A GA BASED K-MEANS CLUSTERING FOR WINE CLUSTERING DATA

Bidhan Bashyal
Computer Science and Engineering
Michigan State University
bashyalb@msu.edu

Aarush Mathur
Electrical and Computer Engineering
Michigan State University
mathura5@msu.edu

ABSTRACT

A genetic algorithm along with the K Means algorithm is implemented in this project. This method evolves a population of a chromosome representing a division of objects into a different number of clusters. One point crossover along with mutation strategy which reassigns the objects to the cluster is used. Ranking selection mechanism is used for selection. Two different fitness function have been tested on two different datasets. The main idea of this project is to design a method which is applicable to the datasets with high number of features. Result has shown that this method performs better and has high accuracy than that of traditional K Means algorithm.

KEYWORDS

Fitness Function, Principal Component Analysis, Ranking Selection Mechanism

1 Introduction

Clustering is an unsupervised data analysis technique which is used in categorizing data with similar characteristics in a cluster. There are different kind of clustering methods and have been widely used in various domains such as biology, sociology, image processing. Out of various clustering methods, K-means is the most popular and most widely used because of its efficiency and efficacy of grouping data.

Although, K Means has high efficiency, various works in the past has shown that the genetic algorithm integrated with K Means performed than that of traditional K Means. Evolutionary computation based approached have been playing central role because of their capability of exploring the search space and escaping local minima. The first GA based K Means were proposed by Krishna K Murthy in 1999 and now there have been plenty of works related to it. Currently, there have been lots of work on this field. Among them, a number of approaches combine the ability of K-means in portioning data with that of genetic algorithms of performing adaptive search process to find near optimal solutions for an optimization problem.

This paper consists of a genetic algorithm which integrates the local search K-means principle into genetic operators of crossover and method. The method evolves a population of chromosomes representing a division of objects in a number of clusters. Each chromosome length is dependent on the number of features. Chromosome length is given by the multiplication of maximum number of cluster and number of features. Ranking based selection mechanism is used and tested on two datasets taken from Kaggle (Iris Dataset and Wine Data Clustering). The iris dataset has only 4 features while Wine Dataset has 13 features. The main goal of this project and the reason behind the selection of Wine Dataset is to test on the data with high number of features. The paper we presented in the class and the other few methods didn't test on the data with large number of features.

1.1 K-Means Clustering

K Means Clustering is the most popular and widely used clustering algorithm. The main objective of K means clustering problem is to partition data objects into a number of clusters such that both within cluster similarity and between cluster heterogeneity are high.

The K-means method is a partitional clustering algorithm that groups a set of objects into k clusters by optimizing a criterion function. The technique performs three main steps (1) selection of k objects as cluster centroids, (2) assignment of objects into the closest cluster, (3) updating of centroids on the base of the assigned data. Steps 2 and 3 are repeated until no object changes its membership cluster or the criterion function does not improve for a number of iterations. The K-means method finds a partition that minimizes the total within cluster variance which is known as sum of squared error, given by:

$$CV_W = \sum_{i=1}^{k} \sum_{x \in C_i} ||x - m_i||^2$$

Where, m_i is the centroid of the cluster C_i and ||.|| denotes the Euclidean distance.

1.2 K-Means Based Genetic Algorithms

Genetic algorithms are randomized search and optimization techniques guided by the principle of evolution and natural genetics, having a large amount of implicit parallelism[3]. GA performs search in complex, large and multimodal landscapes and provide near optimal solution for objective or fitness function of an optimization problem. In genetic algorithms, the parameters of the search space are encoded in the form of strings called chromosome. A collection of such string is called population. Initially, a random population is created which represents the different points in the search space. An objective and fitness function is associated with each string that represent the degree of goodness of the string[2]. Based on the principle of survival of the fittest, a few of the strings are selected and each is assigned a number of copies that go into the mating pool. Biologically inspired operators like crossover and mutation are applied on these strings to yield a new generation of strings. The process of selection, crossover and mutation continues for a fixed number of generation or till a termination condition is satisfies.

