Fast Approximate Spectral Clustering with PAM and EWKM

Group 10 YZZQD

Yuanqi Li
Department of Computer Science
University of California, Los Angeles
lyuanqi@cs.ucla.edu

Yunqi Guo
Department of Computer Science
University of California, Los Angeles
luckiday@ucla.edu

Xueyin Yu
Department of Electrical Engineering
University of California, Los Angeles
soniayu@ucla.edu

ABSTRACT

Spectral clustering is a technique that makes use of the spectrum of the similarity matrix of the data to perform dimension reduction and then clustering[17]. Spectral clustering has been shown to be a very effective clustering method and can be applied to common data mining problems such as community detection by modularity maximization, community detection by statistical inference, and normalized-cut graph partitioning[9]. While spectral clustering can produce high-quality clusterings, it is not a practical choice for clustering large data sets. This is due to its unscalable computational complexity of $O(n^3)$, where n denotes the number of data points. To address this problem, Donghui Yan, et al. published the paper 'Fast Approximate Spectral Clustering' and proposed to use K-means and RP trees as a data preprocessing step before the actual spectral clustering step, to improve the overall runtime efficiency while upholding the clustering quality[18]. In this paper, we will extend Donghui Yan, et al.'s research by applying and evaluating partitioning around medoids (PAM) and entropy weighted K-means (EWKM) for the data preprocessing step.

Categories and Subject Descriptors

I.2.6 [**Information Search and Retrieval**]: Clustering; [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Performance

Keywords

Unsupervised Learning, Spectral Clustering

1. INTRODUCTION

Data clustering has been one of the most popular topics in data mining. While various researchers are introducing many clustering methods, a clustering method called spectral clustering stands out from the others due to its adoption to both convex and non-convex datasets, and its high quality in clustering results.

Despite the flexibility and effectiveness, spectral clustering has a significant disadvantage of unscalable computation time of $O(n^3)$. Such high computation time makes spectral clustering only practically applicable to clustering problems with small to medium dataset size at most. However, there are more and more clustering applications which require large size of the dataset, to ensure sophisticated and hence useful clustering results.

In the effort of resolving such scalability problem of spectral clustering, Donghui Yan, et al. from University of California, Berkeley, developed a general framework for fast approximate spectral clustering method. It mainly consists of three steps. The first step is to apply a preprocessor to the original input dataset, to reduce the size of the original dataset. The reduced dataset acts as the representative points of the original data points, and will then be passed to the actual spectral clustering method. Lastly, the clustering results for the reduced dataset, i.e. the representative points, will be used to assign cluster memberships to the original data points. Donghui Yan, et al.[18] chose to apply K-means and RP trees respectively as the preprocessor to the framework, and conducted evaluation and analysis against each preprocessor.

Since the effectiveness of the preprocessor is the key factor to the final performance of the fast approximate clustering method, we will focus on extending the research done by Donghui Yan, et al. by applying partitioning around medoids (PAM) and weighted K-means (EWKM) as the preprocessor to the fast approximate spectral clustering framework, respectively.

The remainder of this paper is organized as follows. In Section 2 we will give a brief introduction to the original spectral clustering method. Section 3 will introduce some major-related work of spectral clustering and fast approximate spectral clustering methods. In Section 4, we will give a brief overview of the fast approximate spectral clustering framework, along with the new preprocessors of PAM and EWKM. In Section 5, we will describe the experimental design and setup, along with the experimental results. We will then analyze and discuss the empirical performance for each preprocessor in Section 6. Finally, we will present a conclusion of our research in Section 7.

2. SPECTRAL CLUSTERING

2.1 Spectral clustering algorithm

Considering an undirected graph $G = (V, E)$, where the vertex $\mathbf{x}_i$ corresponds to a data point $\mathbf{x}_i$ in the feature space, which belongs to a set of data points $\mathbf{x}_1$, $\mathbf{x}_2$, . . . , $\mathbf{x}_n$ ($\mathbf{x}_n \in \mathbb{R}^d$). We define the similarity matrix $A = (a_{ij})_{i,j=1}^{n}$, where the element $a_{i,j}$ is the weight assigned to each of the edge $(i, j) \in E$, representing the similarity between node i and node j.

