Quantum Spectral Clustering Application

Daniel E. Molina
Ming Hsieh Department of Electrical and Computer Engineering
University of Southern California
Los Angeles, California, USA
dem_634@usc.edu

***Abstract*— This research paper aims at exploring the possibility of implementing Quantum Spectral Clustering onto current day quantum computers and simulators. Spectral Clustering, in the classical case, aims at being a more generalized clustering algorithm compared to the well-known K-means algorithm. In this paper, we explain generally what K-means does, and how the classical spectral clustering works, to then explain how quantum spectral clustering on a quantum processing unit (QPU). This paper also aims to show the size of data that can be handled with today's QPUs and simulators and a general outlook on when it would be practical.**

***Index Terms*— Clustering, Spectral Clustering, Quantum, IBM, Qiskit**

I. INTRODUCTION

Year over year, the amount of unstructured data accumulated is growing at unprecedented rates. Machine learning clustering algorithms are able to extract inferences from those large unstructured data sets and group them according to some similarities. K-means is a vector quantization method that groups n data points into k clusters. Each data point is part of the nearest centroid in euclidean distance. The centroid is defined as the mean of each individual cluster of points. We iterate repeatedly to reduce the cost (ex: squared distance to nearest centroid). The only problem is that there is a high dependence on choosing the initial centroids to optimize over. This can be the difference between a good clustering and bad clustering. Putting the initial centroids too close to each other can also affect performance and outcome. On the other hand, classical spectral clustering has been shown to be a more general form of the K-Means algorithm and has shown to cluster various data sets much better. K-means algorithm is better capable of dealing with linearly separable data, while spectral clustering has the advantage on non-linearly separable data [1].

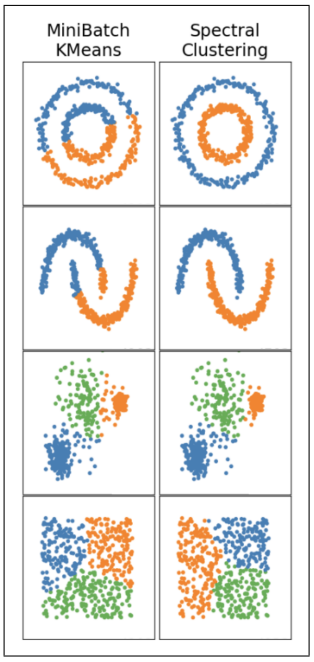

Fig. 1. K-Means vs Spectral Clustering classifying against different data sets (concentric circles, moons, overlapping blobs and highly noisy clusters). Evidently Spectral Clustering seems like a more general algorithm when it comes to classification. (Image taken from [2])

Quantum Spectal Clustering seeks to use the same underlying methodology as its classical counterpart but with some polynomial speed-up using a QPU. This paper shall talk about how classical and quantum spectral clustering works, and how it may or may not be able to be implemented of a current (2023) QPU.

Most quantum spectral clustering papers use a theoretical method called Quantum-RAM (QRAM). This is used to index and extract data quickly, but how to physically build it, is still an open question.

Instead this paper will utilize Grover's algorithm to search through data at $O(\sqrt{n})$ time complexity, and implement the algorithm.

This paper is structured as follows: Section II explains algorithm for classical spectral clustering. Section III explains its quantum spectral clustering counterpart. In Section IV provides details about running on a Quantum device or simulator; followed by conclusions and references.

II. CLASSICAL ALGORITHM

Given a data set $D = \{v_i\}_{i=0}^{N-1}$, where v_i represents a row of data with various features (or columns), our task is to separate the data D into k clusters based on certain similarities. The clustering outcomes is based on partitions $\{P_j\}_{j=0}^{k-1}$ with $D = \bigcup_{j=0}^{k-1} P_j$, and where $P_i \bigcap P_j = \emptyset$ for i $\neq$ j.

The similarity function $S : S(v_i, v_j)$ where $S(v_i, v_j) \in [0,1]$ which is the similarity between two data instances v_i and v_j given its individual features. In this article we shall use the gaussian similarity function which is given by $S(v_i, v_j) = e^{\frac{||v_i - v_j||_2^2}{2*\sigma^2}}$ where σ is the standard deviation of the data set.