In past two decades, lots of methods that integrated Genetic Algorithms and K-means in several different ways have been proposed. The first proposal was by Krishna and Murty in 1999 which hybridizes a genetic algorithm with the K-means technique and named GKA method. This method adopts the integer encoding scheme where a chromosome is a vector of n elements, i.e. the number of the data objects and each gene has value in the alphabet {1,…..,k}, with k the fixed number of clusters to find. At the beginning each gene receives a random number between 1 and k. KMO is used to improve the convergence. The fitness function minimizes the within-cluster variance. Inspired by this, other different methods have been proposed such as FGKA which is different than GKA by using different mutation operator. Similarly, Banyopadhayay and Maulik presented a method named KGA which uses the principle of K-Means. One of the latest methods is CluGA method which we presented during the paper presentation uses similar approaches but the crossover technique was different. This method used group-based crossover which is followed by the application of the K-means operator (KMO)[1]. This paper mentions that in group-based crossover standard one- and two-point crossover has some drawbacks while cluster-oriented crossover KMO is used to avoid these drawbacks. The CluGA method can divide a dataset into several clusters without prior knowledge of the number of the clusters. The results shows that the CluGA method outperformed traditional k means testing in all datasets thus making it faster and better in clustering problems[1]. The common among all these methods is that they use similar fitness functions on all three papers. These papers have used Silhouette Index, Davis Bouldin Index and Calinski Harabasz Index as fitness function. They show the comparative results on each of the fitness function for different datasets. Out of these three fitness functions, the results have shown that Davis Bouldin Index and the Silhouette Index performed better than that of Calinski Harabasz Index.

One of the drawbacks of the traditional K-means clustering is that the number of clusters must be pre assigned. Evolutionary computation based clustering methods has the capability of exploring the search space and escaping the local minima during the optimization process[1]. It can find the solutions having the exact number of clusters without any prior knowledge with high values of evaluation indexes used to assess the performance of the method.

2 Our Method for the project

We also implemented a K-means algorithm similar to the method discussed above with the variation in operators such as Selection, Crossover and Mutation. Our initial goal was to implement a K-means integrated with genetic algorithm which would be able to cluster data with relatively high number of features. We have to change our approaches to achieve our goal which wasn't discussed during the project proposal. In our method there are N dimensional feature space, each chromosome is represented by a sequence of N*K floating point numbers where the first N positions or genes represent the N dimension of the first cluster center, the next N positions represent those of the second cluster center and so on. The sum of the Euclidean distances of the points from their respective cluster centers is taken as the clustering metric. The goal is to search for the appropriate cluster such that the clustering metric is minimized.

Our project also uses similar method and very close approaches to the previous works. The basic flow of our project is given by the following diagram: Initializing Population, Fitness Function, Selection, Crossover and Mutation.

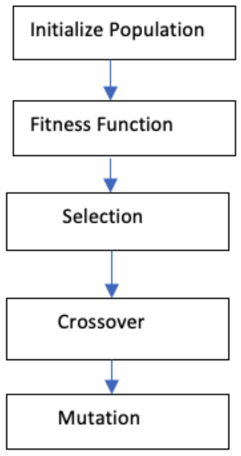


Fig: Flowchart of our method GA-K means

2.1 Steps in Our Method

```
Begin
    1. t=0
    2. initialize population P(t)
    3. compute fitness P(t)
    4. t=t+1
    5. if termination criterion achieved go to step (10)
    6. select P(t) from P(t-1)
    7. crossover P(t)
    8. mutate P(t)
    9. go to step 3
    10. Output best and step
End
```

2.2 Population Initialization

The chromosomes are generated randomly based on the length of the chromosome. The length of the chromosome is given by:

Chromosome length= k_{max}* no. of features

The length of the chromosome is the multiplication of maximum number of clusters and the number of features in the dataset. The approach of deciding the length of the chromosome in this way is for the algorithm to perform better with both the high and low number of features. Since, we are applying this algorithm into two datasets with two different number of features, we predicted this approach would be better for generating chromosome.

Once the length of the chromosome is fixed, we generated chromosome randomly based on their length. The K clusters centers encoded in each chromosome are initialized to K randomly chosen points from the dataset. This process is repeated for each of the P chromosomes in the population, where P is the size of the population.

