Quantum Clustering Algorithm

APPROACHING UNSUPERVISED CLUSTERING AS AN OPTIMIZATION PROBLEM

Group 3 - Final Report

TN3175 Quantum Engineering Group Project
2022/23 Q2

The project repository can be found on GitHub [1]

AUTHORS
Arnav Chopra
Jules Drent
Rens Dur
Ion Plamadeala
Oliver Sihlovec

COURSE STAFF
M. Veldhorst
G. Scappucci
M.F. Russ
A.S. Ivlev

Delft University of Technology
Faculty of Applied Sciences

January 2023

Abstract

Here we present a quantum data clustering algorithm, using a Variational Quantum Eigensolver (VQE). The algorithm allows us to distinguish data points, with multiple features, into several clusters. The basis of the algorithm is to treat the clustering problem as an optimisation problem and use the VQE method to solve it. Our approach relies on encoding each data point onto a Bloch sphere and minimising a cost function in order to optimise the clustering of the data points, into a specified number of classes. We use qubit-states as reference states for each cluster, the relation between these reference states and data points are the parameters that we attempt to optimise in the quantum algorithm. We apply the algorithm to a variety of datasets, such as datasets with multiple features and multiple classes, in order to benchmark the algorithm with an assortment of inputs. We utilise singular qubit systems as well as systems with multiple qubits, for more complex inputs. Furthermore, we parallelise our circuit, in order to compute several data points simultaneously, such that it can be efficiently implemented on quantum hardware. We benchmark our algorithm using real datasets, and find that it is comparable to classical clustering algorithms in terms of accuracy. The algorithm can also be extended to accommodate higher dimensionality.

1 Introduction

Quantum computing experienced a notable development in the previous year. IBM unveiled the 433-qubit quantum processor Osprey and is set to increase this number up to four thousand by 2025 [2]. This breakthrough has sparked significant interest among researchers and enterprises, which strive to utilise quantum computers in a wide variety of fields. In addition to its other applications, quantum computation has been employed in the domain of machine learning. For example, quantum neural networks [3] or support vector machines [4] have yielded promising results for label prediction, or in other words classification, of previously unseen data. Unlike classification tasks native to supervised learning, clustering or clustering analysis aims to group a set of unlabeled data points into smaller subsets, such that the average similarity between points belonging to the same subset is higher than that of objects from other subgroups [5]. While ample classical algorithms for such tasks exist, efficient quantum variations thereof, resilient to limitations of current hardware, have yet to gain traction. For instance, the quantum K-Means clustering algorithm [6] cannot be implemented for complex settings on current NISQ devices due to insufficient computing power such prototypes offer.

This report outlines our implementation of a quantum clustering algorithm. We accomplish this by utilising a quantum optimiser called Variational Quantum Eigensolver (VQE) that optimises for a cost function describing the fitness of an assignment of datapoints to clusters. More specifically, each input data point is first represented as a quantum state, which the variational circuit transforms into a new state that has a high fidelity with the reference state of its corresponding cluster. This technique achieves clustering accuracy comparable to those of classical algorithms as well as being able to cluster quantum data. To our knowledge, existing implementations assume feature vectors whose dimensionality does not exceed two, however, our algorithm is compatible with an arbitrary feature space. Finally, our circuits are parallelisable in order to allow for clustering of multiple data points without having to construct them for each input individually, hence reducing the overhead.

2 Methods and Implementation

In order to discover the capabilities of this algorithm, several experiments were carried out. Each time, the experiments got more sophisticated and attempted to achieve a greater goal. Most experiments provide an extension to the ones before. This chapter contains an extensive explanation of the algorithms and the parts it is composed of. Most of this explanation can be found under the details of experiment 1. Further experiments may require understanding of the implementation used in experiment 1. A high-level overview of the algorithm can be found in section 7 in the appendix.

2.1 Clustering Two-Dimensional Points Into Two Clusters

As a simple start, it was decided to implement a VQE optimiser that can be used to cluster a small group of two-dimensional points into two different clusters. Because of its simplicity, this experiment could serve as a reference for more complex problems of similar nature. However, the most important reason for this decision was that most aspects of the implementation are relatively easy to visualise. For example, two-dimensional points can easily be visualised in a scatter-plot and grouping the points in two clusters allows selecting two orthogonal reference-states on the Bloch-sphere. In this experiment, the two reference points that were selected are $|0\rangle$ and $|1\rangle$. These reference points are used in the following way: if a data point, after being run through the quantum circuit, ends up in a state that has a higher fidelity with state $|0\rangle$ than with $|1\rangle$, it will be assigned to cluster 0. A similar thing happens when assigning a point to cluster 1.

This optimisation problem consists of the following parts:

1. A cost-function or objective-function that is used to assess how well the selected optimisation algorithm was able to split the set of data points into two clusters.

2. A variational circuit-form that is used as an ansatz for the solution. This circuit contains inputs for the properties of the individual data points, as well as

the parameters that are optimised by the optimisation algorithm.

3. A parameter-optimisation algorithm.

2.1.1 Constructing the Cost Function

The cost-function, used this experiment, was inspired by the article '*The Variational Quantum Eigensolver: A review of methods and best practices*' [7].

The optimiser, which will be explored in a later section, will try to find optimal values for the parameters that are part of the variational quantum circuit, by minimising the outcome of the cost-function. The cost-function in Eq.1 is used in this experiment:

$$H = \frac{1}{2} \sum_{i,j=1}^{N} d(\vec{x}_i, \vec{x}_j) \sum_{a=1}^{K} f_i^a f_j^a \tag{1}$$

The main component of this cost-function is the distance metric $d(\vec{x}_i, \vec{x}_j)$ that represents the (classical) distance between the two data points, indexed by i and j. Because it is only desired to minimise the cost for two data points that belong to the same cluster, two fidelity-terms are included in this function. f_i^a represents the fidelity between data point i, after it is rendered through the quantum circuit, and the reference point a. f_j^a does the same for data point j. By multiplying the distance metric by these two fidelities, the optimisation algorithm will attempt to reduce the cost by assigning nearby points to the same cluster. When the optimiser decides to changes (one of the) parameters, the algorithm may decide to place data points in different clusters, at which point the fidelities with the corresponding reference-point will increase. The optimiser therefore changes the fidelities, until it has found a minimum for the cost-function. The distance-metric for the input data points never changes. Lastly, the $\frac{1}{2}$ pre-factor, in Eq.1, is included since combinations (i, j) are encountered twice by the summation. A more detailed explanation of the construction of this cost-function can be found in section 8.1 in the appendix.

