CLUSTERING WITH UNKNOWN NUMBER OF CLUSTERS: A COMPARATIVE STUDY OF THREE METHODS

FINAL REPORT

Ashutosh Sanan, Athar Roshandelpoor and Sadra Sadraddini

Boston University
Department of Electrical and Computer Engineering
8 Saint Mary's Street
Boston, MA 02215
www.bu.edu/ece

2 May

Technical Report No. ECE-2016

1 Introduction

Clustering is one of the most important tools in unsupervised learning problems. It deals with grouping things into natural categories when no class label is available. The goal of clustering is to categorized data into clusters in a way that, objects in one cluster are similar to each other in some way and dissimilar to objects belonging to other clusters. In this project we consider problem of clustering data without prior knowledge of the number of clusters. We investigate the performance of three clustering methods on different datasets and compare their performance. The first method that we consider is k-means and DP-means. We apply k-means with different number of clusters, however, we add a penalty of number of clusters on the objective function of k-means. Second we consider DBSCAN method which clusters data base on density and as third method, we work on regularization of k-means. We apply these methods on different datasets such as Boston Airbnb data and Earthquake data and compare performance of these methods. We use silhouette function and Dunn index value as a metric for comparing performance of our algorithms.

Related Work

Clustering paradigms have a long history, dating back to Aristotle. The authors in [1, 6] have extensively surveyed the broad range of clustering algorithms and their history. In this project, we focus on more common methods. K-Means algorithm, first proposed in [10] and heavily influenced by the nearest neighbor classification algorithm, is perhaps the most prevalent method in data clustering. However usually symmetric clusters are formed. For instance, using Euclidian metric, the resulting clusters are spheres or using Mahalanobis distance as the underlying metric, ellipsoidal clusters are found. DP-means is a recent nonparametric extension to K-Means algorithm that is introduced in [7]. Hierarchical approaches [2], are based on organizing the data set into a hierarchy but its quadratic complexity and termination conditions have prohibited its application [6]. More recently, the authors in [10] provided a method called "Density Based Spatial Clustering Applications with Noise" (DBSCAN), which is rapidly gaining popularity in the machine learning community. DBSCAN, due to its nature based on the density of the data, is able to deal with large data sets and non-flat geometry.

2 Problem Formulation and Approach

Informally, we aim to investigate the following problem in this project:

Problem 1 *Given a set of data X, find an appropriate integer k and a set of k clusters such that the there exists a type of homogeneity among the data within in a cluster and a type of heterogeneity among the data across different clusters.*

The biggest challenge in this framework is determining the number of clusters. In this project, we revisit the k-means algorithm via two distinct approaches. In the first one, we follow a top-bottom approach by initially assuming a single cluster and adding one if a regularized performance metric is improved. In the second, we follow a bottom-up approach of assuming as many clusters as the number of data samples, and merging sufficiently close clusters. We also investigate the DBSCAN algorithm, where the number of clusters is explicitly determined by the user but is a function of tuning parameters.

Clustering Performance

For characterizing the quality of clusters we use standard performance metrics. The silhouette function for a data point $x \in X$ is defined as follows [9]:

$$s(x) = \frac{b(x) - a(x)}{\max\{a(x), b(x)\}}, \tag{1}$$

where $a(x)$ is the average dissimilarity to the other points of the same cluster and $b(x)$ is the lowest average dissimilarity to the points in other clusters. The silhouette function provides a number in the $[-1, 1]$ interval where 1(-1) stands for the best(worst) clustering. Usually the average silhouette function of the data set is reported as the clustering performance. Another useful clustering performance metric is provided by Dunn Index, which is [3]:

$$DI = \frac{\min\limits_{i \neq j} \delta_{i,j}}{\max\limits_{i} \Delta_i}, \tag{2}$$

where $\delta_{i,j}$ is the inter-cluster distance between cluster i and j and Δ_i is the distance within a cluster. Intuitively, larger Dunn index indicates better clustering.

3 Revisiting K-Means

K-means in one of the most famous clustering algorithm. It is really straight forward to implement and work well on the well-known clustering problems. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed a priori. The first step is define k centroids for each clusters and then assign data to the cluster which has the nearest centroid. Then it updates the center of clusters by simply averaging the points in that clusters and continue this until convergence. Here, convergence means no changes in assigning data to clusters. This algorithm aims to minimize the following objective function:

$$\sum_{j=1}^{k} \sum_{i \in C_j} \|x_i - \mu_j\|^2 \tag{3}$$

The procedure is minimizing the objective function in each step and also it is bounded from below, so it will converge. However, there is no guarantee that it terminates to the global minimum and the algorithm is really depends on the initial value for clusters centroid. In Fig. 1 we can see an example of the k-means method on some data.