In this paper, we mainly adopt the *normalized cuts (Ncut) algorithm* [15] our spectral clustering technique, which aims to partition the data into disjoint sets so that each data point belongs to only one cluster. Let $\mathbf{V} = (V_1, V_2, ..., V_m)$ denote a partition of $\mathbf{V}$ into m disjoint sets and consider the following optimization criterion:

$$\text{Ncut} = \sum_{i=1}^{m} \frac{W_{V_i,\mathbf{V}} - W_{V_i,V_i}}{W_{V_i,\mathbf{V}}} \tag{1}$$

where $W_{V_k,V_l} = \sum_{i \in V_k, j \in V_l} a_{i,j}$, $V_k$and V_l are two (possibly overlapping) subsets of $\mathbf{V}$. In equation (1), the value of the denominator is equal to the total degrees of all the vertices in subset V_i. As for the numerator, the i^{th} term is the sum of the similarity on edges while excluding the subset . The target of minimizing the sum of such terms is to find a partition where the similarity among the vertices in a set Vi is high and the sizes of V_i are balanced while the similarity of V_i, V_j is low.

Unfortunately, minimizing normalized cut in (1) is NP-complete. However, when we embed the normalized cut problem in the real value domain, an approximate discrete solution can be found efficiently. In particular, we can rewrite equation (1) as a normalized quadratic form involving indicator vector, which are later replaced with real-valued vectors. The optimization problem is thus relaxed into the continuous domain by solving a generalized eigenvector problem. The normalized Laplacian matrix $\mathcal{L}$ of A is defined as follows:

$$\mathcal{L} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \tag{2}$$

where $D = \mathbf{diag}\,(d_i, ...d_n)$, $d_i = \sum_{j=1}^{n} a_{i.j}$, $i = 1, ..., n$.

The normalized Laplacian is sometimes defined as $\mathcal{L} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$, where I is the identity matrix, but the two matrices differ only in a trivial transformation of their eigenvalues and the eigenvectors are the same for both.

The eigenvector of the normalized Laplacian is the real valued solution to Ncut problem. In this paper, we mainly focus on the classical recursive Ncut algorithm, which is the simplest case of a binary partition. [15] components of the second eigenvector are thresholded to define the class memberships of the data points. We assume that the number of clusters is set in advance and we recursively run the bipartitioning algorithm until desired numbers of clusters have been formed. We can refer to Algorithm 1 for a specific spectral bipartitioning algorithm, where a Radial Basis kernel function 'Gaussian' is used as the pairwise similarity measure.

```
Algorithm 1: Spectral Clustering
  Input  : $n$ data points $\mathbf{x}_i$, $i = 1, ..., n$, and $\mathbf{x}_i \in \mathbb{R}^d$
  Output: Two-way partition $S_1$ and $S_2$ of the input
          data
    1. Compute the similarity matrix $A$ with elements:
           $a_{i,j} = e^{-\frac{||x_i - x_j||^2}{2\sigma^2}}$
    2. Compute the diagonal matrix $D$ with elements:
           $d_i = \sum_{j=1}^{n} a_{i.j}$
    3. Compute the normalized Laplacian matrix $\mathcal{L}$ of $A$:
           $\mathcal{L} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$
    4. Compute the second eigenvector $v_2$ of matrix $\mathcal{L}$
    5. Obtain the two partitions according to the signs of
       the elements of $v_2$:
           $S_1 = i : v_{2i} > 0, S_2 = i : v_{2i} \leq 0$
```

2.2 Spectral clustering vs. K-means

K-means is a simple and easy-to-implement method for clustering, which optimizes intra-cluster similarity. Moreover, it is relatively efficient with time complexity $O(tkn)$, where n represents the number of objects, k is the number of clusters, and t stands for the times of iterations[4]. However, K-means is not suitable to discover clusters with non-convex shapes and shows weakness in handling noisy data and outliers.

In contrast, spectral clustering can deal with both convex and non-convex data sets. It exhibits superior clustering quality on small datasets. Nevertheless, spectral clustering is not scalable since it is computationally expensive for large datasets with $O(n^3)$ runtime [3].

