter space is too large.
[3]We don't have a test accuracy above 80%.
[4]One can reproduce this plot by modifying the epochs parameter in the `train_classifier` function.

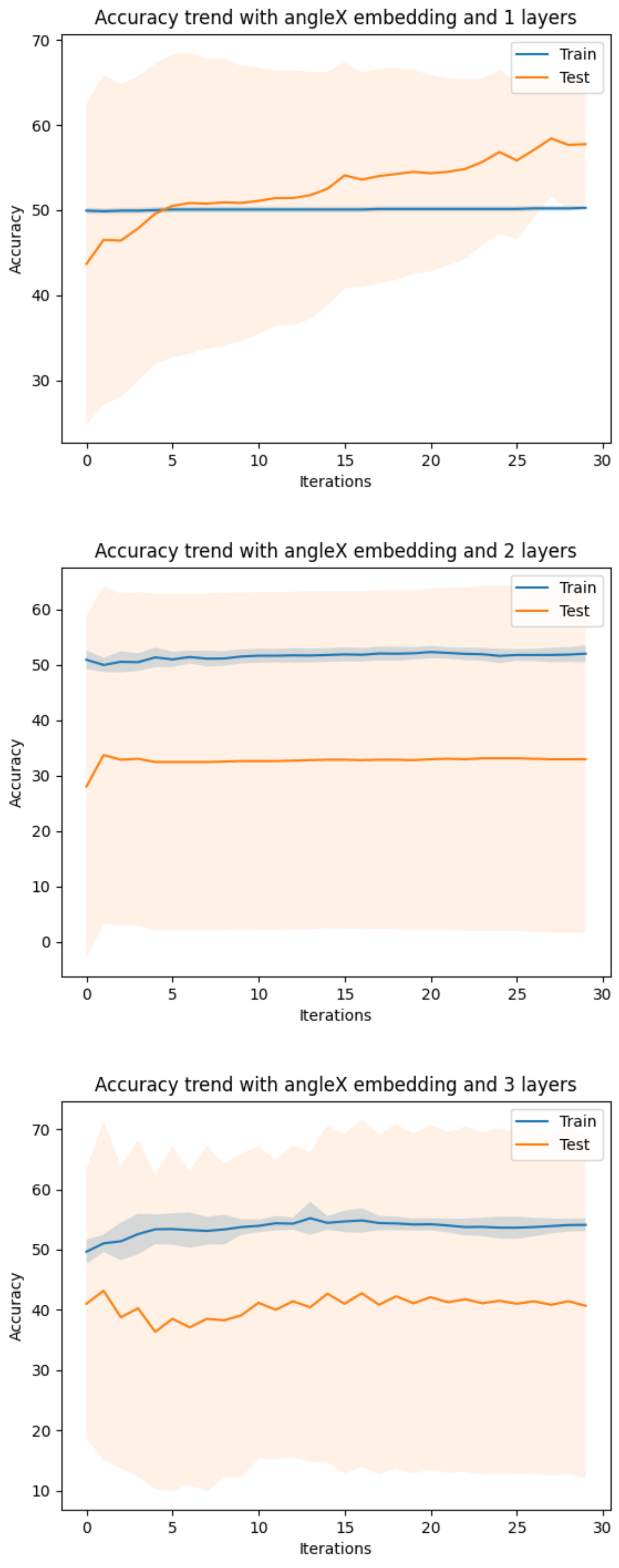


Figure 3: Train and test accuracy average trend with angle embedding with X rotations for 1, 2 and 3 layers. Also the standard deviation from the mean is highlighted.