Using the similarity function, we construct the graph $G(V_D, E_{DS})$ where V_D is the vertices (or each of the data points) and E_{DS} is each of the edges connecting $d-1$ nearest neighbors (d-1-NN). It must be noted that d is a user chosen value and $d << N$. The weights on each of those edges is given by the similarity function $S(v_i, v_j)$. This allows us to build the weight matrix given by:

$$w_{ij} = \begin{cases} S(v_i, v_j), & \text{if } i \neq j, \\ & \text{and } v_i \& v_j \text{ are d-1-NN} \\ 0, & \text{if } i = j \end{cases} \quad (1)$$

The degree matrix MD is a diagonal matrix where $MD \in \mathbb{R}^{N \times N}$.

$$MD(i,i) = \sum_{j=1}^{N-1} w_{ij} \quad (2)$$

The associated Laplacian matrix $L \in R^{N \times N}$ becomes:

$$L = MD - W \in \mathbb{R}^{N \times N} \quad (3)$$

L is symmetric and positive-semi-definite. L has eigenvectors u_i and eigenvalues λ_i which satisfy $0 = \lambda_0 \leq \lambda_1 \leq ... \leq \lambda_{N1}$. Notice that L has at least one zero eigenvalue, i.e. $\lambda_0 = 0$.

To normalize the Laplacian matrix, we use:

$$L_s = MD^{-1/2} \times L \times MD^{-1/2} = I - (MD^{-1/2} \times W \times MD^{-1/2}) \quad (4)$$

where L_s is a symmetric matrix. For more information about L_s, see Ref [3].

For step 1 of the spectral clustering algorithm, we obtain the k smallest eigenvalues of L_s, $\{\lambda_i\}$, where $i = 0, 1, .., k-1$ and their corresponding eigenvectors $\{u_i\}$. We take those eigenvectors to construct a matrix $A \equiv [u_0, ..., u_{k-1}] \in \mathbb{R}^{N \times k}$. The rows of A can be denoted as $\{y_i\}$, $i = 0, 1, ..., N-1$. For step 2 we apply the K-Means clustering algorithm to $\{y_i\}$ and group them into k subgroups $\{C_j\}_{j=0}^{k-1}$. It turns out that clustering $\{C_j\}$ using $\{y_i\}$ leads to a good clustering $\{P_j\}$ on $D = \{v_i\}$: if y_i and y_j belong to the same subgroup C_j , then v_i and v_j belong to the same subgroup P_j [4], [5].

The whole algorithm is given by Algorithm 1. The overall time complexity is given by $O(k \times N^3)$ where $O(N^3)$ is given by finding the eigenvalues of L_s and doing this k times.

Algorithm 1 Classical Spectral Clustering

Input: Dataset $D = \{v_i\}_{i=0}^{N-1}$, a given a user selected value d, and the number of clusters k.

Output: Clusters P_1,...,P_k

1) Obtain a similarity function $S(v_i, v_j)$ (i.e.: Gaussian Similarity function $S(v_i, v_j) = e^{\frac{||v_i - v_j||_2^2}{2*\sigma^2}}$)
2) Use the k-nearest-neighbor (kNN) algorithm to find (d-1) nearest neighbors to each of the data points (or vertices V_D in the graph $G(V_D, E_{DS})$. This will give you an adjacency matrix if 2 vertices v_i & v_j are connected.
3) Construct the weight matrix W and the Degree Matrix MD.
4) With the two matrices above, create the normalized Laplacian matrix L_s.
5) Calculate the k smallest eigenvalues $\{\lambda_i\}_{i=0}^{k-1}$ and corresponding eigenvectors $\{u_i\}_{i=0}^{k-1}$ of L_s.
6) Create $A \equiv [u_0, ..., u_{k-1}] \in \mathbb{R}^{N \times k}$ where the i^{th} row is denoted as $y_i \in \mathbb{R}^k$.
7) Cluster $y_i \in \mathbb{R}^k$, where $i = 0, ..., N-1$, into k clusters $\{C_j\}_{j=0}^{k-1}$ with the K-Means clustering algorithm.
8) Create $P_1, ..., P_k$ where $v_i \& v_{j'} \in P_j \iff y_i \& y_{j'} \in C_j$.

