Quantum K-means Clustering

Rupak Chattarjee* and Abhijat Sarma†
Center for Quantum Science and Engineering
Department of Physics, Stevens Institute of Technology,
Castle Point on the Hudson, Hoboken, NJ 07030

Various machine learning classification algorithms, namely Support Vector Machines (SVMs), and K-means clustering, perform well on classifying wide ranges of datasets due to their versatility. However, as one increases the amount of data points and features, the computation time for training and using these statistical models grows intractably. Here, we attempt to use quantum analogues to these algorithms to lower the computational complexity of these models. We analyze a previously developed quantum SVM, as well as a quantum K-means clustering algorithm we developed, on trinary classification of a real dataset. We find that our quantum clustering algorithm can classify linear boundaries much faster than and at least as accurately as SVMs and classical clustering algorithms.

I. Introduction

Machine learning offers solutions to several classes of problems unreachable through conventional computing means. For example, solutions to classification problems and regression of large datasets based on machine learning techniques are in general much more powerful than previously available solutions. Here, we focus specifically on classification algorithms. So-called Support Vector Machines (SVMs) are one such algorithm, suited to binary classification problems with broad applications to many different kinds of datasets. It is a family of supervised learning algorithms, meaning that it must first be "trained" on a dataset with known classifications, based on finding linear boundaries to datasets which are not necessarily linearly separable. Their power stems from their reliance on kernels, a typically nonlinear distance function, interpreted as an inner product in a higher dimensional space. SVMs use kernels to implicitly map input vectors to these higher, not necessarily finite dimensional, so-called "feature spaces", in which a separating hyperplane can be found. This ability to find linear separation boundaries in feature space, regardless of the shape of the decision boundary in input space, allows SVMs to be very versatile in their application. Additionally, kernel functions can be calculated without explicitly mapping each input vector to its representation in feature space, a technique known as the "kernel trick", saving lots of computational time. Despite their power, an insurmountable issue arises with regards to these machines: namely, that computation time of kernels tends to scale exponentially with the size and dimension of the datasets used. This problem can be circumvented using quantum computing methods. Two approaches to this problem exist: namely, direct implementation of the Support Vector Machine on a variational quantum circuit which is then optimized, as detailed in [2, 3], or direct computation of kernels on a quantum circuit which is then used to classically compute the Support Vector Machine. Here, the latter approach is investigate. We implement and characterize the algorithm introduced in [1], and compare it to a Radial Basis Function (RBF) classical kernel.

Another algorithm we investigate is the K-means clustering algorithm. It is an unsupervised clustering algorithm in which the dataset is arranged into K clusters of similar datapoints. The algorithm relies on a distance measure, here taken to be Euclidean square distance. This distance can be calculated more efficiently on a quantum computer, as we will show, in order to speed up the algorithm as a whole. We compare our quantum K-means clustering algorithm to a classical one, as well as a quantum and classical multiclass extension of the SVM in solving trinary classification problems on a real dataset. We begin the paper with introductions of classical SVMs and K-means clustering algorithms. Skip sections 2 and 4 if you are already familiar with these concepts.

II. Support Vector Machine

In machine learning theory, it is often mathematically convenient to consider the data as encoded in a vector. Therefore, each data point with P different variables (features), can be encoded as a P-dimensional feature vector. The total dataset is therefore a set of vectors in P-dimensional space, known as input space. The support vector machine algorithm attempts to find a separating hyperplane - informally, an $M-1$ dimensional generalization of a line - in M-dimensional *feature space* (a higher dimensional vector space of nonlinearly transformed feature vectors) from which all of the data points in one class will lie on one side of the hyperplane, and all of the data points in the other class will lie on the other. Let us consider a dataset consisting of N different feature vectors $\mathbf{X}^i = (X_1^i, X_2^i, \ldots, X_P^i)$. The problem boils down to the convex optimization problem of finding the factors α^i