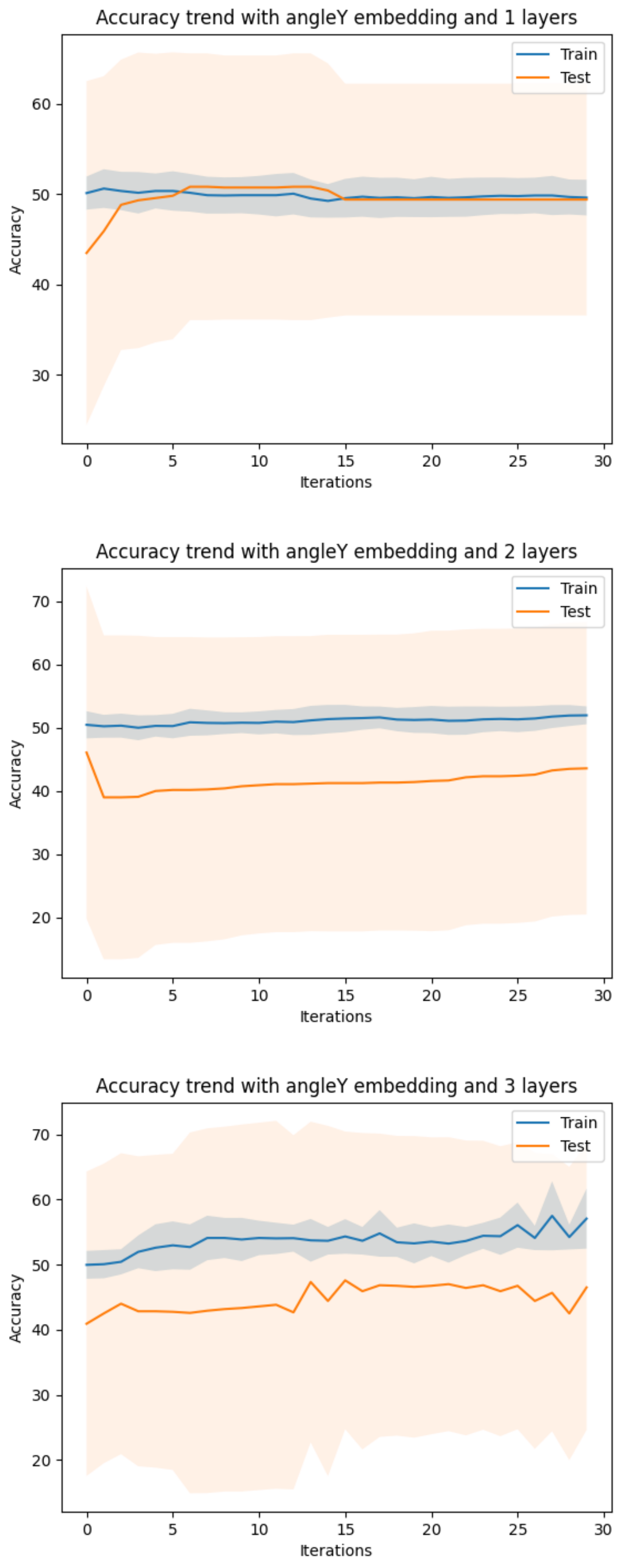
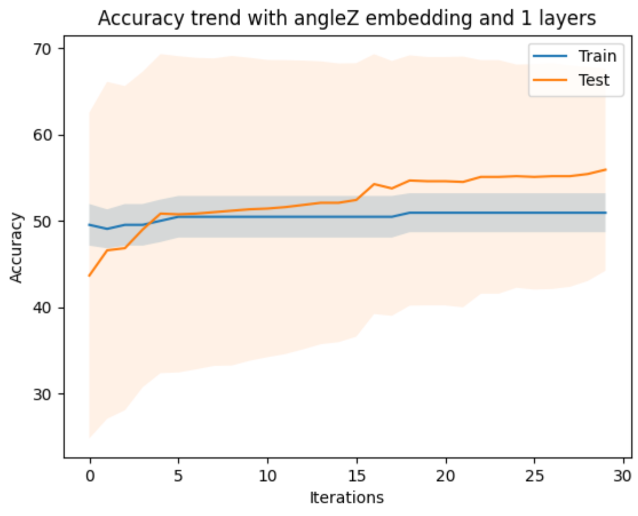
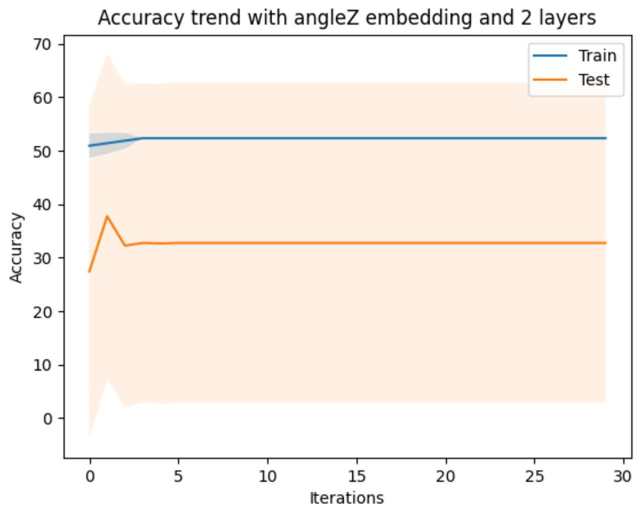


Figure 4: Train and test accuracy average trend with angle embedding with Y rotations for 1, 2 and 3 layers. Also the standard deviation from the mean is highlighted.