III. QUANTUM ALGORITHM

In the quantum version, there are several differences to operations of how the unitaries are implemented on a quantum processing unit (QPU). At a high level, the underlying concept of what we are doing in comparison to the classical case is still the same, but at a low level there are significant differences in terms of how to implement the algorithms on a QPU. We begin by using the same data set $D \in \mathbb{R}^{N\times m}$, with N data points and m features. We still use the same similarity function $S(v_i, v_j) = e^{\frac{\|v_i - v_j\|_2^2}{2*\sigma^2}}$ where σ is the standard deviation of the data set, but other similarity functions can be used. We construct the weight matrix W once again using the same d-1 nearest neighbor approach to its classical counterpart. We form the diagonal degree matrix MD, and construct the associated symmetric Laplacian matrix $L_s = I - (MD^{-1/2} \times W \times MD^{-1/2})$. L_s can be further re-scaled by $\frac{1}{2d}$ to make all eigenvalues $\{\lambda_i\} \in [0,1]$ of L_s. Scaling by such factor only affects the eigenvalues, but not the eigenvectors. We can still find the k smallest eigenvalues with convenience since it re-scales all eigenvalues evenly. We denote the eigenvalues of L_s as $\{\lambda_i\}_{i=0}^{N-1}$ and eigenvectors $\{|u_i\rangle\}_{i=0}^{N-1}$.

In order to encode the normalized Laplacian matrix into a unitary, we use $U = e^{2\pi i L_s}$. Then the eigenvalue estimation is $\lambda_j = 0.\lambda_{j1}\lambda_{j2}...$, where $\lambda_{ji} \in \{0,1\}$ is a binary decimal estimation. Using quantum phase estimation (QPE) we are able to utilize the Unitary to estimate all eigenvalues and eigenvectors of the matrix L_s $(e^{2\pi i\lambda_k}, |u_k\rangle)$ of U. The quantum phase estimation circuit U_{pe} can be constructed from the inverse quantum Fourier transform and a series of controlled-U^j gates, where $U = e^{2\pi i L_s}$. Quantum Phase Estimation satisfies

$$U_{pe}|0\rangle^{\otimes t}|u_k\rangle = \left|\tilde{\lambda}_k\right\rangle|u_k\rangle \tag{5}$$

where $\left|\tilde{\lambda}_k\right\rangle$ is given by $|\lambda_{k1}\lambda_{k2}...\lambda_{kt}\rangle$.

This is estimated up to t-bits with an error of $O(2^{-t})$.

Before applying phase estimation, it is essential to create a initial Unitary U_{in} which entangles the states to be able to represent all $2^n = N$ states. $U_{in} \equiv \Pi_{i=1}^{n} CNOT_{i,i+n} H^{\otimes n}$ where $|\Phi\rangle = U_{in}|0\rangle^{\otimes n}$.

For step 1, we prepare everything in the initial state $\left|\psi_0\right\rangle = |0\rangle^{\otimes t}|0\rangle^{\otimes n}|0\rangle^{\otimes n}$. Then we apply U_{in} to get $U_{in}\left|\psi_0\right\rangle = \frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}|0\rangle^{\otimes t}|i\rangle|i\rangle$.

For step 2, we apply the quantum phase estimation U_{pe} to get:

$$\left|\psi\right\rangle_{pe} \equiv U_{pe}U_{in}\left|\psi_0\right\rangle = \frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}|\lambda_i\rangle|u_i\rangle\left|u_i^*\right\rangle \tag{6}$$

The complexity for quantum phase estimation is given by $O(poly(logN)d^4/\epsilon)$ where ϵ represents the error rate [6].

For step 3, we apply grovers search to find the eigenvalues of L_s less than the threshold we provide. We have to find the eigenvalues smaller than $\tilde{\lambda}$ which we approximate through U_{pe}. To apply grover, we define an oracle f given as:

$$f(x) = \begin{cases} 1, & \frac{x}{2^t} < \tilde{\lambda} \\ 0, & \frac{x}{2^t} \geq \tilde{\lambda} \end{cases} \tag{7}$$

where x is a t-bit boolean variable. Now we can generate the quantum oracle O_f given by

$$O_f|x\rangle = (-1)^{f(x)}|x\rangle \tag{8}$$

Thus as O_f adds a negative phase to all $|x\rangle$ which satisfy the condition, we create the grover iteration G given by

$$G \equiv U_{inv}O_f = (2\left|\psi\right\rangle_{pe}\left\langle\psi\right|_{pe} - I)O_f \tag{9}$$

where U_{inv} is the intial state inverse with respect to $\left|\psi\right\rangle_{pe}$.

An important note is that $t = n + \left\lceil 2 + log\frac{1}{2\epsilon_0}\right\rceil$, $n = \lceil logN\rceil$ where $1-\epsilon_0$ is the success probability of the phase estimation.