* Rupak.Chattarjee@stevens.edu
† absarma@ctemc.org

such that

$$f(\mathbf{X}) = \sum_{i=1}^{N} \alpha^i y^i K(\mathbf{X}^i, \mathbf{X}) + b \tag{II.1}$$

is a decision function acting on a datapoint $\mathbf{X}$ whose sign correctly classifies it - namely, positive values of f($\mathbf{X}$) correspond to one classification for $\mathbf{X}$, and negative values correspond to the other. See [8] for more information. $K(\mathbf{X}, \mathbf{X}^i)$ is a positive-definite *kernel* function, equal to an inner product in some high dimensional vector space.

$$K(\mathbf{X}, \mathbf{X}^i) = \sum_{j}^{\infty} \varphi_j(\mathbf{X}^i)\varphi(\mathbf{X}). \tag{II.2}$$

Here, φ is some non-linear transformation into higher dimensional vector space. The power of the SVM stems from the fact that the kernel function can be calculated without explicitly calculating φ. Different kernels give rise to different decision boundaries, but many kernels are classically intractable. However, some can be more efficiently calculated on a quantum computer, as will be explored in the next section.

III. The Quantum Support Vector Machine

In this section, we switch to the notation $\vec{x}$ instead of $\mathbf{X}^i$ to denote our feature vectors to avoid indexing confusion. To achieve calculation of a classically intractable kernel, we must encode the feature vectors $\vec{x}$ into quantum states that can be manipulated to compute our desired kernel. Here, we explore one such algorithm. Note that the algorithm presented here was developed by Havlicek et al. in [1]. We define the unitary gate

$$\mathbf{U}_{\Phi(\vec{x})} = \exp(i \sum_{S\subseteq[n]} \phi_S(\vec{x}) \prod_{i\in S} Z_i), \tag{III.1}$$

as well as the n-qubit gate

$$\mathbf{M}_{\Phi(\vec{x})} = \mathbf{U}_{\Phi(\vec{x})}\mathbf{H}^{\otimes n}\mathbf{U}_{\Phi(\vec{x})}\mathbf{H}^{\otimes n} \tag{III.2}$$

where $\mathbf{H}$ is the usual Hadamard gate. Here we take $n = 2$. The data is encoded through the coefficients $\phi_S(\vec{x})$, such that

$$\phi_1(\vec{x}) = x^i, \phi_{1,2}(\vec{x}) = (\pi - x^1)(\pi - x^2) \tag{III.3}$$

We define our kernel as follows

$$K(\vec{x}, \vec{z}) = |\langle \Phi(\vec{x})|\Phi(\vec{z})\rangle|^2 = |\langle 0^n|\mathbf{M}^{\dagger}_{\Phi(\vec{x})}\mathbf{M}_{\Phi(\vec{z})}|0^n\rangle|^2. \tag{III.4}$$

This kernel is thought to be classically intractable. However, it can be easily calculated by applying the gate $\mathbf{M}_{\Phi(\vec{x})}$ followed by the gate $\mathbf{M}^{\dagger}_{\Phi(\vec{z})}$ to an initial state $|0\rangle^n$and experimentally calculating the frequency of getting the zero strings 0^n as a result from the circuit.

IV. *K*-means Clustering

K-means clustering is an unsupervised learning algorithm which considers the problem of partitioning N feature vectors into K *subsets*, or *clusters*. The algorithm seeks to find the K clusters which minimize the dissimilarity between each cluster's members. Consider a dissimilarity metric between two features $D(\mathbf{X}^i, \mathbf{X}^j)$, which has the characteristic property that it increases in value as the features $\mathbf{X}^i, \mathbf{X}^j$ become more dissimilar. The measure used does not necessarily have to be a metric. Let $d_k(X_k^{(i)}, X_k^{(j)})$ be a dissimilarity measure between the k^{th} features of two observations $\mathbf{X}^i$ and $\mathbf{X}^j$. The canonical dissimilarity measure for clustering is then

$$D(\mathbf{X}^i, \mathbf{X}^j) = \sum_{p=1}^{P} w_p \cdot d_p(X_p^{(i)}, X_p^{(j)}), \sum_{p=1}^{P} w_p = 1 \tag{IV.1}$$