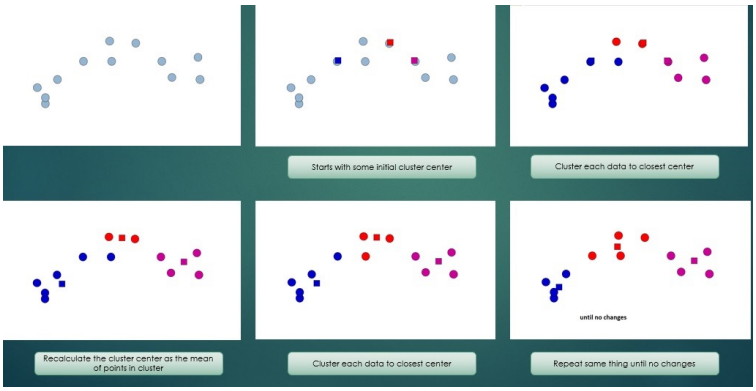

Figure 1: Example of parking lot image used in experiments

3.1 Penalizing the Number of Clusters

In our project, we add a penalty term on number of clusters to the objective function of k-means and apply k-means on our data set with different number of clusters. Then, we pick the cluster number which gives us the minimum objective function. In this method the objective function that we tried to minimize is:

$$\sum_{j=1}^{k} \sum_{i \in C_j} \|x_i - \mu_j\|^2 + \lambda k \tag{4}$$

This method is computationally very expensive because we have to try k-means several times. As a result, we look at another DP-means which is similar to this method but less computationally expensive.

3.2 DP-means Algorithm [7]

This algorithm works like k-means, except the number of clusters is not fixed at starting point and a new cluster will be added when the distance of a point to the nearest cluster centroid is greater than some threshold λ. The procedure is shown in Algorithm in Figure 2:

DP-means algorithm tries to minimize the following objective function:

$$\sum_{j=1}^{k} \sum_{i \in C_j} \|x_i - \mu_j\|^2 + \lambda k, \tag{5}$$

Algorithm 1 DP-means

Input: $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$: input data, λ : cluster penalty parameter
Output: Clustering $\ell_1, ..., \ell_k$ and number of clusters k

1. Init. $k = 1, \ell_1 = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$ and $\boldsymbol{\mu}_1$ the global mean.
2. Init. cluster indicators $z_i = 1$ for all $i = 1, ..., n$.
3. Repeat until convergence
   - For each point $\boldsymbol{x}_i$
     - Compute $d_{ic} = \|\boldsymbol{x}_i - \boldsymbol{\mu}_c\|^2$ for $c = 1, ..., k$
     - If $\min_c d_{ic} > \lambda$, set $k = k + 1$, $z_i = k$, and $\boldsymbol{\mu}_k = \boldsymbol{x}_i$.
     - Otherwise, set $z_i = \text{argmin}_c d_{ic}$.
   - Generate clusters $\ell_1, ..., \ell_k$ based on $z_1, ..., z_k$: $\ell_j = \{\boldsymbol{x}_i \mid z_i = j\}$.
   - For each cluster ℓ_j, compute $\boldsymbol{\mu}_j = \frac{1}{|\ell_j|} \sum_{\boldsymbol{x} \in \ell_j} \boldsymbol{x}$.

Figure 2: DP-Means Algorithm [7]

which is the same objective function as in regularized k-means. The DP-means algorithm monotonically decreases the target objective function until local convergence. The reason is that the reassignment step in the algorithm is definitely a non-increasing step. For each point it checks the squared distance of that point to the nearest cluster centroid, if it was greater than some threshold λ, that point will be a new cluster center and then add a penalty term λ to our objective function. As a results it subtracts a number greater than or equal to λ from the objective function and add λ to the objective function. So the objective function will not increase in this step. Also, mean update step also is a non-increasing step because, mean is the best representative point for the cluster center in terms of squared Euclidean distance.