3. RELATED WORK

This paper is greatly inspired by the paper 'Spectral methods for network community detection and graph partitioning' by M. E. J. Newman [9]. From Newman's paper, we learned how flexible and effective spectral clustering method could be. It turns out that spectral clustering method could be used to solve popular problems such as community detection by statistical inference, and normalized-cut graph partitioning. Nevertheless, the original spectral clustering is only practically useful to small to medium size datasets. Such constraint of spectral clustering is what motivates the paper 'Fast Approximate Spectral Clustering ' by Donghui Yan, et al. [18], in which Donghui Yan, et al. developed the fast approximate spectral clustering framework to try to improve the overall computational performance of the original spectral clustering method.

Our paper is in fact mostly based on, and yet with certain research extensions to the findings from Donghui's paper. In this paper, we are essentially utilizing the fast approximate spectral clustering framework described in Donghui. While examining the effectiveness of the framework using one of the original K-means preprocessors, we are introducing two new preprocessors, partitioning around medoids (PAM) and entropy weighted K-means (EWKM). We will evaluate the performance of these two preprocessors in the same manner as described in Donghui's paper, and check if any performance improvement can be observed.

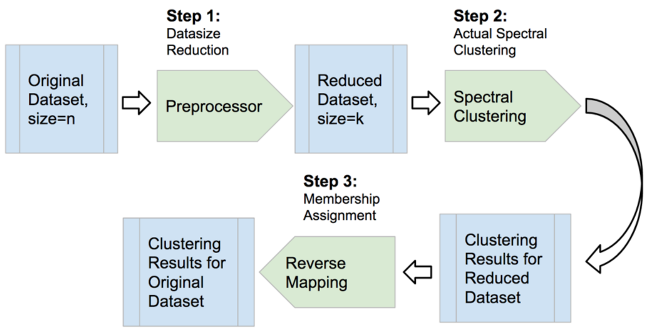

Figure 1: Fast Approximate Spectral Clustering Framework

4. FAST APPROXIMATE SPECTRAL CLUSTERING

As illustrated in the Figure 1, the fast approximate spectral clustering framework consists of three major steps of operations. The first step is to apply a data preprocessor to the original input dataset that has a size of n. The output of the preprocessor is a reduced dataset that has a size of k, with data points which act as the representative data points of the data points from the original dataset, where k should be much smaller than n. Secondly, the reduced dataset with size k is then fed into the actual spectral clustering operation. Lastly, the clustering results from the spectral clustering are used to assign cluster memberships to the original data points according to their respective representative points. Hence, the overall runtime for the fast approximate spectral clustering method would be $O(n) + O(k^3) + O(n)$, while the original runtime is $O(n^3)$.

The key factor of such fast approximate spectral clustering method is the choice of the preprocessor. The effectiveness of such preprocessor directly determines the final performance of the approximate spectral clustering, in terms of accuracy and runtime. The reduced size k directly determines the overall runtime of the fast approximate spectral clustering. Hence we need to minimize the size of k in order to minimize the overall runtime. In the meantime, however, reducing k by too much could result in too much data distortion during the data reduction process, and thereby introducing too much inaccuracy in the final clustering results. Hence the goal is to choose a preprocessor and find an appropriate k value that minimizes the overall runtime while maintaining an acceptable clustering accuracy. The final chosen k value should be within the range that is appropriate to be passed into spectral clustering such that $O(k^3)$ runtime is acceptable.

4.1 Fast approximate spectral clustering with PAM

The Partitioning Around Medoids (PAM) algorithm is the most common realization of k-medoids clustering. The k-medoids algorithm is a clustering algorithm related to the K-means algorithm since they both aim to partition the dataset into groups. K-means attempts to minimize the total squared error while k-medoids minimizes the sum of dissimilarities between points labeled to be in a cluster and a point designated as the center of that cluster. In contrast to the K-means algorithm, k-medoids chooses datapoints as centers (medoids or exemplars)[11].

In our paper, we replace K-means with PAM to be a preprocessor for spectral clustering since theoretically it is more robust to noise and outliers as compared to K-means. Particularly, we name it 'PAM-based approximate spectral clustering' (PAMSP) algorithm, which has the form in Algorithm 2.