where w_p are weights for the different features. The standard choice for $d_p(X_p^{(i)}, X_p^{(j)})$ is the squared distance

$$d_p(X_p^{(i)}, X_p^{(j)}) = (X_p^{(i)} - X_p^{(j)})^2. \tag{IV.2}$$

For continuous, real feature variables, the canonical dissimilarity measure with $w_k = 1$ becomes the squared Euclidean distance,

$$D(\mathbf{X}^i, \mathbf{X}^j) = d_p(X_p^{(i)}, X_p^{(j)}) = |\mathbf{X}^i - \mathbf{X}^j|^2. \tag{IV.3}$$

The K-means clustering algorithm is initialized with some K clusters $C_1, C_2, ..., C_K$ Each observation $\mathbf{X}^i$ belongs to one and only one cluster. The goal is to create clusters of observations that are the least dissimilar. The canonical intra-cluster measurement of this dissimilarity is

$$E(C_k) = \frac{1}{|C_k|} \sum_{i,j\in C_k} \sum_{p=1}^{P} (X_p^{(i)} - X_p^{(j)})^2 \tag{IV.4}$$

Defining m_{kp} as the *mean* of feature p in cluster k,

$$m_{kp} = \frac{1}{|C_k|} \sum_{i\in C_k} X_p^{(i)}, \tag{IV.5}$$

one can express the *energy* term (5.4) as variance from the mean,

$$E(C_k) = 2 \sum_{i\in C_k} \sum_{p=1}^{P} (X_p^{(i)} - m_{kp})^2. \tag{IV.6}$$

In order to fulfill the least dissimilar criteria, one must minimize a cost function equal to the sum of each cluster's energy,

$$min_{\{C_1,C_2,...,C_K\}} \left\{ \sum_{k=1}^{K} E(C_k) \right\}. \tag{IV.7}$$

This minimization is most commonly achieved with *Lloyd's Algorithm*, as follows:

Lloyd's Algorithm

1. Randomly assign a cluster assignment $\{1, 2, ..., K\}$ to each of the observations $\mathbf{X}^i$.

2. For each cluster C_k, calculate the *vector of feature means* $(m_{k1}, m_{k2}, ..., m_{kP})$ called the cluster's *centroid*

3. Assign each observation $\mathbf{X}^i$ to the cluster C_k with the closest centroid according to the Euclidean metric (5.3)

4. Iterate between 2 and 3 until the assignments to clusters do not change.

The required number of iterations can be lowered by cleverly choosing the initial cluster assignments, such as by the K-means++ algorithm, but that will not be explored in this paper. Sampling and estimating Euclidean distances between post-processed vectors on a classical computer is known to be exponentially hard. For big data sets, the algorithm becomes slow as convergence relies on repeated calculations of this distance measure. In the next section, we propose and implement a quantum algorithm for calculating the Euclidean distance, which can be shown to be exponentially faster than the classical counterpart [4]. Note that although we use Euclidean distance in our clustering algorithm, one could use any distance measure. The Euclidean distance is desirable for our purposes because it utilizes linear decision boundaries between clusters. For non-linear boundaries, one could use a corresponding distance measure.

V. Quantum K-means Clustering

To develop our quantum algorithm for estimation of Euclidean distance, we first must introduce the *swap test*. Consider a state

$$|0\rangle \otimes |\psi\rangle \otimes |\varphi\rangle , \tag{V.1}$$

consisting of an ancillary qubit and two equal-qubit states which we wish to find the overlap of. Perform a Hadamard transformation $\mathbf{H}$ on the ancillary followed by a **FREDKIN** gate (also known as a *controlled swap*) on this state,