2.1.2 Constructing the Variational Circuit

Since this experiment is solely concerned with classifying points into two different clusters, no more than one qubit is required. As mentioned earlier, $|0\rangle$ and $|1\rangle$ will be the respective reference points for the two clusters. Data points whose resulting quantum state, after being processed by the circuit, has greater fidelity with a particular reference states, will be placed in that cluster.

The resulting variational circuit will thus remain fairly simple and only consists of two unitary transformations. One that serves as an input for the two features of the object that is currently considered. The other serves as an input for the two optimisation parameters that are slowly altered by the optimiser. This circuit is illustrated in Figure 1.

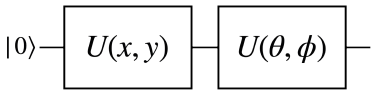

Figure 1: Single Qubit Variational Circuit

2.1.3 Selecting an Optimiser

When selecting an optimiser, it is important to realise that most optimisers will act differently when run on a quantum simulator as compared to on real quantum hardware. The most important reason for this effect is that real quantum hardware can be impacted greatly by presence of quantum noise in all sorts of forms. This noise changes the landscape of the objective function, thereby changing the gradient along its paths. The optimisers may, as a result, select a different direction for convergence towards a (local) minimum. It is thus very likely that an optimiser works well on a simulator, but not on quantum hardware, or the other way around.

2.1.4 Implementation

Now that all parts have been individually discussed, it is time to compile them into the final algorithm used in this experiment.

As with most data processing tasks, the better the input data, the better the overall output. For this process, the data processing means normalising the data between 0 and 1 in order to correctly map the data to the Bloch sphere. For this problem, we have two datasets: one consisting of two clearly separated Gaussian clusters, and one consisting of concave clusters with point aligned in a semicircle.

The variational circuit, defined in section 3.1.2, has two gates that need to be specified. To build up the first gate, $U(x, y)$, we map each point from the interval $[0, 1]$ to $[0, \pi]$. The other gate $U(\theta, \phi)$ is made out of parameters that are initialised at random.

Once the problem has been defined, and the reference vectors have been chosen, we can start the optimisation process. Each iteration of the optimisation starts with finding out the state vectors for each data point, by running each data point through the variational circuit. The results are then used to compute the fidelities f_i^a of the state-vectors of the data points with the reference states. The fidelities are then plugged into the cost function, which gives the total cost for that iteration. Once the optimiser has the results of that iterations, it adapts the parameters to improve the cost function until either the result does not change or the maximum number of iterations is hit.

Computing the final results Once optimal parameters have been found, they are used to run to cluster the points one last time. We recompute the state vectors with those parameters, and compute the fidelity of the state vector of each point with the reference state vectors, choosing the

cluster for which the fidelity is the highest. Those represent the final clusters in our algorithm.

2.2 Clustering Two-Dimensional Points Into Four Clusters

A logical next step would be to try an extend this algorithm to cluster two-dimensional data points into four different clusters. In essence, the experiment remains exactly the same, except an additional qubit is required to accommodate the two extra clusters. The reference points have now become combinations of the output states of both qubits: $|00\rangle, |01\rangle, |10\rangle$ and $|11\rangle$.

The biggest change to the algorithm that is needed to allow clustering into four clusters, is the variational circuit (ansatz) that is used. The goal is to seek a circuit that allows room for more optimisation parameters, as well as a way for the first and second qubit to 'exchange information' in some way. The latter is required as the first and second qubit *together* decide the cluster assignment for a given data point. Just like in experiment 1, the ansatz presented in the article '*Variational Quantum and Quantum-Inspired Clustering*' [7] is used and it is presented in Figure 2.

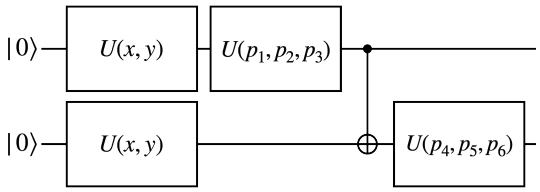

Figure 2: Double Qubit Variational Circuit

When considering this ansatz for the second experiment, it was also decided to opt for six optimisation parameters instead of four, which would be the amount if you extrapolated exactly from experiment 1 in section 2.1. It was expected that adding two optimisation parameters would increase the performance of the clustering algorithm, as it gives the optimiser more room to alter the behaviour of the circuit. In Figure 2, the optimisation parameters are denoted p_1 through p_6.

2.2.1 Constructing the Cost Function

By the same token, the cost-function of the algorithm must also be changed. Using the cost-function from Eq.1, we can create a modified cost-function to further optimise for more refined clustering. The function uses cluster centroids, represented as c_i. This term is the centroid belonging to the same cluster as data point i. We also introduce hyperparameters α and λ. Here, α allows for modified penalisations for the distance between points, and λ accounts for the relative importance of the distance between point $\vec{x}_i$ and centroid $\vec{c}_i$. The modified cost function is shown in Eq.2.

$$H = \sum_{i,j=1}^{N} (d(\vec{x}_i, \vec{x}_j)^\alpha + \lambda d(\vec{x}_i, \vec{c}_i)) \sum_{a=1}^{K} (1 - f_i^a)(1 - f_j^a) \quad (2)$$

Furthermore, we use penalty terms in order to account for scenarios where the number of clusters is not of the form 2^N, where N is the number of qubits. This is to ensure that the data-points are only clustered into the relevant reference states. The penalty term p_i for a data point is represented as shown in Eq.3.

$$p_i = \left(\sum_{a=1}^{K} f_i^a - 1 \right)^2 \quad (3)$$

The final cost-function can be obtained by adding the penalty terms from Eq.3, to the cost-function from Eq.2.