Issues of the DP-means Algorithm

The problem with the algorithm is that, it is highly sensitive to the initialization and the starting point. We tried the algorithm on the artificial data that contains random number in different intervals. Then we tried the algorithm several times after that with the same structure data but use some other random number in same interval. we got totally different results. The results are shown in Fig. 3.

The second issue is that, the clustering method uses a fixed radius. This could split a cluster into multiple ones that are near-by. So it does not satisfy one of the main goal of clustering which is different clusters should be dissimilar in some way.

4 Sum-of-Norms (SON) Clustering

K-means is an unsupervised learning technique in which we partition the data into k clusters, based on their statistical features. Each cluster is represented by a centroid which is calculated from the empirical data given. Each point is assigned the cluster

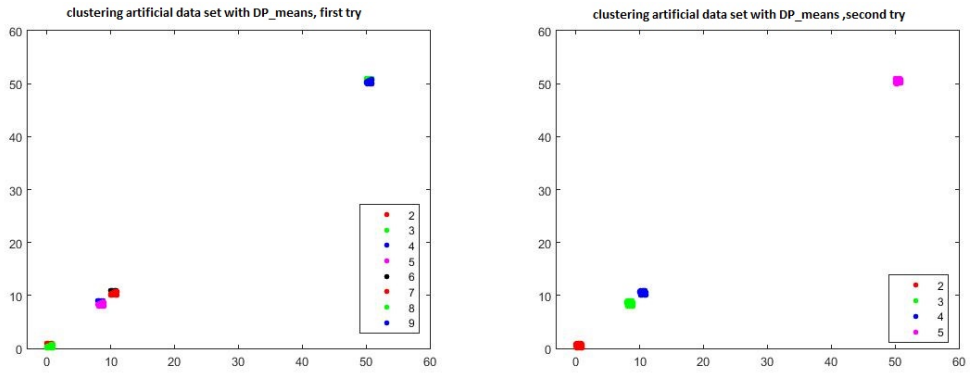

Figure 3: These data is generated randomly in the $[0, 1]$ interval and then copied and shifted so that 4 clusters with diameter 1 are formed. However, it is observed that the algorithm does not provide satisfactory results.

whose centroid is the closest. The goal is to minimize the intra-cluster variance or the sum of squares of distances between data and the clusters.

Although k-means clustering is fast and effective in many real world applications, it does have two big drawbacks: 1) its performance is highly dependent upon the initialization of the algorithm and 2) for certain problems the algorithm takes a lot of time to converge.

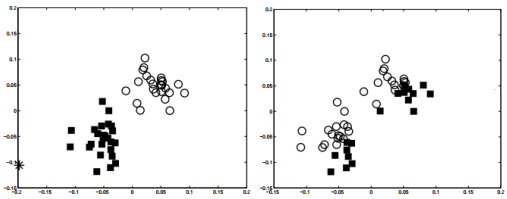

Figure 4: Sensitivity of k-means for initialization condition.

The Fig. 4 mentions sensitivity to initialization for k-means which is due to the non-convex optimization problem underlying the technique. SON clustering [8] is a convexification of k-means algorithm in which the objective function of the original technique is changed to become a convex optimization problem. This is done by relating k-means objective in terms of SON clustering. To modify the objective function lets consider a new formulation for clustering as follows:

$$\min_{\mu} \sum_{j=1}^{N} ||x_j - \mu_j||^2, \tag{6}$$

s.t $\mu_1, ..., \mu_N$ contains k unique vectors.

Therefore instead of fixing the number of clusters beforehand, we see a new approach in which we can automatically find out how many unique clusters are there.

The basic idea behind this is to start with N clusters which is equivalent to the number of samples in the data-set and then reduce the number of clusters such that the optimization condition is satisfied. We say that two data points belong to the same cluster if they have same centroid.

But there is still one huge problem with the above solution, the above solution in its current form will give N clusters no matter what which will be due to overfitting as we have given no bound on the maximum number of clusters. So to overcome that we introduce a regularization term to penalize the number of clusters in the problem:

$$\min_{\mu} \sum_{j=1}^{n} \|x_j - \mu_j\|^2 + \lambda \sum_{j=1}^{n} \sum_{j<i}^{n} \|x_i - \mu_j\|_p$$

Now the above equation is the new objective function which is a convex optimization problem contrary to the original non-convex problem. Now this problem can easily be solved using any convex optimization package.

There are two key benefits of this modification of the cost functions: 1. Now the above problem is convex and we can easily found a global minima, independent of the initialization. 2. Also this algorithm does not require the user to specify the number of clusters beforehand.