Algorithm 2: PAMSP

```
Input  : $n$ data points $\mathbf{x}_i$, $i = 1, ..., n$, and $\mathbf{x}_i \in \mathbb{R}^d$
         number of representative points $k$
Output: $M$-way partition of the input data
  1. Perform PAM with $k$ clusters on $\mathbf{x}_i$, $i = 1, ..., n$, to:
       a. Compute the centroids of each cluster $y_1, y_2, ..., y_k$
          and set them as the $k$ representative points. Build
          correspondence relationship by associating each $x_i$
          with the nearest cluster centroid $y_j$.
       b. Perform spectral clustering on $y_1, y_2, ..., y_k$ to
          assign a m-way cluster membership for each $y_i$.
  2. Perform spectral clustering on $y_1, y_2, ..., y_k$ to assign a
     m-way cluster membership for each $y_i$.
  3. Do the reverse mapping for each $x_i$ according to the
     cluster membership of the corresponding centroid $y_j$.
```

The computational complexity of step 1 (PAM) is $O(n^2)$. Given that the complexity of step 2 is $O(k^3)$ and the complexity of step 3 is $O(n)$, the overall computational complexity of PAMSP is $O(n^2)$ since $n >> t$.

4.2 Fast approximate spectral clustering with EWKM

The entropy weighted K-means (EWKM) clustering algorithm is based on the K-means approach to clustering. An initial set of k means are identified as the starting centroids. Observations are clustered to the nearest centroid according to a distance measure. This defines the initial clustering. New centroids are then identified based on these clusters. Weights are then calculated for each variable within each cluster, based on the current clustering. The weights are a measure of the relative importance of each variable with regard to the membership of the observations to that cluster. These weights are then incorporated into the distance function, typically reducing the distance for the more important variables. New centroids are then calculated, and using the weighted distance measure each observation is once again clustered to its nearest centroid [7].

Using EWKM as preprocessor is out of the consideration that it may converge more quickly than K-means due to the weighted distance measure. We can realize the 'EWKM-based approximate spectral clustering' (EWKMSP) algorithm by replacing the step 1 in Algorithm 2 with the EWKM method. The EWKMSP algorithm has the form in Algorithm 3.

The computational complexity of step 1 (EWKM) is $O(knt)$, where t is the times of iterations. Therefore, the total computational cost of EWKMSP is $O(knt) + O(k^3)$.

```
Algorithm 3: EWKMSP
Input  : $n$ data points $\mathbf{x}_i$, $i = 1, ..., n$, and $\mathbf{x}_i \in \mathbb{R}^d$
         number of representative points $k$
Output: $M$-way partition of the input data
  1. Perform EWKM with $k$ clusters on $\mathbf{x}_i$, $i = 1, ..., n$, to:
       a. Compute the centroids of each cluster $y_1, y_2, ..., y_k$
          and set them as the $k$ representative points. Build
          correspondence relationship by associating each $x_i$
          with the nearest cluster centroid $y_j$.
       b. Perform spectral clustering on $y_1, y_2, ..., y_k$ to
          assign a m-way cluster membership for each $y_i$.
  2. Perform spectral clustering on $y_1, y_2, ..., y_k$ to assign a
     m-way cluster membership for each $y_i$.
  3. Do the reverse mapping for each $x_i$ according to the
     cluster membership of the corresponding centroid $y_j$.
```

5. EXPERIMENTAL DESIGN AND EVALUATION

5.1 Environment

To ensure an isolated, consistent and repeatable evaluation environment, we decided to run all the evaluation experiments on an *AWS EC2* host. The host tier chosen was t2.micro, with the following specifications:

- Operating System: Ubuntu Xenial 16.04.1 LTS
- CPU: Single-core Intel Xeon processor with turbo modes up to 3.3GHz
- RAM: 1GB
- Storage: 8GB SSD

The host was deliberately chosen to have relatively low computational performance, so that the differences in runtime performance could be better observed for different experimental trials.

5.2 Datasets

Our experiments are conducted with three different datasets listed in Table 1, which are all taken from the UCI machine learning repository [1]. The Connect-4 dataset has been normalized so that the mean of all features is 0 and the standard deviation is 1.

Table 1: Experiment Dataset

Dataset	Size	#Features	#Instances	#Classes
Musk	Medium	166	6598	2
mGamma	Medium	10	19020	2
Connect-4	Large	42	67557	3

5.3 Evaluation metrics

We adopted two dimensions to estimate the clustering performances: the runtime and the accuracy. In our experiment environment, the runtime of each method was taken as the past time for the partition function (exclude the data preprocessing time). To compare with the state-of-art method proposed by Donghui, we used the same measurement equation in his paper [18]. This method demands a search over permutations of the classes. The clustering accuracy β is defined as

$$\beta(f) = \max_{\tau \in \Pi_z} \left\{ \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}\{\tau(f(\mathbf{x}_i)) = \theta(\mathbf{x}_i)\} \right\} \qquad (3)$$

where $\mathbb{I}$ is the indicator function, $z = \{1, ..., k\}$ denote the set of class labels, and Π_z is the set of all permutations on z. $\theta(\cdot)$ and $f(\cdot)$ represent the real label and the algorithm-generated label of each data point, separately.