2.2.2 Required Changes

The implementation described in 2.1.4 provides a great fundamental for the code that is required to run this experiment. The biggest change that had to be carried through lies around the construction of the variational circuit. The circuit now has to account for four more optimisation parameters and an additional CNOT-gate should be inserted. In terms of measuring state-vector fidelity between an output state-vector and a particular reference vector, not much had to change. The previous implementation used for computing these fidelities 2.1.4 is based on functionality, provided by Qiskit [8], that already accepts higher dimensional input state-vectors. This is in line with the mathematical description of quantum states that are spread among multiple qubits.

A few other small components in the implementation had to be changed. Firstly, the list of initial parameters, required by the optimiser, had to be extended to six parameters. And secondly, the list of reference points had to be extended.

2.2.3 Scaling Further

By looking closely at the required changes to scale the algorithm into a maximum of four clusters, it is also possible to scale the algorithm even further. As with many quantum algorithms, this can likely be done until the circuit depth outgrows the maximum circuit depth allowed by the quantum hardware that the algorithm is performed on.

As an example, a variational circuit for unsupervised clustering using five qubits is presented in Figure 11 in Appendix A. As is clear from the figure, such five-qubit setup has room for 10 features, 15 optimisation parameters and is able to cluster the data points into 32 different clusters. That is, with two reference points per qubit.

2.3 Clustering Four-Dimensional Points Into Four Clusters

We have seen how our algorithm can be scaled to multiple qubits in the previous section. We have used systems with multiple qubits to extend the number of clusters in our algorithm, the maximum number of possible clusters is 2^N with N qubits. Naturally, the next course of action would be to further scale our algorithm with multiple

qubits. Specifically, we want to increase the number of features allowed in our algorithm. We are also still able to keep the same scaling of the number of clusters, as discussed in the previous section. This section will look at the implementation of multiple features using two qubits specifically.

2.3.1 Implementation

Since we are extending the functionality of our algorithm, the variational circuit (ansatz) must be updated. Thus, we must change the circuit which is used in the previous experiments. In order to optimise our implementation, we use four layers of y and z rotations, per qubit. Each layer of rotations is followed by a CNOT-gate, in order to facilitate 'information exchange' between qubits. A diagram of this updated variational circuit is presented in Figure 12 in Appendix A.

Due to the updated ansatz, we must also update the number of optimisation parameters we use. We have opted for sixteen total optimisation parameters, thus we have two parameters per layer for each qubit.

The cost function used in this experiment is represented in Eq.2.

It is important to note that the system is not bound to 2^N clusters, despite using multiple qubits. We may use less than 2^N features, depending on the dataset. Similarly, the system is not necessarily bound to $2N$ features.

2.3.2 Scaling Further

This experiment has presented an implementation of four features using two qubits. In principle, the scaling of this circuit is quite simple. For N qubits, we can have up to $2N$ features. Furthermore, the number of optimisation parameters also scales linearly with the number of qubits, although the exact number of features depends on the number of rotation layers in the circuit, which can be altered arbitrarily to a certain extent.

2.4 Parallel Qubit Clustering

An interesting property of the first experiment is that it uses only one qubit, because there are multiple ways in which that experiment can be scaled. In the second experiment, it was attempted to scale into the 'more clusters'-direction, using an additional qubit. In this experiment, the goal is to scale in terms of training time. The main idea is that many quantum computers that exist today have more than one qubit, for example five of seven qubits [2], but this clustering-algorithm utilises only one at a time. Since the algorithm requires only one qubit (and therefore does not require any connections between qubits), it can be performed on several data points at a time, utilising all available qubits on the selected hardware.

In this experiment, an algorithm will be constructed that can be run on a quantum simulator that has N qubits.

2.4.1 Parallelisation: Computing the Results of Multiple Data Points Using A Single Circuit

Each iteration of the training phase, the cost-function (objective-function) must be evaluated. This computation requires that each data point is pulled through the variational circuit using the hyperparameters that the optimiser has currently selected. Since the optimisation parameters do not change between data points inside this iteration of the training-phase, data points can in principle be pulled through the circuit simultaneously.

The circuit that is used in this experiment utilises N qubits, but the qubits do not have interactions between each other. At first glance, this suggests that running the algorithm on multiple qubits simultaneously would not affect the quality with which the individual qubits can perform the calculation. In other words, having more qubits should not affect quality of single-qubit-evaluation.

Unfortunately, this is not the case. Having more qubits affects the noise-levels in the circuit and therefore definitely has an effect on the quality of single-qubit-evaluation. Having a larger number of qubits introduces unwanted interactions between the qubits. And since this algorithm requires no such interactions, the effect of unwanted interactions is likely to be significant.

This experiment opts to find out whether the speed-up due to this parallelisation weighs out the extra time required to compensate for the added noise between qubits.

2.4.2 Constructing the Variational Circuit

The biggest change that is required for this experiment lies within the construction of the variational circuit. Based on the number of available qubits, N, a circuit should be composed that includes the information about N data points and that applies the same optimisation parameters to each of them. As an example, Figure 3 presents the variational circuit for a five-qubit machine.

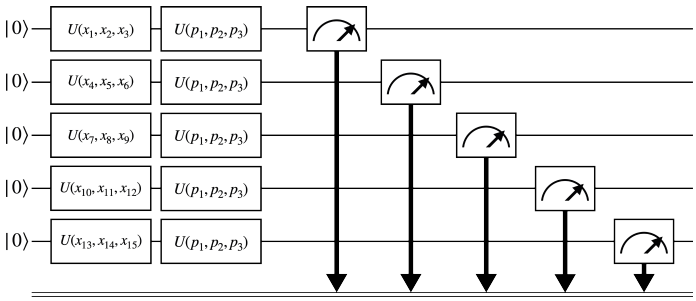

Figure 3: Parallel Qubit Circuit with five qubits

Of course, running multiple data points through one circuit reduces the number of circuits that have to be evaluated. The first N data points are evaluated by the first circuit, the second group of N data points by the second circuit, and so on. If the number of data points is no exact multiple of N, the last circuit will simply have to evaluate a smaller amount of data points.

