Image Clustering using Q-Means

Muhilan M || EP19B005

30 Nov 2022

K-Means is a widely used classical ML algorithm that groups similar datapoints together in a dataset and thereby identifying a single representative datapoint(mean) for each group. One of the main reasons for its widespread use as a clustering algorithm is due to the ease of implementation. However, the algorithm's major downfall is its time complexity, which scales as $O(nkd)$, where n : number of datapoints, k : number of clusters and d : dimension of the datapoint. With this in consideration, we do a comparative study between the K-means algorithm and its quantum counterpart, "Q-means"[2][4] to analyse their performance and efficiency. "Q-means", a quantum circuit based algorithm offers a a logarithmic speedup under a suitable quantum encoding of a given dataset.

I Introduction

Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in terms of some metric) to each other than to those in other groups (clusters). It belongs to the paradigm of unsupervised machine learning, where the primary aim is to derive some "inference" or "comprehend" the given non-labelled (no pre-assigned categories or classes) dataset such that any downstream tasks that follow using the dataset is less cumbersome and more efficient. K-means is one such cluster analysis technique or a clustering algorithm that is centroid or mean-based and uses the Euclidean distance as the similarity measure between the datapoints in the dataset.

Although K-means is usually carried out over a non-labelled dataset(dataset with no pre-assigned classes/categories), here we are interested in comparing two algorithms that carry out the same task of grouping data into clusters and so we make use of a labelled dataset so that it is easier to compare and contrast the performance and the resultant clusters obtained via K-means and the Q-means algorithm. So, in addition to using just the "objective function" that the algorithms try to minimize as the metric to evaluate the "goodness" of the clusters, we also use the labels associated with each datapoint in the labelled dataset.

The logarithmic speedup that the Q-means offers is related to the way in which it handles the dimensions of the datapoints(d) and thus the quantum speedup is more evident while handling high-dimensional data, such as images and audio signals. Therefore, for the comparative study of the algorithms, we have chosen to implement the algorithms over an image dataset and analyse their performance.

The report is organised as follows. Sec II talks about the dataset chosen for the project. Sec III discusses the data preprocessing steps that needs to be done before the implementation of algorithms. The theory and the pseudo code of K-means is discussed in Sec IV, followed by an introduction to the quantum Swap-Test circuit in Sec V, which forms the basis for the Q-Means circuit. Sec VI discusses the Q-means algorithm, its pseudo-code and the quantum circuit employed by the algorithm. Finally, we take a look at the results obtained from both the classical and the quantum algorithm, in Sec VII which is followed by concluding remarks in Sec VIII.

II Dataset

The researchers of Qatar University have compiled the "COVID-QU-Ex dataset"[1][5], which consists of 33,920(recently updated) human chest X-ray images including: 11,956 COVID-19 cases,11,263 Non-COVID infections (Viral or Bacterial Pneumonia) and 10,701 Normal X-rays, which is also available in the "Kaggle" data repository[7]. At the time of this project, the dataset was comprised of:

- 2000 covid infected lung x-ray images(Type 1)
- 2000 normal lung x-ray images(Type 2)
- 2000 lung opacity images(Type 3)
- 1345 viral pneumonia affected x-ray images(Type 4)

The key aspect of this dataset is that along with the lung x-ray images, the corresponding masks for the x-ray images are also available. A mask is a binary image consisting of zero-(black pixel) and non-zero values(usually only 255-white pixel). If a mask is applied to another mage of the same size, all pixels which are zero in the mask are set to zero in the output image. All other pixels remain unchanged. An example image and its corresponding mask image is shown below.

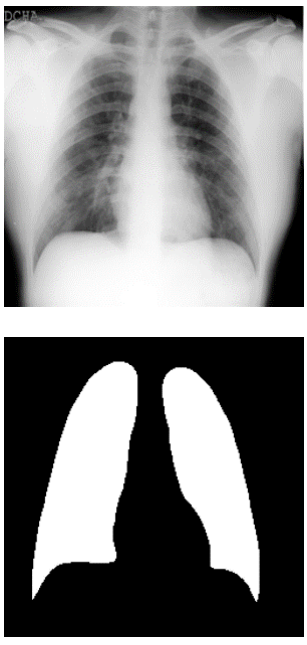


Fig.1: A normal lung X-ray image along with its mask from the dataset7

From the figure, it is evident that the masks can help in eliminating the irrelevant regions of the X-ray and retain only the lung portion in the X-ray images.