5.4 Competing algorithms

We compare the performance of PAMSP and EWKMSP with four algorithms, first three of which are K-means and K-means related methods, PAM and weighted K-means, and the last of which, KASP, is the state-of-art algorithms proposed by Donghui [18]. All of these six algorithms were implemented in R code and can be download from GitHub repository luckiday/fastSqectralClustering.

K-means can be implemented with a built-in function in R, which has multiple choices for methods in different purpose and performance. These methods includes Hartigan-Wong's algorithm[5], Lloyd's algorithm[8], MacQueen's algorithm[16], etc. We applied Hartigan-Wong's algorithm[5] to have a comparison with KASP [18]. For that the performance of K-means can be dramatically different depending on the initialization function [12] [2], we used the sampling-based two-stage algorithm, which is reported to be the highest level of accuracy attained for the UCI datasets.

Partitioning Around Medoids (PAM), a more robust version of K-means, is provided by the library 'cluster' in R[13], which can Partition the data into k clusters 'around medoids'. The basic PAM function we used is based on *Partitioning around medoids (program pam)*[10].

Entropy weighted K-means (EWKM) is provided by R package 'wskm'[14] based on the theory proposed by Jing[7]. This algorithm is a subspace clusterer ideal for high dimensional data. Along with each cluster this function also obtains varying weights that supplies a relative measure of the importance of each variable to that cluster. From our observation, this method outperform the original K-means algorithm with less iterations needed to converge.

As for the KASP algorithm[18], we utilized the open source code provided by the authors. To make the experiment more convincing, we run the code in the same environment and with the same datasets.

See below for discussion of the performance comparison of these algorithms.

5.5 Evaluation results

The first step to make the algorithm work is to find the number of representative points k for best performance. We applied heuristic process in this case to obtain the correlation between different ks and performance.

Figure 2 and Figure 3 give us visualized relation between k and runtime, k and accuracy. What we can observe from the charts is that when k increase from 10 to 300, the runtime will increase linearly. However, as for the accuracy, the accuracy meet an upper bound when k reach 150. So, in this

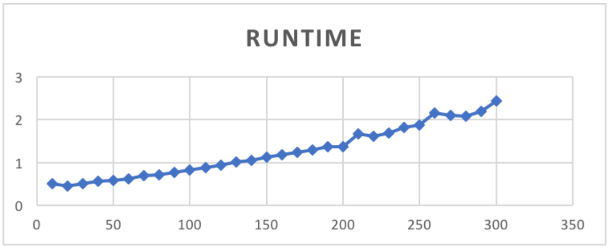

Figure 2: Relation between k and runtime with EWKMSP algorithm for dataset Connect-4

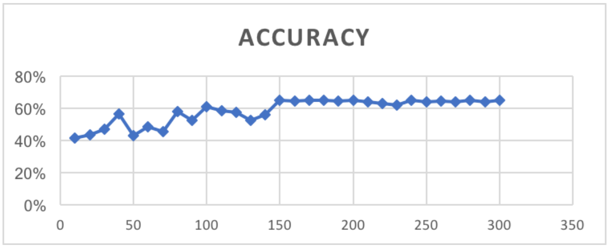

Figure 3: Relation between k and accuracy with EWKMSP algorithm for dataset Connect-4

case, we chose 200 representative points in the first step. We worked with this method to calculate the best performance ks in spectral-related methods $KASP, PAMSP, EWKMSP$ in our experiments.

We evaluated six clustering methods with three datasets from UCI machine learning repository[1]. Two of them represent medium size datasets and one is large dataset with over 60,000 instances. To compare the performances, the running results are listed in Table 2 and Table 3.