But, there is one other very important thing that must be changed about the variational circuit. The outcome of the

qubit measurements will no longer be simply the result of a single qubit, but the combined results of the N qubits. This combined result first has to be converted into the exact same type of result we opted for in the first experiment (section 2.1), per qubit.

After the circuit has been completely executed, a measurement is performed on each qubit. By doing this many times, the number of times each of the qubits falls into state $|0\rangle$ or $|1\rangle$ can be obtained. In essence, these counts can be used as fidelities with those respective states (which are of course the reference states used in this experiment). This is done by dividing the number of times a qubit-measurement results in one of the reference states by the total number of (simulation) shots. These fidelities can then be used to evaluate the cost-function.

3 Results and Discussion

In this chapter, the experiments are tested on multiple input data sets and their results are presented in the form of plots. The plots contain the display the clustering decisions made by the algorithms. Results are grouped by the experiments they belong to.

3.1 Single qubit clustering

The experiment is to show a proof of concept on artificially generated data. As seen in Figure 4, the algorithm correctly clusters two different groups distributed in the y direction but concentrated on the x direction. It correctly clusters the two groups, with an accuracy of 100%.

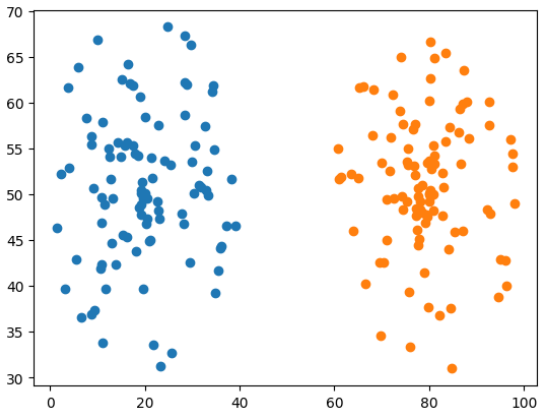

Figure 4: Results of the single qubit clustering algorithm on a dataset containing two Gaussian-distributed clusters.

This dataset was built to find the limitations of the clustering algorithm compared to classical clustering algorithms that revolve around centroids. The result in Figure 5 shows that the clustering algorithm works by dividing the points with a line rotated along the centre. It finds the rotation angle where the number of points in each cluster is similar to the other one. That is indeed a consequence of the way we choose the reference points. They are orthogonal and the fidelity cut-off would be in the middle point of the feature space, on both axes.

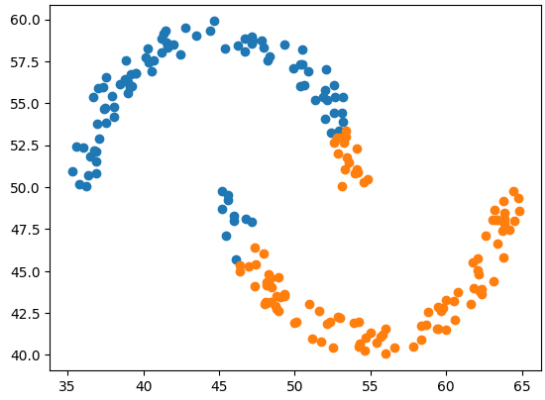

Figure 5: Results of the single qubit clustering algorithm on two semicircles.

3.2 Multiple qubit clusters

To gauge the performance, we validate the two-qubit variational circuit on three artificially created datasets. The simulations utilised the cost function from Eq.2. In Figure 6(a) the algorithm is tasked to cluster a dataset of three Gaussian distributed groups. As seen in the figure, the algorithm performs well and can achieve an accuracy of 100% in approximately 20 to 30 epochs. In contrast, it was unable to differentiate between the adjacent groups in Figure 6(b). This is not necessarily a flaw of the ansatz, but can also be attributed to shortcomings in the cost function. The cost function penalises distant data points, hence they are pushed to different clusters. As a result, intricate structures in the data are lost. This flaw can also be seen in Figure 6(c). The dataset contains two distinct clusters with another cluster sparsely distributed between them. The algorithm performs relatively well, however, it is unable to classify the data points in the centre correctly.

3.3 Multiple features

We tested our approach to multi-feature clustering on the Iris dataset [9]. The dataset contains four features and 50 samples from each of the three species of Iris. The simulation utilised the cost function from Eq.2 with $\alpha = 1$ and $\lambda = 0.725$. This yielded an accuracy of 96% in 13 epochs with a learning rate of 0.1. The figure displays that the algorithm can distinguish overlapping clusters which would be classified incorrectly if the feature space was two-dimensional.

However, for all the parameters in the variational circuit, the gradient has to be determined. Therefore, the circuit needs to be run 17 times per iteration, which makes optimisation lengthy. Moreover, the algorithm is sensitive to the initial choice of optimal parameters and more importantly, the model hyperparameters. Choosing the wrong hyperparameters can result in lower accuracy and longer optimisation-time. This poses a problem to real-world applications in which the performance of the model cannot be compared to the labels of the data which would make choosing the hyperparameters problematic. Compared to our approach, some classical clustering algorithms can achieve a slightly higher accuracy on the

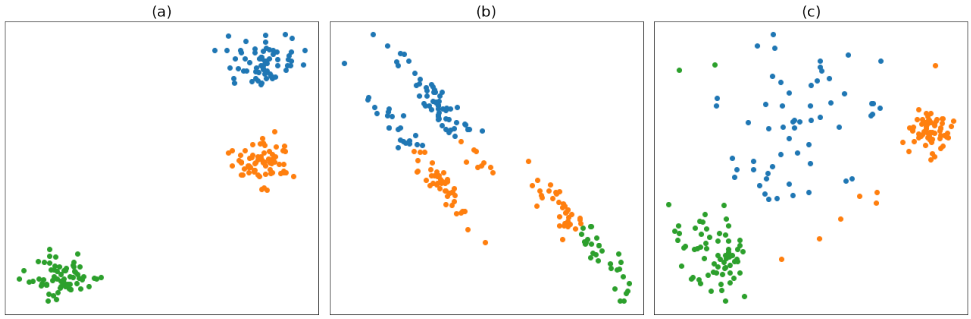

Figure 6: Clustering results, obtained with a double qubit variational circuit. The performance of the algorithm was validated on three artificial datasets