2.3 Fitness Function

It is always important to solve a GA based problem using appropriate fitness function. In above mentioned literatures there have been use of various of indexes to evaluate clustering results. These indexes have been mainly used for cluster evaluation, in order to choose the best clustering method and to determine the optimal number of clusters. These fitness functions are used to find the compactness and separation of the cluster[1]. Based on both compactness and separation, we tried using two different indexes as the fitness function for our project.

Firstly, we used Davis-Bouldin Index as a fitness function. Similarly, we also used Silhouette Index for calculating fitness. The comparative results of these fitness function is provided in the result section of the paper.

The Davis Bouldin Index quantifies the average separability of each cluster from its nearest counterpart. It does this by calculating the ratio of the within cluster variance to the separation/ distance between cluster centroids. If we fix the distance between clusters but make the cases within each cluster more spread out, the Davies-Bouldin Index get larger, Conversely, if we fix the within cluster variance but move the cluster farther apart from each other, the index will get smaller. In theory, the smaller the value the better the separation between the clusters.

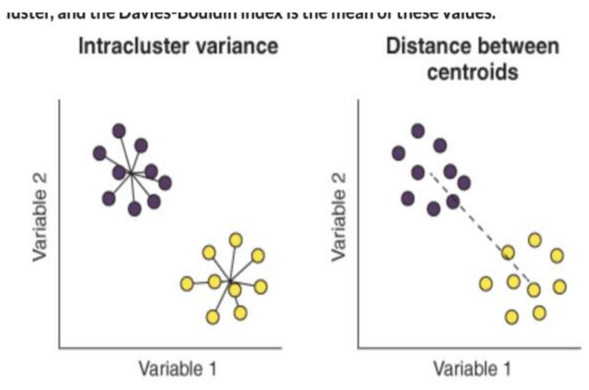

luster, and the Davies-Bouldin index is the mean of these values.
Intracluster variance
Distance between centroids
Variable 2
Variable 1
Variable 2
Variable 1

Fig: A figure showing Davis Bouldin Index in K means clustering

Since, Davis Bouldin Index is based on a ratio of within-cluster and between cluster distances. It is defined as:

$$DB = \frac{1}{k}\sum_{i=1}^{k} max_{i\neq j}\{\frac{\overline{d_i}+\overline{d_j}}{d_{i,j}}\}$$

Where d_i and d_j are the average distances between each object in the i-th and j-th cluster respectively and the centroid of the own cluster, and $d_{i,j}$ is the Euclidean distance between the centroids of the i-th and j-th clusters[5].

Similarly, the Silhouette index for each object measures how similar that object is to the elements in its own cluster, when compared to objects in other clusters[6]. The silhouette criterion is defined as:

$$S = \frac{1}{k}\sum_{i=1}^{k}\frac{1}{n_i}\sum_{x\in C_i}\frac{b(x)-a(x)}{max(a(x),b(x))}$$

Where a(x) is the average distance from x to the other objects in the same cluster as x, and b(x) is the minimum average distance from x to points in a different cluster minimized over all clusters.

While using the Silhouette analysis we are normally looking for a value of k that provides high average silhouette scores, number of clusters with a maximum silhouette score less than the average score as this can indicate poor clustering and a bad choice of k and well cluster data tends to have the clusters where the silhouettes are similar in size and the silhouettes of each cluster member are similar.

2.4 Selection

The selection process selects chromosomes from the mating pool directed by the survival of the fittest concept of the natural genetic systems[4]. The previous discussed method has used different selection methods such as proportional selection strategy, elitism selection method and so on. Our method uses Ranking based selection and also, we decided to test Tournament selection in order to check the possibility of improvement, however no improvement was noticed with the used of Tournament selection.

Ranking Selection sorts the population first according to fitness value and rank them[8]. Then every chromosome is allocated selection probability with respect to its rank. Individuals are selected as per their selection probability. Ranking selection is an explorative technique of selection. Below is the pseudocode of the algorithm we implemented, in order to sort the chromosome based on their fitness.

```
$ind$ ←No. of Individuals
$PS$ ←Prob of Selection
for $i = (ind * PS)$ do
    generation[ind-1-i]=generation[i]
    sort(generation)
end
```

Once the chromosomes are sorted, the selection probability passed as a parameter is provided to each chromosome based on their fitness. In this way, fit individuals are only selected and consider for further process such as crossover and mutation. Since, we considered the selection probability to be a hyperparameter, it varies depending on the data. We used 0.8 and 0.7 as the selection probability in Iris Dataset and Wine Dataset respectively.