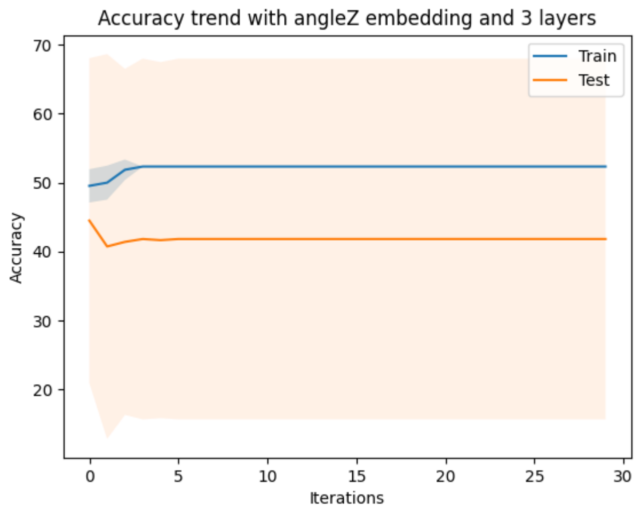


Figure 5: Train and test accuracy average trend with angle embedding with Z rotations for 1, 2 and 3 layers. Also the standard deviation from the mean is highlighted.

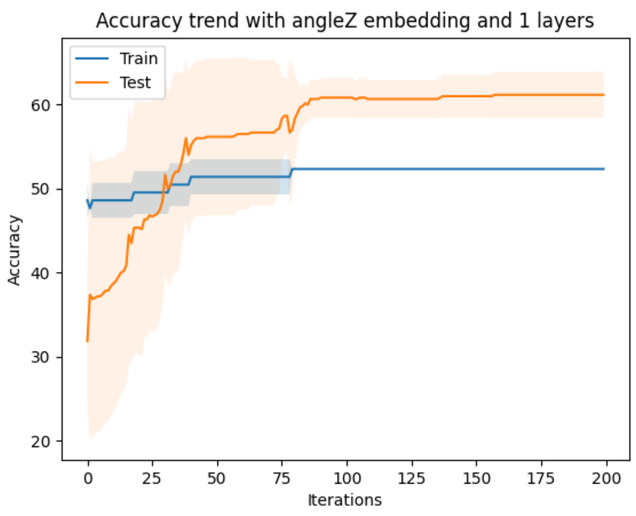


Figure 6: Train and test accuracy average trend over 200 epochs with angle embedding with Z rotations for 1 layer. Also the standard deviation from the mean is highlighted.

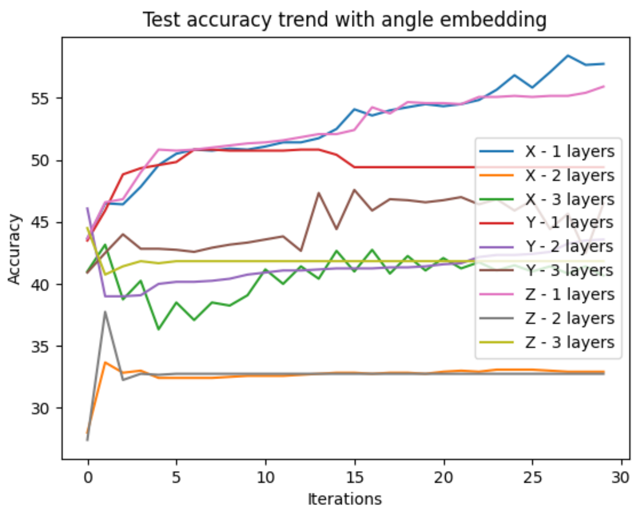


Figure 7: Comparison of test accuracy trend for angle embedding with different number of layers in the variational classifier and different embedding rotations.

4.2 Amplitude encoding

On the other hand, the amplitude embedding strategy works extremely well reaching a test accuracy of 100%. As a matter of fact, as one can see from Fig. 8, only in the case with 1 layer the classifier cannot reach a satisfying performances within the range of 30 epochs[5].