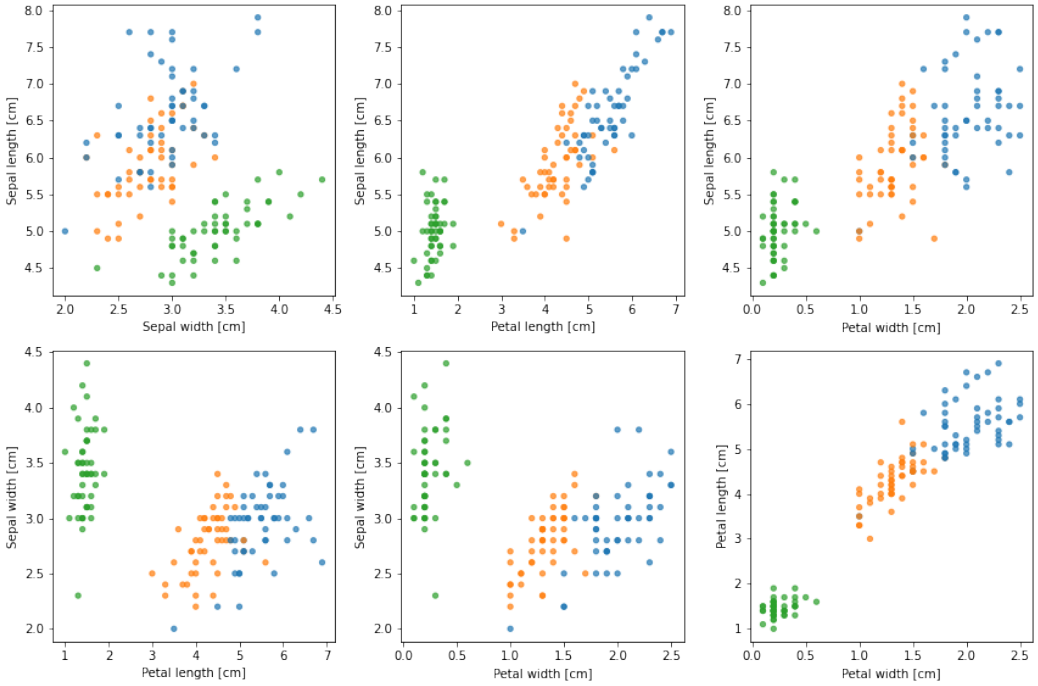

Figure 7: Clustering results of the multiple feature algorithm on the Iris dataset. The Iris dataset contains four features and 150 samples. The algorithm was able to achieve an accuracy of 96% in 10 epochs

Iris dataset [10] [11]. However, this is dependent on the algorithm and they can also underperform.

3.4 Parallel Qubit Clustering

The goal of parallelising the single-qubit algorithm, to cluster data points between two clusters, was to find out whether the speed up due to parallel execution would weigh out the extra execution-time required because of additional noise. It is important to mention that, within the time-span allocated for this project, it was only possible to obtain results from a quantum simulator. This is a very interesting experiment that is definitely worthwhile expanding to real quantum hardware in the future.

First, a baseline had to be created in order to allow for good comparison with variations in the number of qubits, number of shots or number of iterations for the optimiser. This meant that the algorithm had to be run with one single qubit. The other configurables were set to 256 shots per circuit and a maximum of 100 iterations for the optimiser. All variations are performed on the same 100-point dataset. In order to obtain a reliable result, the runtime is averaged over twenty runs. The resulting runtime of this baseline is presented in table 1 on the first row.

Having set the baseline, it is now time to alter some configurations. Just like the baseline, each configuration is run twenty times to make the recorded runtime more reliable. Table 1 presents all the performed variations together with their resulting runtime and Figure 8 displays the average runtime, over twenty runs, against the number of qubits that were used in that experiment.

Qubits	**Shots**	**Max. iterations**	**Runtime** (s)
1	256	100	45.15
2	256	100	51.02
3	256	100	44.33
4	256	100	38.92
5	256	100	39.48
6	256	100	43.89
7	256	100	50.88
8	256	100	66.03

Table 1: Variations and resulting run-time

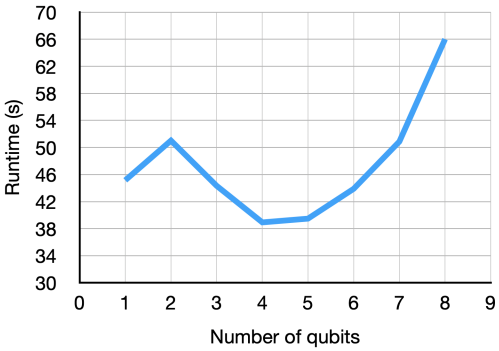

Figure 8: Average runtime, over twenty runs, as the number of qubits increases

As becomes clear from figure 8, increasing the number of data points, that are simultaneously (by adding qubits) being pulled through the circuit, seems to increase the performance up to a certain limit. In fact, the performance becomes worse than the baseline single qubit experiment when using seven or more qubits. From these results, the ideal number of qubits would lie around four or five. Both these experiments showed a performance increase, relative to the baseline single qubit experiment, of almost 15%.

In terms of clustering quality, the simulator showed no significant differences between lower- and higher-qubit-count circuits. Figures 9 and 10 show the clustered data points after running the algorithm on one and eight qubits, respectively.