$$\begin{aligned} &\mathbf{FREDKIN}(\mathbf{H} \otimes \mathbf{I} \otimes \mathbf{I}) |0\rangle \otimes |\psi\rangle \otimes |\varphi\rangle \\ &= (|0\rangle\langle 0| \otimes \mathbf{I} \otimes \mathbf{I} + |1\rangle\langle 1| \mathbf{SWAP}\rangle) \frac{1}{\sqrt{2}} [|0\rangle \otimes |\psi\rangle \otimes |\varphi\rangle \\ &\quad + |1\rangle \otimes |\psi\rangle \otimes |\varphi\rangle] \\ &= \frac{1}{\sqrt{2}} [|0\rangle \otimes |\psi\rangle \otimes |\varphi\rangle + |1\rangle \otimes |\varphi\rangle \otimes |\psi\rangle] \end{aligned} \tag{V.2}$$

Applying the Hadamard transformation once more to the ancillary qubit gives

$$\begin{aligned} |\Psi\rangle &= (\mathbf{H} \otimes \mathbf{I} \otimes \mathbf{I}) \frac{1}{\sqrt{2}} [|0\rangle \otimes |\psi\rangle \otimes |\varphi\rangle \\ &\quad + |1\rangle \otimes |\varphi\rangle \otimes |\psi\rangle] \\ &= \frac{1}{2} [(|0\rangle + |1\rangle) \otimes |\psi\rangle \otimes |\varphi\rangle \\ &\quad + (|0\rangle - |1\rangle) \otimes |\varphi\rangle |\psi\rangle)] \\ &= \frac{1}{2} |0\rangle \otimes [|\psi\rangle \otimes |\varphi\rangle + |\varphi\rangle \otimes |\psi\rangle] \\ &\quad + \frac{1}{2} |1\rangle \otimes [|\psi\rangle \otimes |\varphi\rangle - |\varphi\rangle \otimes |\psi\rangle] \end{aligned} \tag{V.3}$$

Finally, measure the state of the ancillary qubit. The probability of measuring $|0\rangle$ is given by

$$\begin{aligned} &\langle\Psi|(|0\rangle\langle 0| \otimes \mathbf{I} \otimes \mathbf{I})|\Psi\rangle \\ &= \frac{1}{4} \{\langle\psi| \otimes \langle\varphi| + \langle\varphi| \otimes \langle\psi|\} [|\psi\rangle \otimes |\varphi\rangle + |\varphi\rangle \otimes |\psi\rangle] \\ &= \frac{1}{2} + \frac{1}{4} [(\langle\psi| \otimes \langle\varphi|)(|\varphi\rangle \otimes |\psi\rangle) \\ &\quad + (\langle\varphi| \otimes \langle\psi|)(|\psi\rangle \otimes |\varphi\rangle)] \\ &= \frac{1}{2} + \frac{1}{4} [\langle\psi|\varphi\rangle \langle\varphi|\psi\rangle + \langle\varphi|\psi\rangle \langle\psi|\varphi\rangle] \end{aligned} \tag{V.4}$$

Therefore, as first shown in [5],

$$P[|0\rangle] = \langle\Psi|(|0\rangle\langle 0| \otimes \mathbf{I} \otimes \mathbf{I})|\Psi\rangle = \frac{1}{2} + \frac{1}{2} |\langle\psi|\varphi\rangle|^2 \tag{V.5}$$

The swap test therefore allows us to experimentally determine the overlap between two states $|\psi\rangle$ and $|\varphi\rangle$. This will be integral in calculating the Euclidean distance. In order to calculate Euclidean distance, we must first encode our feature vectors into Hilbert Space. We do this by encoding the vectors into quantum states using the base-2 *bit string configuration* introduced in [6]. Recall that $|p\rangle = |p_{n-1}p_{n-2}...p_1p_0\rangle, p = 2^0p_0 + 2^1p_1 + ...2^{n-1}p_{n-1}, P = 2^n$. Note that if the dimension of the feature vector $\mathbf{X}^i$ is not a power of two, one must pad it with zeroes in order to make it such.

$$|\mathbf{X}^i\rangle = \frac{1}{|\mathbf{X}^i|} \sum_{p=1}^{P} X_p^{(i)} |p\rangle \tag{V.6}$$