The dataset images(both lung X-ray and the masks) are 256x256 RGB channel images, where the pixel values can range from 0 to 255.

In the next section, we look at how we process these raw images into vectors belong to the real vector space.

III Data Pre-processing

The first pre-processing step is to apply the mask to its corresponding lung image so that the irrelevant regions are blacked-out. This is achieved by doing a simple logical AND between the x-ray image(pixel matrix 1) and its corresponding mask(the pixel matrix 2). The following image shows the output after the "masking" or the logical AND operation between the data images in Fig.1.

Fig.2: Output of "masking"

The "masking" operation is followed by resizing the resulting image to a size of 224x224, which is suitable for the other pre-processing steps that follow.There are two key pre-processing procedures that are further done on this masked and resized output image: feature extraction using a nueral network and dimensionality reduction.

III.I Feature extraction using VGG-16 neural network

The 224x224 RGB masked image corresponds to $224 \times 224 \times 3 = 150528$ pixel values each ranging from 0 to 255.Directly using the pixel values as the datapoint could be cumbersome as it belongs to R^{150528} space. Therefore, it becomes important to extract the relevant "features" from each image, which for the lung images could be the "cloudy regions","rib-cage lines" etc, as these are the most relevant, when dealing with similarities and dissimilarities between the images which essentially is the basis for the grouping or clustering.

Here, the "features" of the images are extracted using VGG16 Neural Network image classification architecture, specifically trained over the given labelled dataset. VGG16 NN architecture was proposed by Karen Simonyan and Andrew Zisserman of the Visual Geometry Group Lab of Oxford University in 2014 in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition"[6]. This model won 1st and 2nd place in the "object localisation" and "image classification" categories respectively in the 2014 ImageNet Large Scale Visual Recognition Challenge(ILSVRC) challenge. It is considered to be one of the excellent computer vision model architecture till date.

The VGG-16 is a convolutional neural network(CNN) that is 16 layers deep and is comprised of the following layers:

- Input layer: 224x224x3
- Convolution layers: Trainable filters
- Max-Pooling layers
- Dense/Fully connected layer

Given below is the schematic of the VGG16 NN architecture.

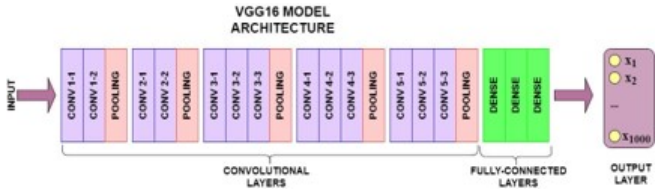


Fig.3: Schematics of the VGG16 Neural Network[8]

The input layer represents the 224x224 RGB image input to the network. Each convolutional layer essentially consists of multiple trainable(in ML sense) filters that select or activate certain features in the image and as we go deep into the network, these features get more complicated. The max-pooling layers on the other hand are non-trainable and is used for making the model translationally invariant, meaning images that are shifted would not be considered as different or unique. The fully-connected or dense layers do the job of a function that maps the output vector from the previous layers to one of the classes/labels.

The original VGG16 architecture was used for 1000 classes(labels/types) classification problem but for our case the data belongs to 4 classes: namely covid(type 1),normal(type 2), lung opacity(type 3) & viral pneumonia(tye 4), therefore we replace the final dense layer such that the output layer is of size 4 or in other words, the model is modified for a 4-class classification problem as required. For the purpose of training the NN, the dataset was split randomly into 70:30 training and validation sets, with training done using the 70% split and testing(for classification accuracy) on the 30% split.Since the VGG16 is a very deep CNN with 134,276,932 trainable parameters, the training was done using Google Colab via a cloud-accessible GPU hardware.