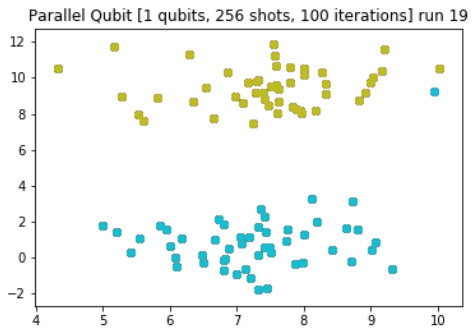

Figure 9: Clustering result after running algorithm on one qubit

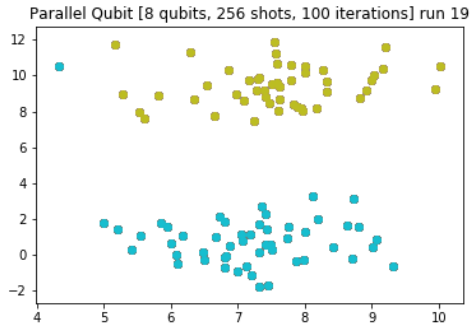

Figure 10: Clustering result after running algorithm on eight qubits

It is important to mention, again, that this experiment was carried out on the Qiskit [8] quantum simulator and not on real quantum hardware. The obtained results may give a stable representation of what happens on real quantum computers, but there will likely be differences. For example, when running this experiment on a simulator, the biggest factor that makes higher-qubit-count circuits slower is the fact that the simulator simply has to simulate more quantum computations. But if one were to run this experiment on real quantum hardware, they would probably find that this slower simulation time is (partly) replaced by higher levels of noise in higher-qubit-count circuits, which dramatically increases the required number of shots and therefore also decreases the performance. This increase in unwanted noise between qubits will likely affect the ability of the algorithm to continue clustering points into the correct clusters, as well as it did on the simulator. In that case, increasing the number of shots would be a possible solution.

4 Future recommendations

One limitation of the current implementation is the implicit cap on the maximum number of clusters. Astute readers may notice that the number of qubits, N, that we use is directly correlated with the dimensionality of the data, hence the output state vector being a superposition of 2^N basis states. Since each of the reference states corresponding to the clusters is orthogonal with respect to one another, it impossible to encode more than 2^N clusters. However, research from Bermejo and Orus (2022) introduces a technique wherein they utilise maximally orthogonal states as reference states [12]. Since we can define an arbitrary number of reference states, the dependence of the amount of clusters on the number of qubits is resolved.

Furthermore, it is also possible to increase the number of features per quantum circuit by assigning each feature to a distinct basis state and encoding its value as the respective amplitude [13]. This would allow us to encode an order of 2^N as opposed $2N$ features, which presents an opportunity for future enhancement.

An additional point of improvement would be a thorough analysis of additional variational circuits. Due to the complex nature of the problem, designing an ansatz specifically tailored for our task is beyond the scope of the project. That being said, constructing an optimal

circuit would undoubtedly improve both the accuracy of the algorithm by avoiding so called vanishing gradients, as well as the speed due to prohibiting the search space of the parameters. For instance, Hartree-Fock ansatz can be employed to constrain the wave function so that the trial state is similar to the actual ground state of the molecule [14].

Finally, the next logical course of action would be to run the algorithm on a quantum device. The experiments presented in this report have all been performed on a simulator and thus do not perfectly reflect a real scenario. Even though we believe that the simulator is faithful to an extent to an actual quantum computer, benchmarking our algorithm on a quantum device is imperative for verification of our work.

5 Conclusion

In this paper, we have implemented a quantum clustering algorithm. The algorithm is treated as an optimisation problem, which is then solved using a Variational Quantum Eigensolver (VQE). Each data point is encoded onto a Bloch sphere, using a variational quantum circuit comprised of parameterised rotational gates. We attempt to minimise a cost-function, which depends on the distance between each pair of data points, as well as their fidelity to a set of reference states. The algorithm optimises the set of gate parameters in order to minimise the cost. Its accuracy was then measured on a variety of datasets with multiple features and multiple classes. Furthermore, we utilise singular qubit systems as well as systems with multiple qubits, for more complex datasets. The results of our experiments demonstrate that the algorithm compares well to classical clustering variants in terms of accuracy and that it can be extended to higher dimensional feature sets. Moreover, we have introduced a computationally efficient implementation of the algorithm on quantum hardware via parallelisation. The future of the procedure is full of possibilities, ranging from encoding an exponential number of clusters (2^N), an exponential number of features(2^N), increased accuracy from different variational circuits and running it on actual quantum hardware.

References

[1] *GitHub Link to project repository.* https://github.com/RensDur/Quantum-Engineering-Group-Project-Group-3.git.

[2] IBM Quantum. *The IBM Quantum Development Roadmap.* URL: https://www.ibm.com/quantum/roadmap. (accessed: 31.01.2023).

[3] Kerstin Beer. "Quantum neural networks". en. In: (2022). DOI: 10.15488/11896. URL: https://www.repo.uni-hannover.de/handle/123456789/11991.

[4] Motohiko Ezawa. "Variational quantum support vector machine based on Gamma matrix expansion and variational universal-quantum-state generator". en. In: (2022). DOI: 10.1038/s41598-022-10677-z. URL: https://www.nature.com/articles/s41598-022-10677-z.

[5] G. Nathiya, S. C. Punitha, and M. Punithavalli. "An Analytical Study on Behavior of Clusters Using K Means, EM and K* Means Algorithm". In: *CoRR* abs/1004.1743 (2010). arXiv: 1004.1743. URL: http://arxiv.org/abs/1004.1743.

[6] Nathan Wiebe, Ashish Kapoor, and Krysta Svore. "Quantum Algorithms for Nearest-Neighbor Methods for Supervised and Unsupervised Learning". In: (2014). DOI: 10.48550/ARXIV.1401.2142. URL: https://arxiv.org/abs/1401.2142.

[7] Jules Tilly et al. "The Variational Quantum Eigensolver: A review of methods and best practices". In: *Physics Reports* 986 (Nov. 2022), pp. 1–128. DOI: 10.1016/j.physrep.2022.08.003. URL: https://doi.org/10.1016%2Fj.physrep.2022.08.003.

[8] *Qiskit Open-Source Quantum Development.* https://qiskit.org. Accessed: January 2023.

[9] R.A. Fisher. *UCI Machine Learning Repository.* 1936. URL: http://archive.ics.uci.edu/ml.

[10] L. Svetlova, Boris Mirkin, and H. Lei. "MFWK-Means: Minkowski metric Fuzzy Weighted K-Means for high dimensional data clustering". In: Aug. 2013, pp. 692–699. DOI: 10.1109/IRI.2013.6642535.