2.5 Crossover

Crossover is a probabilistic process that exchanges information between two parent chromosomes for generating two child chromosomes[9]. The above discussed methods and previous other works have used one point crossover, two-point crossover and group-based strategy followed by K-means operator[1]. We used one point crossover for our project. We also tried using two-point crossover however no improvement was noticed. Thus, we stick with one point crossover.

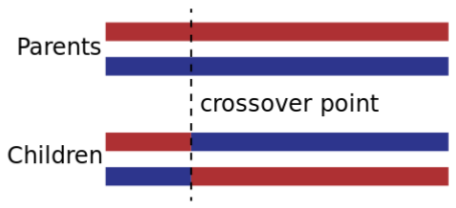


Fig: One point Crossover

In our method, single point crossover with a fixed probability of Pc is used. For a chromosome length l, a random integer called the crossover point is generated in the range [1, l-1]. The portions of the chromosomes lying to the right of the crossover point are exchanged to produce two offspring. We used crossover probability as a hyperparameter thus it would be different for different dataset. We have used 0.8 and 0.85 as crossover probability for Iris Dataset and Wine Dataset respectively.

2.6 Mutation

Simply, the mutation operator which performs a local search reassigns objects to the clusters at local distance. For each object, it checks the distance from the cluster to the point. Each chromosome undergoes mutation with a fixed mutation probability Pm. For binary representation of chromosome, a bit position (or gene) is mutated by simply flipping its value. Since we are considering floating point representation in this project. A number 'a' in the range [0,1] is generated with uniform distribution. The mutation is performed by comparing Pm with a. However, one problem with this form is that if the values at a particular position in all the chromosomes of a population become positive (or negative), then we will never be able to generate a new chromosome having a negative (or positive value) at that position. In order to overcome the limitation, we incorporated a factor of 2 while implementing mutation.

In our project we have used mutation probability as hyperparameter, thus we used Pm of 0.02 and 0.01 respectively for Iris Dataset and Wine Dataset.

3 Experiments and Results

We implemented this algorithm in Python. A trial-and-error procedure has been adopted for fixing parameter values. We had to keep changing Pc, Pm, Ps and population size again and again. These parameter values differ when used in different dataset.

3.1 Data Standardization

In the dataset, the value of the data has different range. It is very important for a model to have data in the range between 0 to 1. The standardization is done using following formula:

$$Z = X - mean / X + mean$$

Once the data is normalized the data is ready to be used for finding fitness value and performing other different operations.

3.2 Results

We used two datasets to check the performance of our implemented algorithm. In Iris dataset, our algorithm performed better as per our expectation, however, in wine dataset the results weren't quiet satisfying. Our initial guess at the beginning was that this might be because of the selection operator and other hyperparameters such as Ps,Pm, Pc and no. of population. We performed trial and error for multiple times for finding optimal number of population and other hyperparameters. And also, the number of features in the Wine Dataset was relatively higher than that of Iris data which cause wine dataset to have slightly long chromosome.

After trying trial and error method and using different selection mechanism (Tournament selection), the results didn't get better. In the meantime, we scrutinize the fitness value of wine dataset in each iteration. It never got improved. Thus, we decided to perform a dimension reduction on the data. We used Principal Component Analysis to reduce the number of features in the data.

3.2 .1 Principal Component Analysis

PCA is a dimensionality reduction method that is often used to reduce the dimensionality of large datasets, by transforming a large set of variables into a smaller one that still contains the most of the information in the large set. Reducing the number of variables of a data set naturally comes at the expense of accuracy but the trick in dimensionality reduction is to trade a little accuracy for simplicity. Because smaller data sets are easier to explore and visualize and make analyzing data much easier and faster for machine learning algorithms without extraneous variables to process. Thus, the idea of PCA is to reduce the number of variables of a dataset while preserving as much information as possible.

Below is simple process for performing PCA
(1)Standardize the range of continuous initial variables
(2)Compute the covariance matrix to identify correlations
(3)Compute the eigenvectors and eigenvalues of the covariance matrix to identify principal components
(4)Create a feature vector to decide which principal components to keep

(5) Recast the data along the principal component axes

For our project we used, scikit learn's Principal Component Analysis for dimensionality reduction. The wine dataset with 13 feature was reduced to two features. This eases the process while implementing out algorithm in Wine dataset.