Training set accuracy(4-label classification problem) of 100% validation(test) set accuracy of 81.55& was achieved during the training. With the training done, now the model is ready for feature extraction. The fully connected layers(which is function that maps the output vector from previous layers to the classes) are removed and the masked images in the dataset are passed through the remaining layers one by one, which results in a 25,088 column vectors(usually referred to as "feature vectors") each corresponding to one of the masked images.

Thus, starting from raw masked images, we have extracted the features resulting in a feature vector for each of $N = 7345$ images belonging to real vector space of dimension, $D = 25,088$

III.II Dimensionality reduction: PCA

For doing the comparison between the performance of K-means & Q-means, we can restrict ourselves to applying the algorithms on a uniformly sampled set of feature vectors from the dataset obtained from the previous feature extraction step. This is done mainly for computational convenience.

So, from $N = 7345, D = 25,088$ feature vectors/datapoints, we uniformly sample 100 datapoints from each class/type(covid,normal,lung opacity. Thus, the sampled dataset contains $N = 400$ datapoints belonging to $R^{25,088}$ space. The nature of this sampled dataset with D much greater than N, provided an opportunity to invoke another unsupervised learning algorithm that could reduce the dimensions of the datapoints, making all the downstream processes efficient.

Since the sampled dataset contains only 400 datapoints, the number of linearly independent vectors can be atmost 400, meaning that the datapoints lie in a low dimensional subspace of $R^{25,088}$. PCA or Principal component Analysis essentially outputs the set of orthogonal vectors that define the low-dimensional subspace in which the datapoints reside and the projections of the datapoints onto this subspace would become the new datapoints(they act as very good proxies for original ones).

Therefore, a linear PCA(non-kernelised version) was applied onto the sampled dataset and the dimensions of the feature vectors were reduced from $25,088$ to 399, such that the subspace obtained from PCA captured all the variance(Variance captured=100%) in the data. In other words, no information was lost during this dimensionality reduction process.

With all the pre-processing steps completed, we are left with $N = 400$ datapoints with 100 from each class, belonging to real vector-space of dimensions, $D = 399$.

IV K-Means

K-Means is a centroid or mean based clustering technique that groups datapoints such that the points in a given group or cluster is closer to mean $(\frac{1}{L}\sum_i x_i)$ of the cluster to which it belongs to than the other cluster means. So, the aim of the algorithm is to assign the cluster indicators(z_i) to each datapoint. Though they are many variations in the implementations of k-means, here we look at the simple iterative algorithm as proposed by Stuart P. Lloyd(commonly referred to as "Lloyd's K-Means algorithm").

There are 3 main characteristics to this algorithm:

i) The hyper-parameter k:

The "K" in K-means is the parameter that defines the number of clusters into which we want to group or divide the datapoints. It is a hyperparameter that is usually set following "hyper-parameter tuning"(sort of hit-and-trial method) or based on the domain knowledge. In our case, we indeed have the domain knowledge that the dataset contains points belonging to 4 types or closes and thus $k = 4$ is a natural choice for our clustering problem

ii) The similarity measure/metric:

K-means groups the datapoints based on the Euclidean distance between the datapoint and the cluster means, which implies that the similarity of points belonging to the same cluster are based on the "Euclidean distance" between them.

iii) The Objective function:

The primary aim of the K-Means is to minimize the objective function which is given as follows:

$$\sum_{i=1}^{N} |x_i - \hat{x}_{z_i}|^2 \tag{1}$$

, where N is the number of datapoints, z_i is the cluster indicator corresponding to each datapoint x_i, that can take values from $\{1, 2, 3, ...k\}$. $\hat{x}_{z_i}$ is the cluster mean of the z_ith cluster to which the point x_i belongs to.

So, the objective function is the sum of squares of the Euclidean distance between the datapoints and their corresponding cluster means and the minimization of this function is over the cluster indicators z_is, that is, finding an assignment of points to the clusters such that objective function is minimised.