[11] Adrija Chakraborty et al. "Comparative Study of K-Means Clustering Using Iris Data Set for Various Distances". In: *2020 10th International Conference on Cloud Computing, Data Science Engineering (Confluence).* 2020, pp. 332–335. DOI: 10.1109/Confluence47617.2020.9058328.

[12] Pablo Bermejo and Roman Orus. *Variational Quantum Non-Orthogonal Optimization.* 2022. DOI: 10.48550/ARXIV.2210.04639. URL: https://arxiv.org/abs/2210.04639.

[13] Maria Schuld and Francesco Petruccione. "Information Encoding". In: *Supervised Learning with Quantum Computers.* Cham: Springer International Publishing, 2018, pp. 139–171. ISBN: 978-3-319-96424-9. DOI: 10.1007/978-3-319-96424-9_5. URL: https://doi.org/10.1007/978-3-319-96424-9_5.

[14] Frank Arute et al. "Hartree-Fock on a superconducting qubit quantum computer". In: *Science* 369.6507 (2020), pp. 1084–1089. DOI: 10.1126/science.abb9811. eprint: https://www.science.org/doi/pdf/10.1126/science.abb9811. URL: https://www.science.org/doi/abs/10.1126/science.abb9811.

[15] Pablo Bermejo and Roman Orus. *Variational Quantum and Quantum-Inspired Clustering.* 2022. DOI: 10.48550/ARXIV.2206.09893. URL: https://arxiv.org/abs/2206.09893.

6 Appendix A: Figures

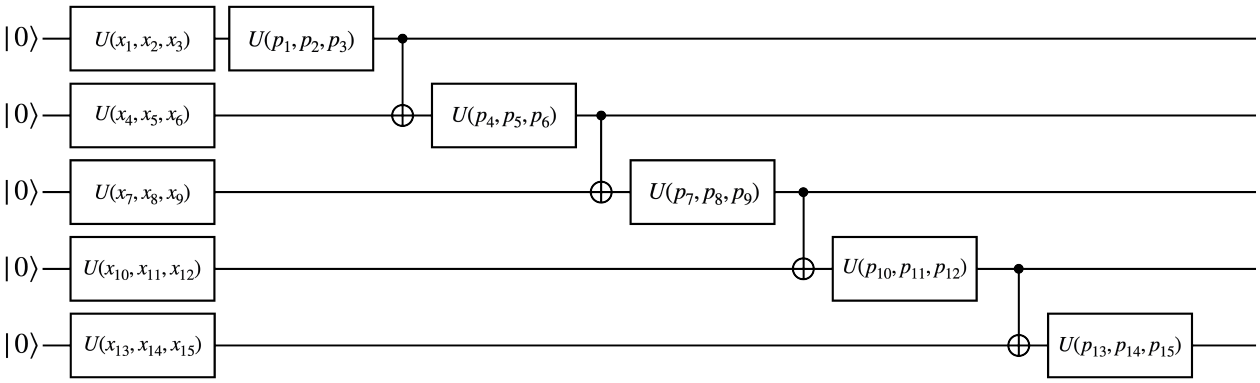

Figure 11: Five Qubit Variational Circuit

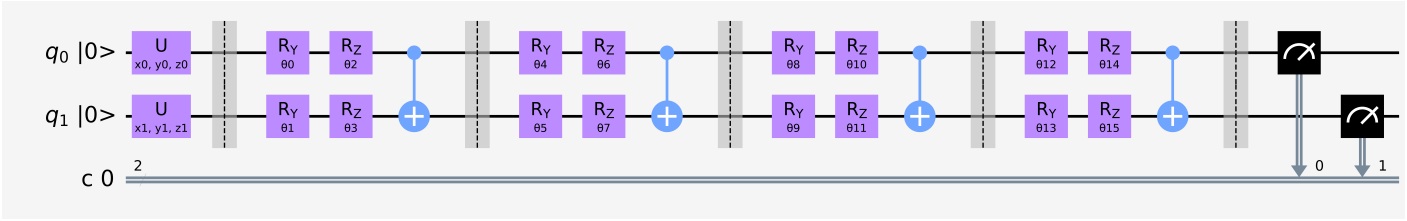

Figure 12: Variational Circuit for Four Features

7 Appendix B: VQE Algorithm flow

Overview of the algorithm flow:

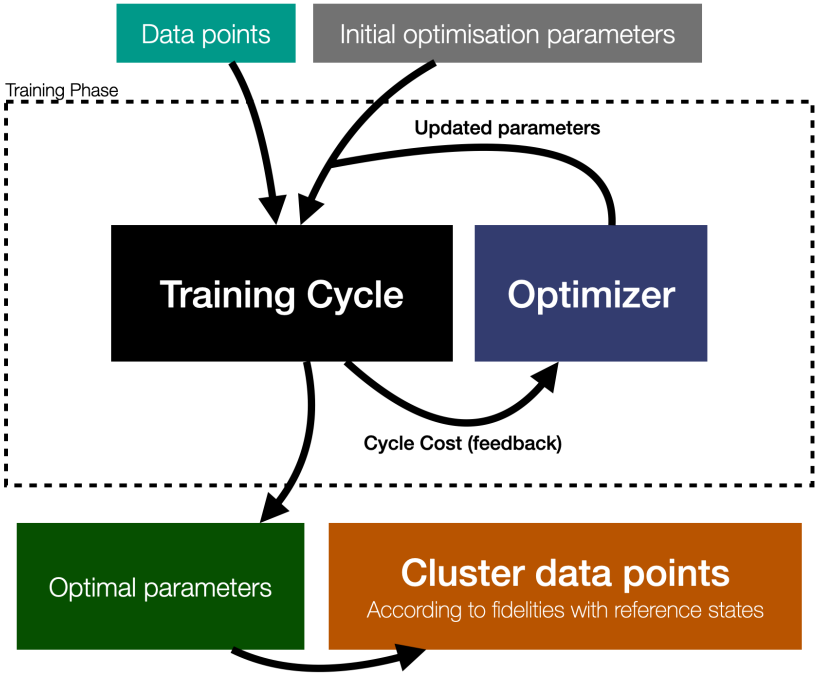

Figure 13: Algorithm overview

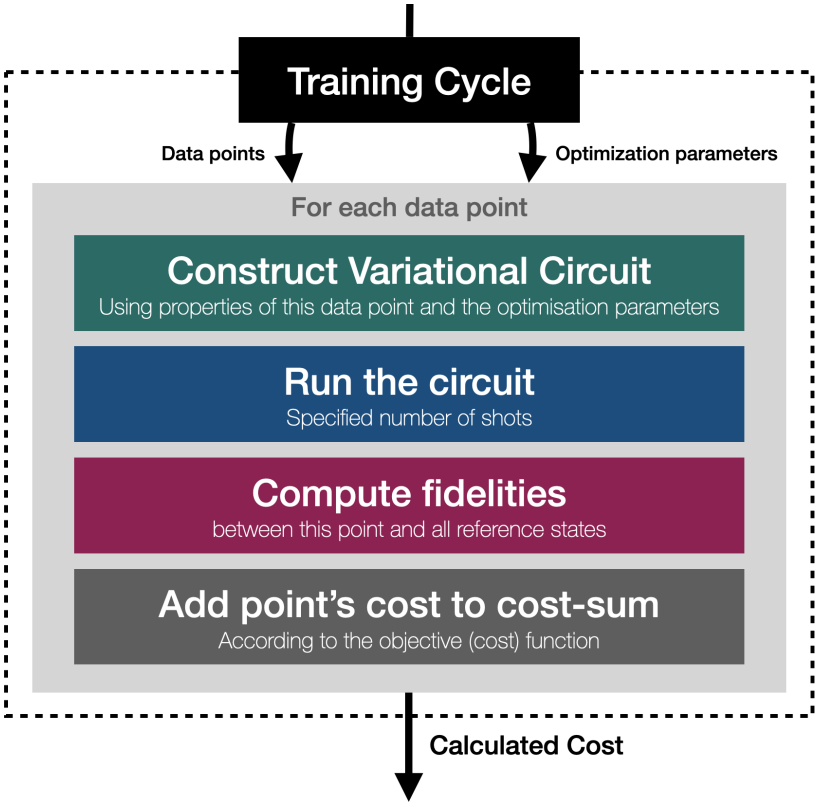

Figure 14: Training Cycle

8 Appendix C: Understanding the Algorithm

In order to fully understand the inner workings of the VQE algorithm and how it could be applied to perform unsupervised clustering, we started by reading several articles about the algorithm and one specifically about the combination of VQE and clustering [15]. We thought the article '*The Variational Quantum Eigensolver: A review of methods and best practices*' [7] was the best starting point on the route to understanding the algorithm. In this article, VQE is explained in the form of a pipeline that consists of the following parts:

1. Hamiltonian construction and representation
2. Measurement strategy and grouping
3. Parameter optimisation

8.1 Hamiltonian Construction and Representation

8.1.1 Construction

The fundamental basis of constructing a hamiltonian for our clustering problem relies on minimising the distance between data points within a cluster. Classically, this is done by finding the minimum sum of distances between two data points if they belong to the same class. This approach would provide the cost function H, such that

$$H = \frac{1}{2} \sum_{i,j=1}^{N} d(x_i, x_j) \sum_{a=1}^{K} q_i^a q_j^a$$

for N data points and K classes, and $q_i^a = 1$ if data point i belongs to class a and 0 otherwise. However, in the quantum method this approach is no longer applicable. We cannot simply determine with absolute certainty whether a data point belongs to a particular class or not, instead we must consider the fidelity of a data point with respect to a predefined quantum state.

8.1.2 Representation

Now that the construction of the hamiltonian has been discussed, we will look into its representation. As discussed in the previous subsection, we will need to find the fidelity of every data point with respect to some predefined states, which

act as our classes. The fidelity of a data point i with respect to a class a is given by, $f_i^a = |\langle \Psi_i | \Psi_a \rangle|^2$. Thus, our new quantum cost function, that we must minimise, is represented as

$$H = \frac{1}{2} \sum_{i,j=1}^{N} d(x_i, x_j) \sum_{a=1}^{K} f_i^a f_j^a$$

where, similar to the classical function, N is the number of data points and K is the number of classes.

8.2 Measurement strategy and grouping

Upon propagating the input state through the quantum circuit, a series of measurements needs to be performed to determine the expectation value of the ansatz. Such series of measurements is referred to as a Pauli string, where the string's characters encode the corresponding measurement basis. Naively, the number of repetitions required to accurately measure the expectation scales quadratically with the error rate. However, this can be optimised by employing techniques that can either jointly measure commuting groups of operators or exploiting the shared information across different Pauli strings, for instance the Hamiltonian partitioning technique. The notion of Hamiltonian partitioning strategy is that rotating the measurement basis can diagonalise a group of commutative Pauli strings.

One such a group is based on qubit-wise commutativity (QWC), wherein all operators of the Pauli string with the same index are either identical, or one of them is the identity operator. This technique has been extensively studied and typically tends to reduce the number of operators to be measured by a third.

Alternatively, it is also possible to group Pauli strings with even number of non-commutative indices. However, finding the unitary rotation is more complicated than in the previous case, as the measurement basis is more complex and involves entangled measurements. This means that deeper circuits are required to achieve more significant speedups.

8.3 Parameter Optimisation

In an earlier chapter, it was demonstrated that **finding an exact solution** to a VQE optimisation problem is NP-Hard and can therefore be intractable. "As such, efficient optimisation strategies that provide a well-approximated solution within an acceptable number of iterations are essential for any variational algorithms to be put into practice" [7]. At first, this seems no different than an optimisation problem in a classical setting, however additional challenges arise when optimising the expectation value of a variational quantum ansatz:

1. Noise in, for example, gates on NISQ devices will disturb the landscape of the function that is optimised for. This could lead to bad convergence towards an optimum.
2. The optimisation-cost depends greatly on the precision required for optimisation, because the precision of the measured value is determined by the number of sample-shots.
3. The barren plateau problem may cause the gradient to vanish quickly.

There exist different types of optimisers that can be installed as a means of approximate optimisation. Gradient-based searching strategies rely on information about the derivative of the objective-function in order to converge. Gradient-free searching strategies do not.