Realize that the overlap between two of these states recovers the usual vector dot product, namely

$$\langle\mathbf{X}^i|\mathbf{X}^j\rangle = \frac{1}{|\mathbf{X}^i||\mathbf{X}^j|} \sum_{p=1}^{P} X_p^{(i)} X_p^{(j)} = \frac{1}{|\mathbf{X}^i||\mathbf{X}^j|} \mathbf{X}^i \cdot \mathbf{X}^j \tag{V.7}$$

Next, we construct the following states

$$\begin{aligned} |\psi\rangle &= \frac{1}{2} [|0\rangle \otimes |\mathbf{X}^i\rangle + |1\rangle \otimes |\mathbf{X}^j\rangle] \\ |\varphi\rangle &= \frac{1}{Z} [|\mathbf{X}^i| \, |0\rangle - |\mathbf{X}^j| \, |1\rangle] \end{aligned} \tag{V.8}$$

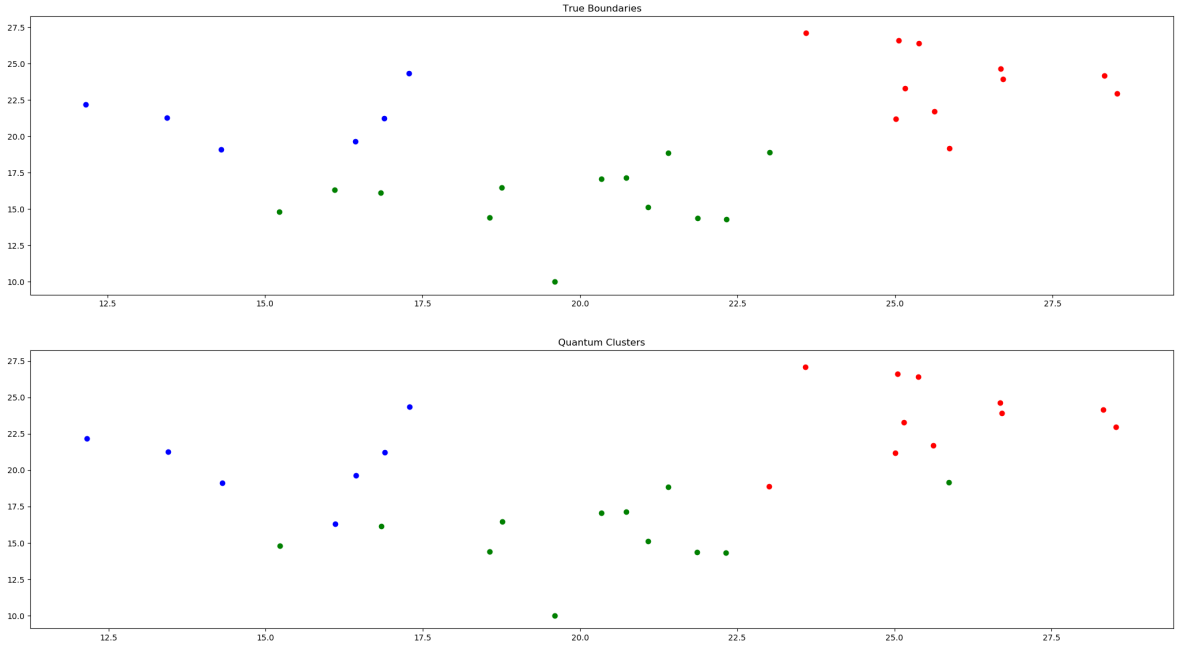

FIG. 1. True and Predicted Classifications by Quantum K-means algorithm

where $Z = |\mathbf{X}^i|^2 + |\mathbf{X}^j|^2$ is a normalization constant. Performing a swap test between the first qubit of $|\psi\rangle$ and the state $|\varphi\rangle$ will give the overlap $|\langle\psi|\varphi\rangle|^2$ of the two states. To calculate this overlap, we first calculate the partial overlaps $\langle\psi|\varphi\rangle$ and $\langle\varphi|\psi\rangle$, which reduce to