Fig. 10 shows a plot with the comparison of the test accuracy reached for the different variational circuit depths. From this we can state that the best classification performances are reached with 2 layers and amplitude embedding, where the 100% test accuracy is always met[6], whereas with 3 layers the model doesn't always get to such value. This means that increasing the depth gives, hence increasing the dimension of the parameters space, only puts the optimization algorithm in difficulty in finding the global minimum of the cost function.

[5]Increasing the number of epochs leads to a plateau in the performances after about 40 epochs, see Fig. 9. One can verify this by modifying the epochs parameter in the `train_classifier` function.

[6]As we can see from Fig. 8 the standard deviation goes to 0.

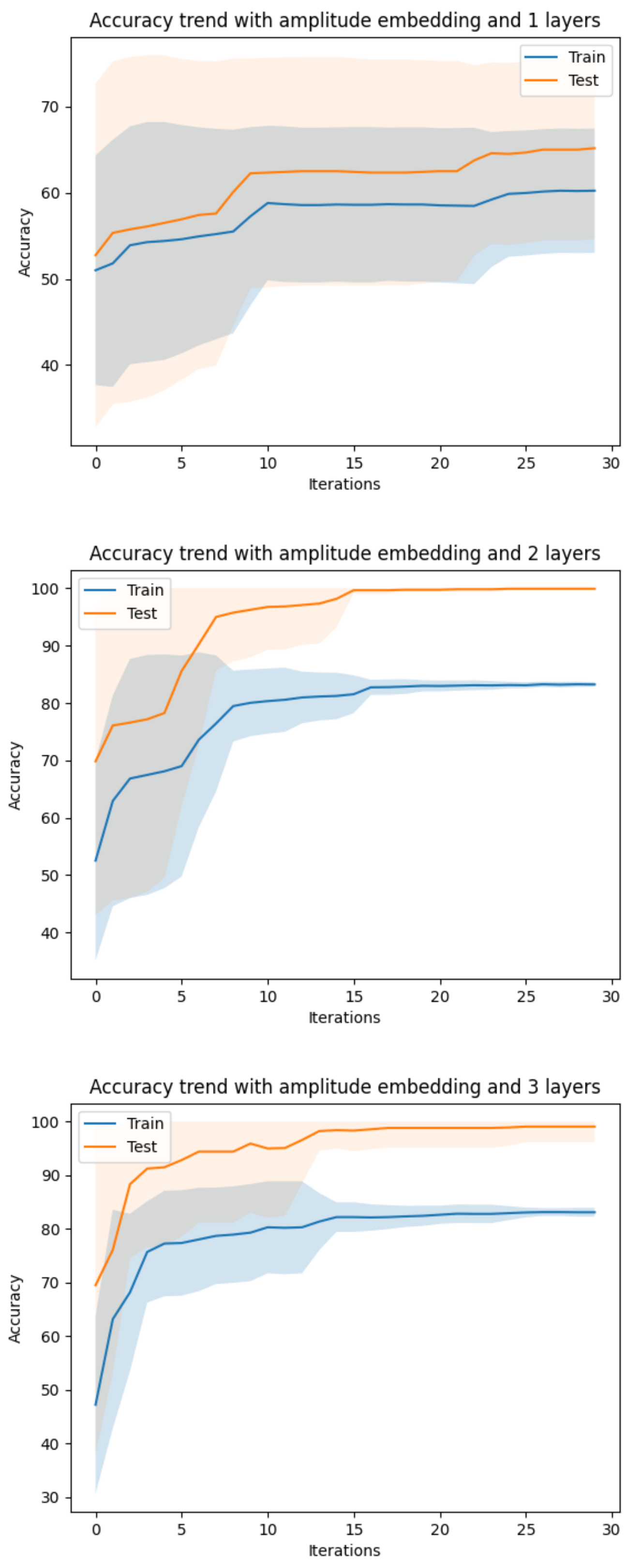


Figure 8: Train and test accuracy average trend with amplitude embedding for 1, 2 and 3 layers. Also the standard deviation from the mean is highlighted.