3.2 Results Continued …

The following table shows the results in respective datasets. We have used the accuracy as an indicator to determine the accuracy for each dataset.

Dataset	**Traditional K-means**	**Our method**
Iris	83.3 %	85.4%
Wine (without PCA)	-	62%
Wine (PCA)	81.5%	80.33%

From the above table, we can see that traditional K-means has the accuracy of 83.3%. Similarly, when the traditional K-means is applied on the wine dataset after the application of PCA, it achieves an accuracy of 81.5%. While our method performed better in Iris dataset with accuracy of 85.4%. Our method achieved an accuracy of 62% on the wine clustering without the application of PCA. When our method is tested on the Wine Dataset after the application of Principle Component Analysis, the accuracy went up from 62% to 80.33%.

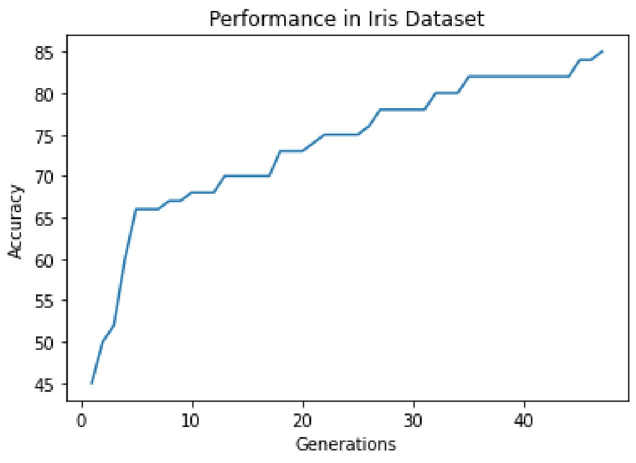
These above results of our method are the best we obtained when we used Single Point based Crossover, Ranking Selection Mechanism and Davis Bouldin Index as fitness function. We also tried using other operators such as Tournament Selection, two-point crossover and Silhouette index as fitness function. Our method didn't perform well when we used latter compared to earlier.

Talking about hyperparameters, we had to use trial and error in the same dataset and also for different dataset. Below is the table which shows the hyperparameters we have used for this project.

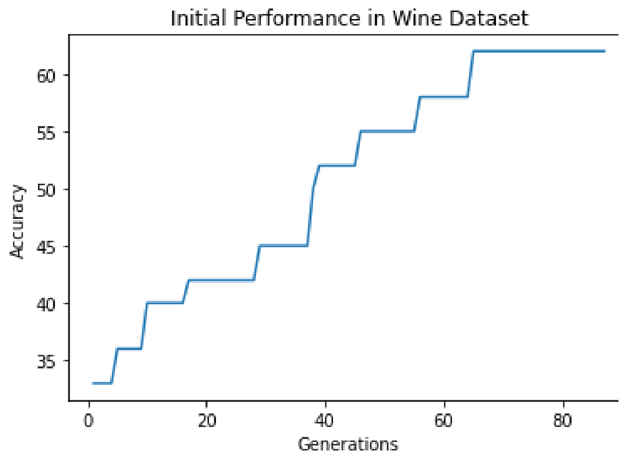
	Ps	Pc	Pm	Pop	Generation
Iris	0.8	0.8	0.02	20	50
Wine	0.7	0.85	0.01	25	100

The above table shows the hyperparameters used by us. We used 0.8 selection probability, 0.8 crossover probability, 0.02 mutation probability, 20 individuals and 50 generations for Iris dataset. Similarly, we used 0.7 selection probability, 0.85 crossover probability, 0.01 mutation probability, 25 individuals and 100 generations for wine dataset.