After applying the grover iteration, we should amplify the eigenvalues smaller than $\tilde{\lambda}$ to yield us the output:

$$\left|\psi\right\rangle_{out} = \frac{1}{\sqrt{k}}\sum_{\lambda_i<\tilde{\lambda}}|\lambda_i\rangle|u_i\rangle\left|u_i^*\right\rangle \tag{10}$$

The total complexity of step 3 is $O(\sqrt{\frac{N}{k}}poly(logN))$ [6]. We need to find the value of k_0 which is the number of eigenvalues smaller than $\tilde{\lambda}$. We can obtain this by applying the quantum counting circuit to $\left|\psi\right\rangle_{pe}$. We will then use k to find the indicator matrix X in step 4.

For step 4, we take the quantum measurement and optimize on the eigenstate register, whose density matrix is given by

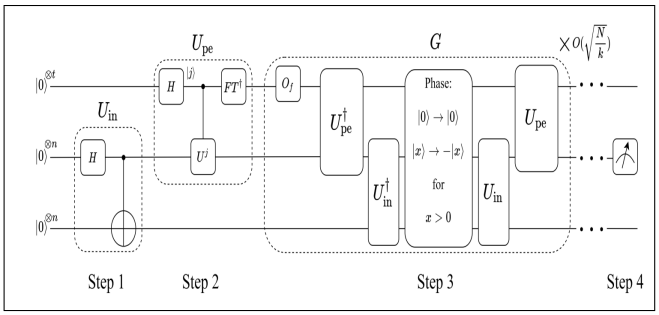

Fig. 2. Step 1 gives us the bell state entanglement between the bottom 2n qubits. Step 2 gives us Unitary quantum phase estimation to estimate the eigenvalues and eigenvectors of L_s. Step 3 is Grover's algorithm which is in charge of amplifying and extracting only eigenvalues which is estimated to be smaller than $\tilde{\lambda}$. Step 4 is given by a quantum measurement on the eigenvector register to evaluate and optimize $< M >= Tr(\rho M)$, where $M = XX^T$ and X is the clustering indicator matrix.

$$\rho = Tr_{1,3}(|\psi\rangle_{out}\langle\psi|_{out}) = \frac{1}{k}\sum_{i=0}^{k-1}|u_i\rangle\langle u_i| = \frac{1}{k}AA^T \tag{11}$$

where A is the matrix told before in the classical case: $A \equiv [u_0, ..., u_{k-1}] \in \mathbb{R}^{N\times k}$ with the rows denoted as $\{y_i\}$. We define $X = (x_{ij}) = [x_0, ..., x_{k-1}] \in \mathbb{R}^{N\times k}$ given by

$$x_{ij} = \begin{cases} \frac{1}{\sqrt{s_j}}, & y_i \in C_j \\ 0, & y_i \notin C_j \end{cases} \tag{12}$$

where $s_j = |C_j|$. The clustering on $y_i{}_{i=0}^{N-1}$ can be formulated as an optimization problem given the following,

$$\max_X Tr(pXX^T) = \max_X Tr(\rho M) = \max_X < M > \tag{13}$$

Finding the optimal X is an NP-hard problem. Therefore, using a heuristic algorithm like hill-climbing, are good enough to find an acceptable yet sub-optimal X within $O(kN)$ iterations.

According to paper [6], the total complexity of the entire algorithm gives us $O(\sqrt{k}N^{\frac{3}{2}}poly(logN)d^4/\epsilon)$. I wasn't very convinced by what this paper said being a "speed-up", so I had to test this theory out myself. I plotted the graphs and in fact showed that this is not the case for certain polynomial functions given by figure 4.

Doing further research it was especially interesting how amazing this algorithm scales as data doubles every 3 years. We can see that approaching 1×10^{12} data points (N) we only need about 180 total qubits

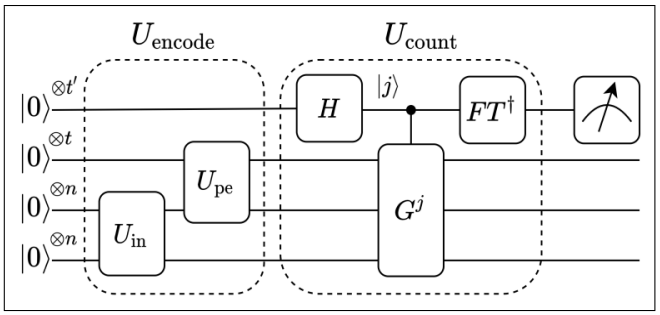

Fig. 3. This is the circuit for quantum counting the values k_0 which are less than where U_{encode} are steps 1 and 2 from the original, and G is given by the grover iteration repeated j times to apply quantum counting U_{count}