Given below is the pseudo-code for the Lloyd's K-means algorithm[2]:

Pseudo code:

1. Choose ***k*** as the number of clusters.
2. Randomly choose ***k*** datapoints as centroids/cluster means
3. Repeat
4. For each data point do
5. Find the **Euclidean distance** to the *k* cluster means
6. Assign the datapoint to the cluster which is at a **minimum distance**(set *z*)
7. End of For
8. Recompute the **cluster means**
9. Until **convergence**

Fig.4: Pseudo code of the iterative Lloyd's algorithm

Convergence, here refers to the state where no more reassignment of datapoints takes place or in other words no changes in $\{z_i\}\forall i$ are observed. Iteration is stopped once the convergence is observed. It is possible to show analytically that at each step of the iteration, the objective function strictly decreases. This result together with the finite-ness of the number of possible combinations of z_i (k^N) guarantees the convergence of the Lloyd's algorithm.

However, there is no guarantee that the algorithm converges to the global minima of the objective function. It usually converges to one of the local minimum subject to the random initialisation.

For our dataset, we apply the domain knowledge that the data belongs to 4 different classes and set k=4 but also choose the 4 points each belonging to different class

and are far apart from each other, as the initial cluster means as we expect the most natural clusters to be such that all images from same class are put in same groups.

V Q-Means Preliminary: Swap Test Circuit

As mentioned earlier the quantum swap-test circuit forms the basis for the Q-means circuit and algorithm, therefore making its discussion a necessary one.

The quantum swap-test [3][2] is used to measure the amplitude squared of the inner product between two quantum states(two vectors encoded as quantum states). Given below is an example Quantum circuit that seeks to estimate the amplitude of the inner product between the states $|\psi_1\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ and $|\psi_2\rangle = \frac{\sqrt{3}}{2}|0\rangle + \frac{1}{2}|1\rangle$. (Basically, it estimates the dot product between the two dimensional vectors $\left[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right]^T$ and $\left[\frac{\sqrt{3}}{2}, \frac{1}{2}\right]^T$.

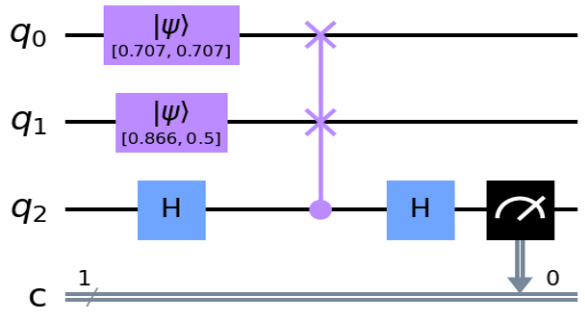


Fig.5: Example circuit: Swap Test [2]

The states $|\psi_1\rangle$ and $|\psi_2\rangle$ are initialed in qubits q0 and q1 respectively and the auxiliary qubit q2 is in $|0\rangle$ state. The Controlled swap gate with q2 as target sandwiched between the Hadamard gates on q2, followed by measurement of the auxiliary qubit on Z-basis, forms the Swap-Test Circuit.

It can can be shown algebraically that the probability of obtaining a measurement outcome of "0" is related to the required inner product as follows[2]:

$$\begin{aligned} |\langle\psi_1|\psi_2\rangle|^2 &= 2Pr(0) - 1 \\ &= 1 - 2Pr(1) \end{aligned} \tag{2}$$

where $Pr(0)$ where Pr(0) and Pr(1) are the probabilities of getting the measurement outcome "0" and "1" respectively. To get an estimation of the amplitude squared of the inner product within an additive constant "ϵ", the number of measurement shots required is of the order, $O(\frac{1}{\epsilon^2})$.

VI Quantum K-means: Q-means

Q-means[2][4]is identical to the Lloyd's algorithm except for a quantum Euclidean distance evaluation subroutine as opposed to the classical computation. Euclidean distance evaluation $|x - \hat{x}|$ in Q-means is derived from the "Swap-Test" quantum circuit and the encoding used for embedding the datapoints as quantum states is based on "amplitude encoding".