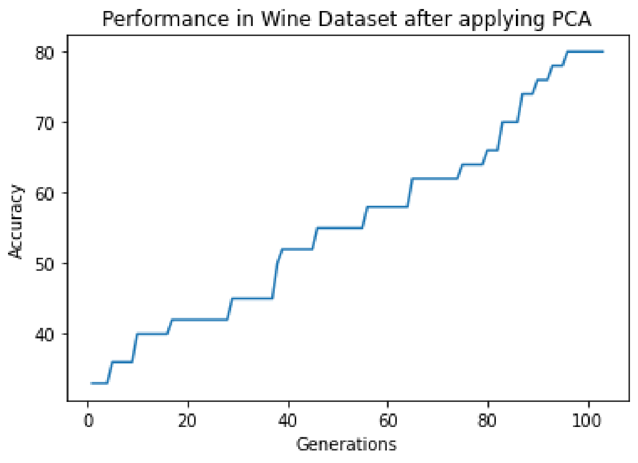
3.2 Visualizing Performance



In Iris dataset, we reached the optimal generations in 50 generations. For few earlier iterations, the iterations were below 50% but the accuracy gradually increases and reach to 85.4% in 50 generations. No improvement in accuracy was observed for 200 generations, thus we decided to show results up to 50 generations by considering it as an optimal solution.



Before applying Principal Component Analysis to the Wine Dataset, we applied our algorithm to the data with 13 features. Our method didn't do well. The above diagram clearly shows that the for almost 50 generations the accuracy was well below 50%. Finally, it increases slowly and reached 62% in 65 generations and never increased after that for more than 200 generations.



Our final approach was applying Principal Component analysis on the Wine dataset. This approach has better performance than the earlier method. The accuracy was better from the beginning iterations and finally increased up to 80 % after 100 generations. We observed there was no increment on the accuracy after 100 generation since not more than 80% accuracy was observed when we run for more than 200 generations.

3.3 Visualizing Clusters

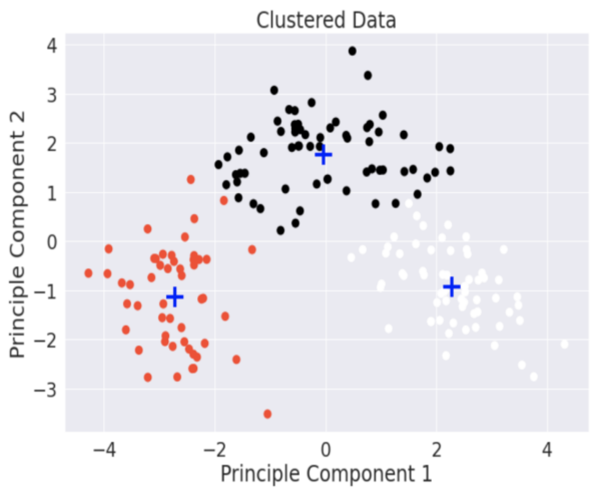


Fig: Figures showing centroid and the cluster for Wine data

We have captured centroid for each cluster along with the fitness function. The fitness function has very important role in keeping track of centroid.

The above image is final cluster we got after the implementation of our method on the wine data after PCA. The diagram shows the three centroids for the three clusters. The diagram clearly shows that our algorithm is able to classify the datapoints into the required three clusters.

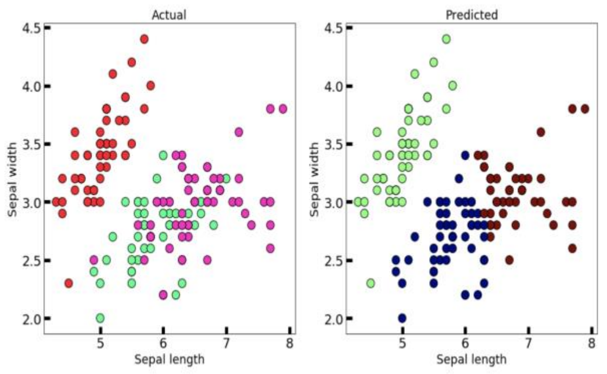


Fig: Figures showing centroid and the cluster for Iris data

The above figure is the actual cluster and predicted cluster by our method. The figure shows that the predicted cluster are very similar to the actual one. This shows that we are relatively doing great

4 Discussions

We observed the performance of our algorithm implemented for the object in two different datasets. Initially, it was tested on Iris data which has four features. Our implemented method performed better than the traditional K-means method. However, when we used the same algorithm with variations in hyperparameters, the proposed method performed poorly with the accuracy of 62% which is not great. We initially guessed that we need to change different parameters, selection method, crossover or mutation. As the number of features was relatively higher on the Wine dataset, the length of the chromosome got bigger. As the length of the chromosome got bigger, the population size has to be increased. Although, we increased the population size upto 100, no significant increment in the result was observed. Thus, we decided to use different approach i.e. reducing the dimension of the data by applying Principal Component Analysis.