From Table 2 we see that the accuracy of spectral-related clustering algorithms (KASP, PAMSP, EWKMSP) are considerably greater than the non-spectral methods(K-means, PAM, EWKM), which validated the conclusion given in the section 8 of Donghui's paper[18]. On average, the former outperformed over 10% in each test case. What we can also see from the table is that in some samples, weighted K-means defeated the original K-means algorithm and PAM algorithm. Due to this reason, the weighted K-means clustering spectral achieved the highest accuracy in dataset *Musk*.

The runtime table (Table 3) shows us the running time in seconds for different clustering experiments. To make it more intuitive, we draw the logarithm bar chart in Figure 4. In this chart, the lower the bar is, the faster the algorithm

Table 2: Evaluation of K-means, PAM, EWKM, KASP, PAMSP, EWKMSP on UCI datasets. The content represent the accuracy of each method in corresponding dataset.

Accuracy	Connect-4	mGamma	Musk
K-means	50.20%	64.90%	54.00%
PAM	47.30%	61.70%	53.90%
EWKM	51.60%	64.10%	69.10%
KASP	65.00%	70.40%	54.80%
PAMSP	65.70%	65.10%	54.93%
EWKMSP	65.50%	69.40%	74.40%

Table 3: Evaluation of K-means, PAM, EWKM, KASP, PAMSP, EWKMSP on UCI datasets. The content represent the runtime of each method in corresponding dataset.

Runtime(s)	Connect-4	mGamma	Musk
K-means	1.04	0.068	0.152
PAM	5.11	0.2	0.916
EWKM	1.87	0.166	0.088
KASP	13.78	40.016	48.244
PAMSP	141	353.544	386.665
EWKMSP	3.42	3.816	18.716

runs. One of the observations we can obtain from this graph is that spectral method truly enlarges the running time in all three K-means-related methods. On the other hand, by comparing the yellow bars: yellow, blue and green, we can see that our method PAMSP has a slower speed than the state-of-art method KASP. However, the second method we proposed EWKMSP had a faster speed than KASP. Specifically, the speedup could be from 3x to 10x.

Comparing Table 2 and 3, we see that PAMSP and EWKMSP are roughly comparable to KASP regarding accuracy. As for speed, EWKMSP is 3 to 10 times faster than KASP in the experiments.

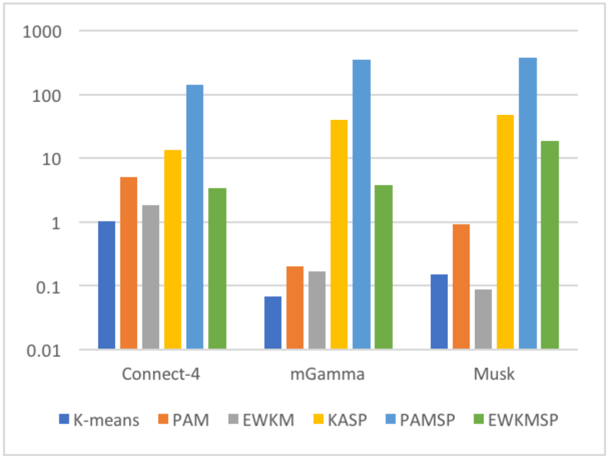

Figure 4: Logarithm bar chart to compare the runtime of different clustering methods. The vertical axis is in seconds.

6. PERFORMANCE ANALYSIS

In this section, we will theoretically analysis the performance of the three spectral-related methods, KASP, PAMSP and EWKMSP. In the end, we will tell why EWKMSP we proposed can beat KASP.

First, we compare the speed of KASP and PAMSP. As we have mentioned in section 4, the time complexity of KASP and PAMSP are $O(knt)$, $O(kn^2)$, respectively. For that n is far greater than the iteration t, the running of PAM method, which take the majority part of the total runtime can make PAMSP algorithm immensely slow in large test cases. The bar chart in Figure 4 can also validate this deduction. For each dataset, PAM(orange bar) runs 4 to 6 times slower than K-means(Navy). And in the spectral-related methods, the runtime of PAMSP(blue) is also around 5 times larger than KASP(yellow).