"Amplitude encoding" encodes each coefficient/feature of the datapoint as the amplitudes or coefficients of computational basis states of suitable size.So, datapoint or a feature vector, $(x_1, x_2, ..., x_D)$, is encoded into a quantum state as:

$$|\psi\rangle = \frac{1}{\sqrt{d}} \sum_{i=1}^{D} x_i |i\rangle \tag{3}$$

where $|i\rangle$ is the ith computational basis state. From the encoding, it is evident that in order to encode a datapoint of "D" dimensions, we require $\lceil log_2 D \rceil$ qubits. If $D < 2^{\lceil log_2 D \rceil}$, then the datapoint is padded with zeros to make $D = 2^{\lceil log_2 D \rceil}$, the encoded state is however unchanged.

Given below is the Pseudo-code for the Q-means algorithm[2]:

Pseudo code:

1. Choose ***k*** as the number of clusters.
2. Randomly choose ***k*** datapoints as centroids/cluster means
3. Repeat
4. Declare a empty list
5. For each **data point** do
6. For each **cluster mean** do
7. Construct the **quantum circuits** and append to the list
8. Use the **JobManager()** to send the list of circuits
9. For each datapoint do
10. From the job results, compute **distances and reassign** the point.
11. End For
12. Recompute the **cluster means**
13. Until **convergence**

Fig.6: Pseudo code for Q-means[10]

The important step of the algorithm is the construction of the circuit for evaluating the distance between a datapoint and the mean of the cluster to which it is currently assigned to. The circuit for this evaluation is given below[2]:

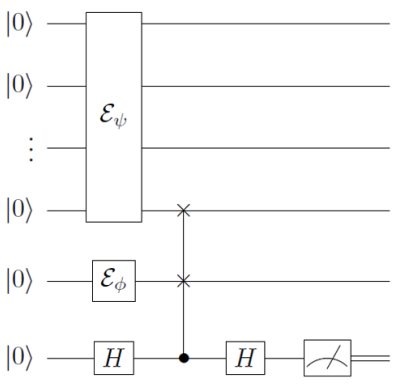


Fig.7: Pseudo code for Q-means

Consider a datapoint $a = (a_1, a_2, ..., a_d)^T$ in the dataset and the mean of the cluster to which a belongs

to, $b = (b_1, b_2, ..., b_d)$, then the modified amplitude encoding used in the circuit shown above are as follows[2]:

$$\begin{aligned} |\psi\rangle &= \frac{1}{\sqrt{2}}(|a\rangle|0\rangle + |b\rangle|1\rangle) \\ |\phi\rangle &= \frac{1}{\sqrt{Z}}(|a||0\rangle + |b||1\rangle) \\ where Z &= |a|^2 + |b|^2 \end{aligned} \tag{4}$$

$|a\rangle$ and $|b\rangle$ are the normalized, amplitude encoded quantum states corresponding to the datapoint a and the cluster mean b. $|0\rangle$ and $|1\rangle$ are the conventional single qubit computational basis states. So, for encoding the ψ state corresponding to "d" dimensional dataset, we would require $\lceil \log_2 d \rceil + 1$ qubits. The state ϕ is a single qubit state and another qubit initialized to $|0\rangle$, is the auxiliary qubit using which the swap-test is carried out. It can be shown algebraically that that the square of the distance between vector a and b is given as[2]:

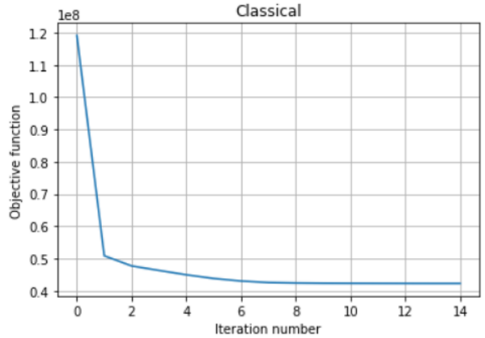
$$|a - b|^2 = 4Z(Pr(0) - \frac{1}{2}) \tag{5}$$

Notice how only a single Cswap gate is utilised in the swap-test part of the circuit and this is due to the extra qubit used while encoding the state ψ. Increasing the number of measurement shots increases the accuracy of the euclidean distance evaluated and the initialisation of states $|psi\rangle$ and $|phi\rangle$ are implemented using Qiskit's circuit.initialize()[9] function.