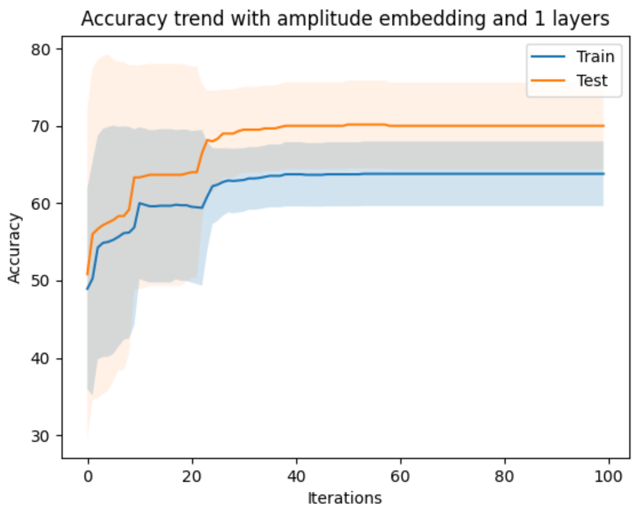


Figure 9: Train and test accuracy average trend over 100 epochs with amplitude embedding for 1 layer. Also the standard deviation from the mean is highlighted.

We can conclude that probably the angle embedding was not a good fit for the given dataset.

4.3 Average training time

Also from a computational point of view the amplitude embedding is the best choice. In fact, since in that case only two qubits are needed to encode the features, the time demand to simulate each circuits, and as a consequence the time to complete each training of the model, is far way lower with respect to angle embedding which in contrast needs four qubits to encode all the information. This is shown in Fig. 11.

As one would expect, the time cost scales linearly with the number of layers for all the kinds of encoding.

5 Conclusion

In conclusion, we can state that using an encoding strategy that exploits a large number of qubits or adding too many layers to the variational circuit only enlarges the parameter space and, as a consequence, it confuses the optimization algorithm. The best classification model, in terms of both test accuracy and average training time, is the one combining amplitude encoding on 2 qubits and a variational circuit made of two layers with RY rotation gates and one $CNOT$ gate. In fact it always reaches test 100% accuracy within a little time.

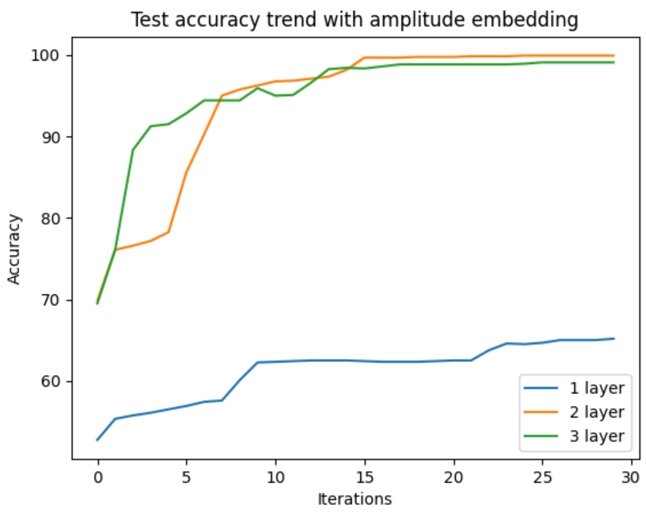


Figure 10: Comparison of test accuracy trend for amplitude embedding with different number of layers in the variational classifier.

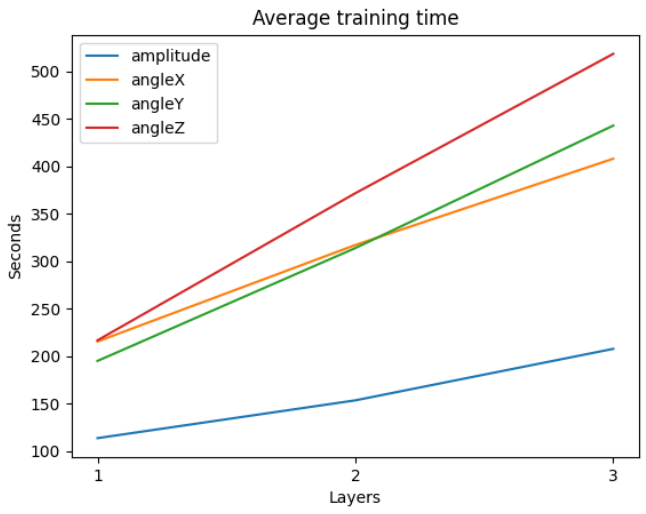


Figure 11: Average training time as a function of the number of layers for all the encodings.