Secondly, we briefly discuss why EWKMSP runs faster than KASP. Section 4 has introduced that the time complexity of EWKMSP is $O(knt)$, the same as KASP. In our experiment, we analyzed the number of iteration to converge, t. What we found was that weighted K-means method needed less than five iterations to converge, however, in the test cases, K-means required 20 to 30 iteration to converge. This is the factor that led to a short runtime for EWKMSP. More significantly, the accuracy of KASP and EWKMSP were almost the same, which means that the shrink of iteration time did not influence the accuracy. There might be two possible reasons. One of the hypothesis is that EWKM runs faster than K-means algorithm to achieve the same precision. So, EWKMSP can get a compatible representative point set as KASP method with a shorter runtime. The second probable reason is that EWKM compromises the partition accuracy to achieve a higher speed in the first step of EWKMSP algorithm. However, this compromise does not have influence on the final accuracy for that cluster number is much smaller than the number of sampled points. The second hypothesis can be explained with the theory[6] proposed by Ling et al. In short, spectral clustering method can handle data with perturbation.

7. CONCLUSION

Based on the fast clustering framework, we have proposed two practical algorithms, PAMSP and EWKMSP. Both algorithms keep high-accuracy performance. EWKMSP method significantly reduces the expense of representative points picking, which retaining superior clustering accuracy. Evaluation on UCI datasets shows that the speed of our proposed method can reach three to ten times faster than the state-of-art algorithm.

8. ACKNOWLEDGEMENT

We would like to show our gratitude to our professor Yizhou Sun who provided insight and expertise that greatly assisted the research. We thank our classmates who gave us feedbacks during our presentation.

9. REFERENCES

[1] A. Asuncion and D. Newman. Uci machine learning repository, 2007.

[2] M. E. Celebi, H. A. Kingravi, and P. A. Vela. A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Systems with Applications*, 40(1):200–210, 2013.

[3] D. Hamad and P. Biela. Introduction to spectral clustering. In *Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on*, pages 1–6. IEEE, 2008.

[4] J. A. Hartigan and J. Hartigan. *Clustering algorithms*, volume 209. Wiley New York, 1975.

[5] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[6] L. Huang, D. Yan, N. Taft, and M. I. Jordan. Spectral clustering with perturbed data. In *Advances in Neural Information Processing Systems*, pages 705–712, 2009.

[7] L. Jing, M. K. Ng, and J. Z. Huang. An entropy weighting k-means algorithm for subspace clustering of high-dimensional sparse data. *IEEE Transactions on knowledge and data engineering*, 19(8), 2007.

[8] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE transactions on pattern analysis and machine intelligence*, 24(7):881–892, 2002.

[9] B. Karrer and M. Newman. Spectral methods for network community detection and graph partitioning. *Physical Review E*, 83:8016107, 2011.

[10] L. Kaufman and P. J. Rousseeuw. Partitioning around medoids (program pam). *Finding groups in data: an introduction to cluster analysis*, pages 68–125, 1990.

[11] H.-S. Park and C.-H. Jun. A simple and fast algorithm for k-medoids clustering. *Expert systems with applications*, 36(2):3336–3341, 2009.

[12] J. M. Pena, J. A. Lozano, and P. Larranaga. An empirical comparison of four initialization methods for the k-means algorithm. *Pattern recognition letters*, 20(10):1027–1040, 1999.

[13] R Core Team. *R Package cluster*. R Foundation for Statistical Computing, Martin Maechler, Peter Rousseeuw, Anja Struyf, 2017.

[14] R Core Team. *R Package wskm*. R Foundation for Statistical Computing, Graham Williams, Joshua Z Huang, Xiaojun Chen, 2017.

[15] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[16] K. Wagstaff, C. Cardie, S. Rogers, S. Schrödl, et al. Constrained k-means clustering with background knowledge. In *ICML*, volume 1, pages 577–584, 2001.

[17] Wikipedia. Spectral clustering — wikipedia, the free encyclopedia, 2017. [Online; accessed 18-March-2017].

[18] D. Yan, L. Huang, and M. I. Jordan. Fast approximate spectral clustering. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 907–916. ACM, 2009.

APPENDIX

A. TASK DISTRIBUTION

Table 4: Task distribution

Task	People
Project topic research/proposal	ALL
Related work survey	Yuanqi Li
Collecting and processing data	Yunqi Guo
Implementing fast spectral clustering with PAM	Xueyin Yu
Implementing fast spectral clustering with EWKM	Yunqi Guo
Evaluating and comparing algorithms	Yunqi Guo
Writing paper	ALL
Formatting paper to ACM format	Yunqi Guo
Proof-reading the final paper	ALL