So, for encoding our preprocessed dataset whose $N = 100, D = 399$, we require $9+1$ for encoding ψ and 2 more qubits for state ϕ and the auxiliary state. Therefore, the implementation of the Q-means was run on "ibmq qasm simulator". Also, initialisation of the means was not random but identical to the initialisations used in K-means(as discussed in Sec IV as we are interested in comparing the performance of the algorithms under identical conditions.

VII Results

K-means (with k=4) was implemented and run on a classical computer and Q-means (with k=4) was run on the Qasm simulator, over the preprocessed $N = 400, D = 399$ lung image dataset. The objective function v/s the iteration number was plotted for both the algorithms, implemented on the above mentioned dataset.




```
Convergence: 42307669.79
```

Fig.8: Pseudo code for Q-means

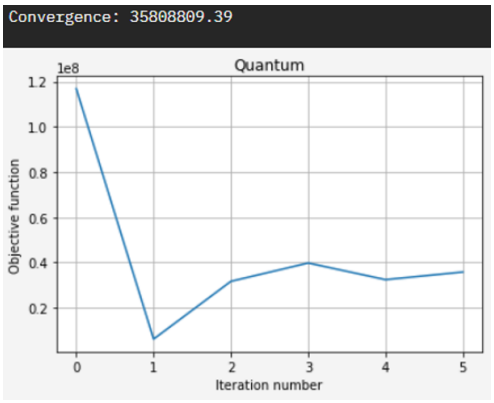


Fig.9: Pseudo code for Q-means

It was observed that the K-means behaved consistently with the theory that the objective function strictly decreases for every iteration and convergence was achieved on an average(mean initialisation is random) in 20 to 25 iterations. Q-means on the other hand, converged much quickly, in about 5 to 10 iterations but the objective function did not strictly decrease with the iteration number, indicating that the distance estimation was not as accurate as compared to the classical one(limitation on the number of measurement shots on the simulator).

However, Q-means plot showed a interesting behaviour, an immediate dip followed by rise and then the eventual convergence, which indicates that the algorithm seems to have escaped one local minima and converged to another one with a lower objective function value(compared to K-means), which again could be attributed to the lower accuracy of the distance evaluation.

The result and the inference that the Q-means converged to a lower objective value and therefore has better performance than K-means is misleading as the plot also revealed the inaccuracies in the important step of distance evaluation. Therefore, it is necessary to also look at the nature of the groups/clusters obtained from the algorithms.

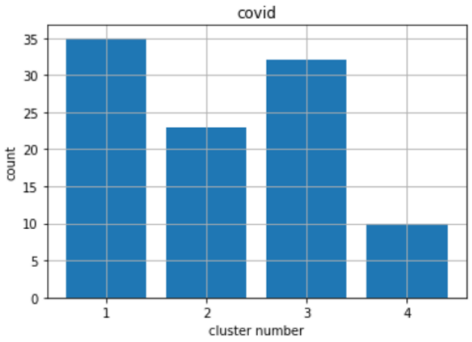


Fig.10: K-Means: Clustering of "covid" type datapoints

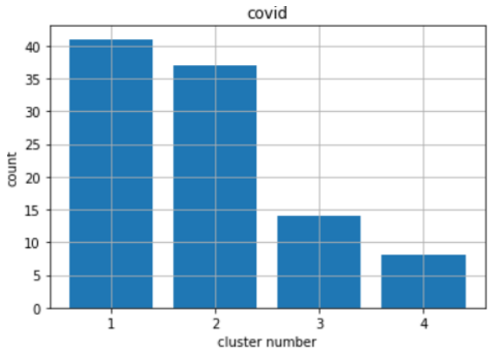


Fig.11: Q-Means: Clustering of "covid" type datapoints

We would expect based on our domain knowledge and the initialisation chosen that for k=4, datapoints of same type to be assigned to the same cluster.From the plots shown above, it is clear that both K-means and Q-means fails to give this results as the 100 datapoints of covid type are assigned to each of the four clusters. The failure of K-means to give the desired result could be attributed to the inherent drawback in the algorithm to converge to a local minimum.