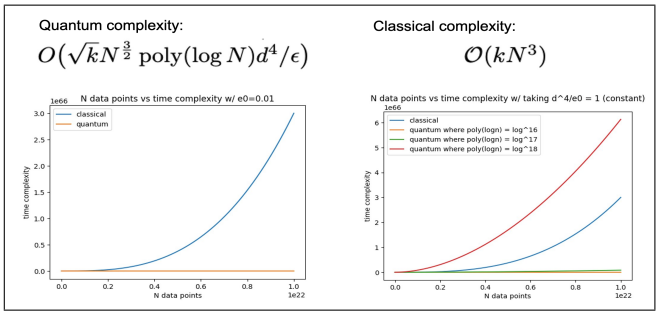

Fig. 4. This graph compares the classical vs the quantum complexities given by the equations.

for this circuit (given by figure 5). The complication becomes creating the gaint unitaries, so there are increasing problems that we must tackle such as quantum error correction, more efficient unitary implementation, among other things.

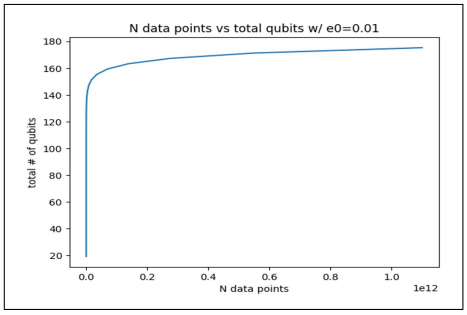

Fig. 5. This graph shows the scalability of the total number of qubits needed vs N (the number of data points).

Upon being able to implement this complicated circuit, I encountered many problems. I first needed to construct $2^{(t+t+n+n)}\times2^{(t+t+n+n)}$ Unitaries. Numpy already struggled with $50{,}000\times50{,}000$ unitary matrix. If the unitary is $50{,}000\times50{,}000$ dimension, this yields us $t+t+n+n\approx16$ with $e_0 = 0.01$ and $n = 0.178$, so not even one qubit. However, modifying the error of

the quantum phase estimation to be 0.13, we yield $n = 2.03$, which is only a 4×4 L_s matrix. This doesn't prove very useful in the current NISQ era and we are better off using classical computers for now, but will definitely be useful once we obtain 400 near-noiseless logical qubits (assuming we don't expand the unitaries to classical storage).

IV. CONCLUSIONS

In conclusion, this is a very useful algorithm once we scale up the number of qubits and reduce the noise in them. This would carry out tons of research and plenty of implementation and experiments before this comes into fruition. This algorithm can encode exponentially (theoretically) the number of data points and achieve results faster than K-Means if $poly(logN)$ is smaller than a certain threshold as we showed in figure 4. With current devices and a high error rate for QPE of 13% we are only able to implement a 4x4 L_s matrix which is the similarity between 4 points. This doesn't yield us much useful information, therefore many advances need to be created to be a significant improvement over classical methods.

ACKNOWLEDGMENT

I would like to thank Dr. Andrea Delgado from Oak Ridge National Laboratory for her guidance and topic of choice which will complement my work for this upcoming summer on Quantum Machine Learning applied to High Energy Physics. Thank you to Professor Rosa, and to the class for listening to my short presentation on Quantum Spectral Clustering.

REFERENCES

[1] Chang, EC., Huang, SC. Wu, HH. Using K-means method and spectral clustering technique in an outfitter's value analysis. Qual Quant 44, 807–815 (2010). https://doi.org/10.1007/s11135-009-9240-0.

[2] (2007-2023). 2.3.1. Overview of Clustering Methods. Scikit-learn. Retreived May 7th, 2023 from https://scikit-learn.org/stable/modules/clustering.html

[3] Hai-Ling Liu, Su-Juan Qin, Lin-Chun Wan, Chao-Hua Yu , Shi-Jie Pan , Fei Gao and Qiao-Yan Wen. (2022). A quantum algorithm for solving eigenproblem of the Laplacian matrix of a fully connected weighted graph. https://doi.org/10.48550/arXiv.2203.14451.

[4] A. Ng, M. Jordan, Y. Weiss, On spectral clustering: Analysis and an algorithm, Advances in neural information processing systems 14 (2001).

[5] U. Von Luxburg, A tutorial on spectral clustering, Statistics and computing 17 (2007) 395–416.

[6] Qingyu Li, Yuhan Huang, Shan Jin, Xiaokai Hou, and Xiaoting Wang. (2022). Quantum spectral clustering algorithm for unsupervised learning. https://arxiv.org/pdf/2203.03132.pdf.