As we applied Principal Component Analysis on the Wine dataset, the number of features reduced to two from thirteen. Thus, there was no need of increasing population size and of using different crossover operator, mutation operator and of using different selection mechanism. Therefore, we just performed heat and trial method for determining the hyperparameter for this approach which would perform better.

Apart from this, we made an important observation that if we can design a optimal operators for integrating K-means and Genetic algorithm, there would be no need of implementing Principal Component Analysis which would be more efficient. But the challenging thing about is we need to select the hyperparameters carefully and the other operators such as crossover, mutation, selection mechanism should be used efficiently.

5 Conclusion

We implemented a K-means algorithm with genetic algorithm. Although there have been various previous works related to it, our method is similar yet different in terms of the operators and mechanism used. Our initial goal of the project was to implement this method on the dataset with large number of features with reducing the dimensions of the data since the early works hasn’t mentioned much about the large number of features. Our method uses Ranking selection mechanism, single point crossover, Davis Bouldin Index as a fitness function which is used to determine the centroid of a cluster, a mutation mechanism.

We tested our proposed method on the two datasets Iris dataset and Wine Dataset. These two datasets are selected for testing because one is with a smaller number of features while other is with relatively large features. Since our initial goal was to implement our proposed method on the dataset with large number of features, these two datasets would be prefect for comparison. Our proposed method performed better on Iris dataset with the accuracy of 85.33%, outperforming the traditional K-means which had accuracy of 83.3%. While the wine dataset with 13 features performed poorly with our proposed method. From different inspections, we decided to perform dimensionality reduction using Principal Component Analysis. The 13 features were reduced to 2 features on which our proposed method was tested. Our method achieved an accuracy of 80% while the traditional K-means achieved an accuracy of 81%. Although, our method didn’t surpass the traditional K-means but the results were close. Therefore, our proposed method can be used for clustering in data with low number of features and also the dataset with relatively large number of features by reducing the dimension of the data with the application of Principal Component Analysis.

REFERENCES

[1] Pizzuti C.,Procopio N. (2017) *A K-means Based Genetic Algorithm for Data Clustering*. In: Grana M.,LopezGuede j., Extaniz O., Herrero A., Quintian H., Corchado E. (eds) International Joint Conference SOCO’16- CISIS ’16- ICEUTE’16. SOCO 2016, CISIS 2016, ICEUTE 2016. Advances in Intelligent Systems and Computing, vol 527. Springer, Cham

[2] Arthur, D., Vassilvitskii, S.: K-means++: the advantages of careful seeding. In: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, pp. 1027–1035 (2007)

[3] Bandyopadhyay, S., Maulik, U.: An evolutionary technique based on k-means algorithm for optimal clustering in rn. Inf. Sci. Appl. 146(1–4), 221– 237 (2002) 222 C. Pizzuti and N. Procopio

[4] Bandyopadhyay, S., Maulik, U.: Genetic clustering for automatic evolution of clusters and application to image

classification. Pattern Rec. 35, 1197–1208 (2004)
[5] Calinski, T., Harabasz, J.: A dendrite method for cluster analysis. Commun. Stat. 3(1), 1–27 (1974)
[6] Davies, D., Bouldin, D.: A cluster separation measure. IEEE Trans. Pattern Anal. Mach. Intell. 1(2), 224–227 (1979)
[7] Falkenauer, E.: Genetic Algorithms and Grouping Problems. Wiley, New York (1998)
[8]. Krishna, K., Murty, M.N.: Genetic k-means algorithm. IEEE Trans. Syst. Man Cybern. Part B 29(3), 433–439 (1999)
[9]. Liu, Y., Li, Z., Xiong, H., Gao, X., Wu, J.: Understanding of internal clustering validation measures. In: Proceedings of the 2010 IEEE International Conference on Data Mining, ICDM 2010, pp. 911–916 (2010)