However, we could still comment on the relative performance of the algorithms by considering the clustering for the covid type datapoints shown above and the clustering for the other types, which are shown in the figures below.

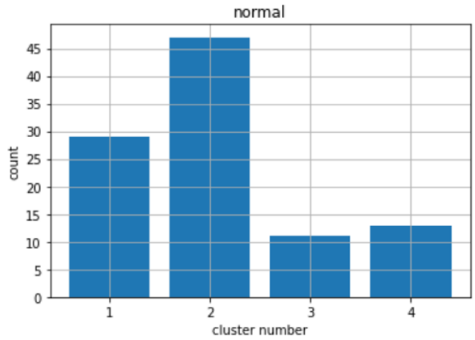


Fig.12: K-Means: Clustering of "normal" type datapoints

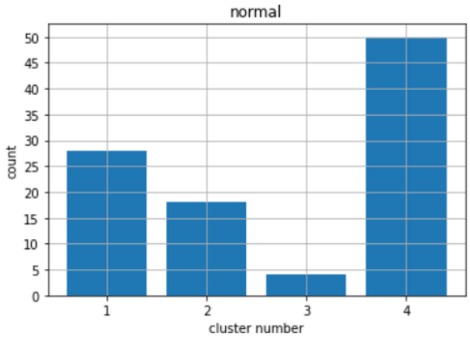


Fig.13: Q-Means: Clustering of "normal" type datapoints

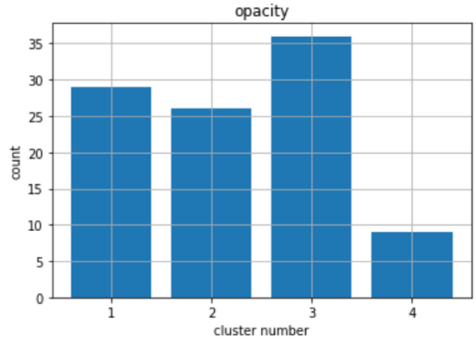


Fig.14: K-Means: Clustering of "lung opacity" type datapoints

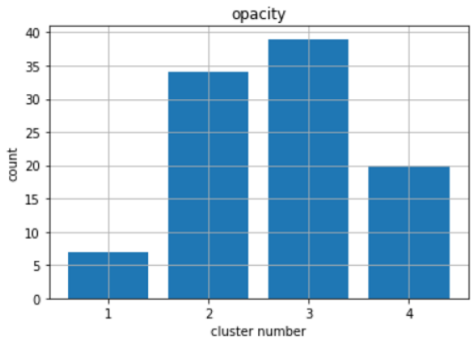


Fig.15: Q-Means: Clustering of "lung opacity" type datapoints

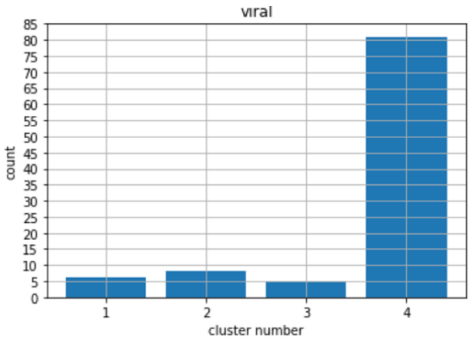


Fig.16: K-Means: Clustering of "viral pneumonia" type datapoints

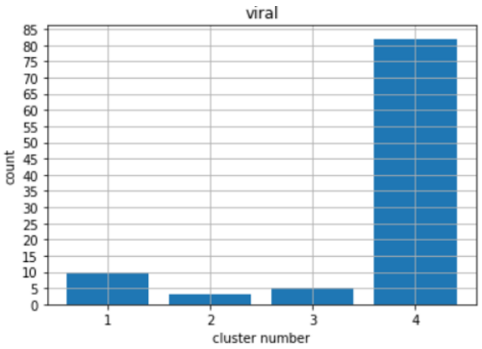


Fig.17: Q-Means: Clustering of "viral pneumonia" type datapoints

One important observation that we make from the plots shown is that, K-means has produced clusters such in each of the four clusters, the maximum number of datapoints are from different type, for example in the plots shown above, "covid" type datapoints predominately are in cluster 1, "normal" type datapoints predominately are in cluster 2, "lung opacity" type datapoints predominately are in cluster 3 and "viral" type datapoints are mostly grouped into cluster. On the other hand, Q-means,on an average, puts each of the 4 types of datapoints mostly in 2 of the 4 clusters(In the plots, shown above, it is cluster 3 & 4)

Therefore, we can make an inference that the clusters provided by K-means are more closer to the "natural" or the expected grouping and is consistent with the result that the objective function strictly decreases with iteration number.Thus, K-menas has better performance than Q-means for the dataset considered, even though the converged objective function value is higher for K-means.

VIII Conclusion

The running time complexity of K-means is $O(Nkd)$, which can be explained by the evaluation of distance between each of the N datapoints and k cluster means, which itself is of the order $O(d)$, where d is the dimension of the datapoints. The running time complexity of Q-means is $O(Nd + kd + \frac{Nk \log_2 d}{\epsilon^2})$ where the first 2 terms are less dominant and correspond to the preparation/encoding of states for the datapoint and the cluster mean respectively, while the 3rd and the dominant term corresponds to the running time of the euclidean distance evaluation circuit. The logarithmic speedup in Q-means is with respect to the dimension "d" of the datapoint and the speedup is attributed to the smart "amplitude encoding" of datapoints into quantum states.