$$\begin{aligned}\langle\psi|\varphi\rangle &= \frac{1}{\sqrt{2Z}}[|\mathbf{X}^i|\langle\mathbf{X}^i| - |\mathbf{X}^j|\langle\mathbf{X}^j|] \\ \langle\varphi|\psi\rangle &= \frac{1}{\sqrt{2Z}}[|\mathbf{X}^i||\mathbf{X}^i\rangle - |\mathbf{X}^j||\mathbf{X}^j\rangle]\end{aligned} \tag{V.9}$$

Therefore, the complete overlap between these states $|\langle\psi|\varphi\rangle|^2 = \langle\psi|\varphi\rangle\langle\varphi|\psi\rangle$ is

$$\begin{aligned}|\langle\psi|\varphi\rangle|^2 &= \langle\psi|\varphi\rangle\langle\varphi|\psi\rangle \\ &= \frac{1}{2Z}\{|\mathbf{X}^i|^2 + |\mathbf{X}^j|^2 \\ -|\mathbf{X}^i||\mathbf{X}^j|\langle\mathbf{X}^i|\mathbf{X}^j\rangle &- |\mathbf{X}^j||\mathbf{X}^i|\langle\mathbf{X}^j|\mathbf{X}^i\rangle\} \\ &= \frac{1}{2Z}\left\{|\mathbf{X}^i|^2 + |\mathbf{X}^j|^2 - 2\mathbf{X}^i\cdot\mathbf{X}^j\right\} \\ &= \frac{1}{2Z}\left\{|\mathbf{X}^i - \mathbf{X}^j|^2\right\}\end{aligned} \tag{V.10}$$

The classical Euclidean distance is therefore proportional to the overlap of the states specified in (6.9). Following [4], the quantum algorithm for calculating the Euclidean distance is to perform a swap test between those two states such that

$$\begin{aligned}P[|0\rangle] &= \frac{1}{2} + \frac{1}{2}|\langle\psi|\varphi\rangle|^2 \\ &= \frac{1}{2} + \frac{1}{4Z}|\mathbf{X}^i - \mathbf{X}^j|^2\end{aligned} \tag{V.11}$$

or

$$|\mathbf{X}^i - \mathbf{X}^j|^2 = Z(4P[|0\rangle] - 2). \tag{V.12}$$

We implement this experimental calculation of the Euclidean distance between arbitrary-dimensional feature vectors in our clustering algorithm on Qiskit in order to achieve a speed-up over similar classical algorithms.

Several limitations exist on experimental applications of this algorithm for quantum distance measure. Firstly, there is large variance on the probability of getting 0 as a measurement result, causing the measure to be unreliable for a low number of shots. We deal with this by simulating each distance circuit 40000 times in order to determine the probability. However, we suspect that this large variance is a consequence of the simulation software provided by Qiskit, and will be less of a factor on real quantum systems as noise mitigation techniques are improved. Another issue which arises is that negative numbers are generally treated as if they were positive by the distance measure, giving wildly incorrect answers. We conjecture that this issue stems from the way that phase differences are handled by Qiskit. In order to fix

	Accuracy	Time (s)		Accuracy	Time (s)
Classical SVM	96.7%	0.0109	Quantum SVM by Havlicek et al.	63.3%	103.9
	96.7%	0.0031		63.3%	105.4
	96.7%	0.0058		63.3%	106.5
	96.7%	0.0029		63.3%	103.4
	96.7%	0.0034		63.3%	104.2
Average	96.7%	0.0052	Average	63.3%	104.7
Classical K-means	70.0%	0.0183	Quantum K-means	100.0%	88.63
	93.3%	0.0139		93.3%	134.1
	90.0%	0.0185		100.0%	131.3
	90.0%	0.0141		93.3%	109.7
	100.0%	0.0204		96.7%	110.9
Average	88.7%	0.0170	Average	96.7%	114.9