With the results suggesting that Q-means has a poor performance on the dataset which is primarily related to the inaccuracies in distance evaluation as we are limited by the number of measurement shots that can be done right now, this accuracy factor also affects the efficiency or the time complexity of Q-Means. The dominant term $\frac{Nk \log_2 d}{\epsilon^2}$, where ϵ is the additive constant upto which the distance is estimated, suggests that once we have the ability to perform greater number of measurement shots, ϵ decreases and the performance of Q-means would improve.

Even though this would result in increased running time, once the ϵ is fixed to a low enough value(by increasing the measurement shots) , the running time essentially scales logarithmic in "d" compared to linear in "d" for K-means. This implies that Q-means would not only have a performance equivalent to K-means but it would be more efficient especially for huge and high dimensional datasets which are increasingly becoming commonplace in the ML context.

References

[1] Muhammad E. H. Chowdhury et al. "Can AI Help in Screening Viral and COVID-19 Pneumonia?" In: *IEEE Access* 8 (2020), pp. 132665–132676. DOI: 10.1109/ACCESS.2020.3010287.

[2] Stephen Diadamo et al. "Practical Quantum K-Means Clustering: Performance Analysis and Applications in Energy Grid Classification". In: *IEEE Transactions on Quantum Engineering* 3 (2022), pp. 1–16.

[3] Steph Foulds, Viv Kendon, and Tim Spiller. "The controlled SWAP test for determining quantum entanglement". In: *Quantum Science and Technology* 6.3 (Apr. 2021), p. 035002. DOI: 10.1088/2058-9565/abe458. URL: https://doi.org/10.1088%5C%2F2058-9565%5C%2Fabe458.

[4] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. *Quantum algorithms for supervised and unsupervised machine learning*. 2013. DOI: 10.48550/ARXIV.1307.0411. URL: https://arxiv.org/abs/1307.0411.

[5] Tawsifur Rahman et al. "Exploring the effect of image enhancement techniques on COVID-19 detection using chest X-ray images". In: *Computers in Biology and Medicine* 132 (2021), p. 104319. ISSN: 0010-4825. DOI: https://doi.org/10.1016/j.compbiomed.2021.104319. URL: https://www.sciencedirect.com/science/article/pii/S001048252100113X.

[6] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014. DOI: 10.48550/ARXIV.1409.1556. URL: https://arxiv.org/abs/1409.1556.