TABLE I. Trinary Classification on Wine Dataset

this issue, we must translate all of the data in preprocessing in order to put it all in the first quadrant. Third, differences in distance on the order of 10^{-1} generally give experimental estimates with very large error. This can be handled by upscaling the data in preprocessing. Finally, in general, the algorithm tends to misclassify (assign to a wrong cluster) between 0-10% of the data per iteration, which can sometimes cause failure of the algorithm to converge, especially for large datasets. Manual switches can be put in to the algorithm to force it to terminate, or else one can declare the algorithm to have converged if some number of features less than a threshold value change clusters, as opposed to the generally used value of zero.

VI. Trinary Classification

While K-means clustering and the support vector machine belong to different classes of machine learning algorithms, with the former being a clustering algorithm and the latter being a supervised binary classification algorithm, we can compare them under certain constraints. We consider a trinary classification problem on the standard Wine dataset provided by Scikit. For the K-means clustering algorithm, we input the test data and then compare the generated clusters with the true labels of the data. In order to determine which cluster corresponds to which label, we calculate the centroid of each true class, and assign the label of that class to the generated cluster with the closest centroid. Then, we calculate the accuracy by calculating the percentage of correctly classified features. For the Support Vector Machine, we utilize the *One Against Rest* multiclass extension provided by Qiskit to extend the SVM to more than two classes. The *One Against Rest* extension constructs a number of SVMs equal to the number of classes, each of which compare one class against all the others. See [7] for more details. We include a classical K-means clustering algorithm as well as a classical RBF kernel SVM as controls. We run five experiments on each classifier, with 2 feature dimensions and 30 test inputs, as well as 30 training inputs for the SVMs. Note that the clustering algorithms are not provided with any training vectors and corresponding labels, while the SVMs are. All simulations were run with Qiskit Aer on a Macbook Pro. Results of the simulations are shown in Table 1. Figure 1 graphically represents one run of the Quantum K-means algorithm.

As expected, the SVM algorithms are much more consistent in terms of time and accuracy than the clustering methods. This is because the SVM is a supervised algorithm, and the amount of time that the clustering algorithm takes depends on how many iterations it must undergo, which consequently depends on the random initial seeding of the clusters. Surprisingly, the quantum K-means clustering algorithm is actually significantly more accurate than its classical counterpart, although that could be attributed to an outlier in the data of the classical algorithm. The quantum algorithms take 5 and 4 orders of magnitude more time respectively to execute than their classical counterparts, but that is a consequence of simulating the quantum circuits as opposed to running them on a real machine.

VII. Bibliography

[1] Havlicek, V., Corcoles, A., Temme, K., Harrow, A., Kandala, A., Chow, J. & Gambetta, Jay. Supervised learning with quantum enhanced feature spaces. *Nature* **567**, 209–212 (2019).

[2] Mitarai, K., Negoro, M., Kitagawa, M. & Fujii, K. Quantum circuit learning. *arXiv preprint arXiv:1803.00745* (2018).

[3] Farhi, E. & Neven, H. Classification with quantum neural networks on near term processors. *arXiv preprint arXiv:1802.06002* (2018).

[4] Lloyd, S., Mohseni, M., & Rebentrost, P. Quantum algorithms for supervised and unsupervised machine learning. *arXiv:1307.0411 [quant-ph]* (2013).

[5] Aïmeur, E., Brassard, G., & Gambs, S. Machine learning in a quantum world. *Advances in artificial intelligence*, 431-442. Springer (2006).

[6] Lloyd, S., Mohseni, M., & Rebentrost, P. Quantum support vector machine for big data classification. *Phys. Rev. Lett.* **113**, 130503 (2014).

[7] Multiclass Extension. (n.d.).
Retrieved from https://qiskit.org/documentation/aqua/multiclass_extensions.html

[8] Smola, A.J. & Schölkopf, B. Statistics and Computing (2004) **14**: 199. https://doi.org/10.1023/B:STCO.0000035301.49549.88