5 DBSCAN

DBSCAN (Density Based Spatial Clustering of Applications with Noise) is a relatively new algorithm that has drew significant attention in recent years [4]. In brief, DBSCAN is simply based on an ϵ-neighborhood of a data point and a density parameter γ. If the number of the data points within the ϵ-neighborhood of a point is more than γ, then the data point becomes a *core point* and a cluster is formed consisting of all the neighborhood points. Otherwise, the point is labeled as noise. If two core points are in the same neighborhood, the clusters are merged. Therefore, any two points in a cluster are reachable via a chain of core points. The pseudocode for DBSCAN algorithm is shown in Algorithm 1. DBSCAN does not require the number of clusters to be priorly determined. Also, as one of its remarkable advantages, it is able to generate irregular cluster shapes and deal with non-flat geometry.

6 Airbnb data Set

6.1 Regularized k-means

We apply k-means with different number of clusters on Boston Airbnb data.

We computed the objective function for different values of λ in the $[0, 20]$ range. As we can see in Fig. 6 when λ is zero, by increasing the number of clusters, the objective function decreases. On the other hand, for large value of λ, by increasing the number of clusters the objective function first decreases and then increases. This

Algorithm 1 DBSCAN Clustering Algorithm

Require: Data Set D, ϵ, γ.

$C = 0$ ▷ No cluster initialized
for Each Unvisited Point P in D **do**
    Mark P as visited, $\mathcal{N} = \{\text{points in } \epsilon\text{-neighborhood of } P\}$
    if If size$(\mathcal{N}) < \gamma$ **then**
        Ignore P as noise
    else
        $C \leftarrow C + 1$, Add P to Cluster C,
        for Each Point P' in $\mathcal{N}$ **do**
            if P' is not visited **then**
                Mark P as visited, $\mathcal{N}' = \{\text{points in } \epsilon\text{-neighborhood of } P'\}$
                if If size$(\mathcal{N}') \geq \gamma$ **then**
                $\mathcal{N} \leftarrow \mathcal{N} \cup \mathcal{N}'$
                end if
            end if
            if If P' is not yet a member of any cluster **then**
                Add P' to cluster C.
            end if
        end for
    end if
end for
return Clusters

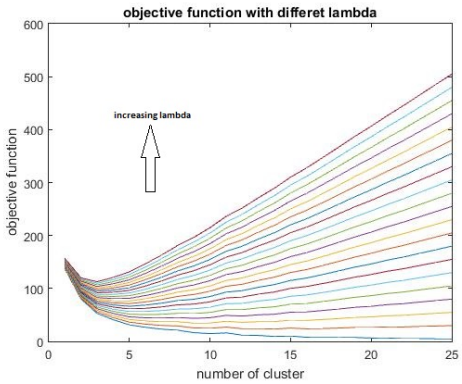

Figure 5: objective function vs. number of clusters for different value of λ

is because, when λ is large we should pay a large penalty for adding a new cluster. We choose $\lambda = 10$ and find the minimum value of the objective function at $k = 4$. We apply k-means with k=4 and got the silhouette number 0.511.

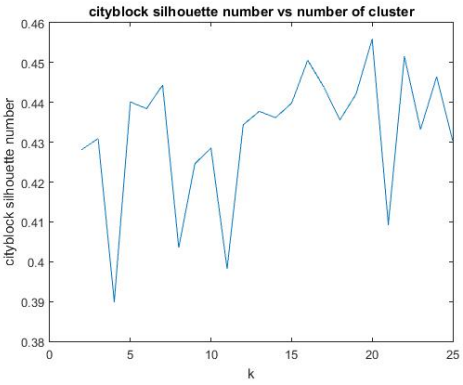

Figure 6: Boston Airbnb data Silhouette Number

It is shown in Fig. 6 that silhouette number does not show a monotonic behavior by increasing number of clusters. This is because having large or small number of clusters does not necessarily mean that data is clustered well. The good clustering is in a way that points in each cluster are similar to each other and dissimilar to the points of other clusters. The clusters are shown in Fig. 7.

The purity metric, which is shown in Fig. 8, is increasing vs the number of clusters. This is because when we increase the number of clusters, the data points in each cluster tend to be more homogenous and each data will correspond to a same ground truth label.

6.2 SON Clustering

We also tried the SON clustering method on this data set. We used a Matlab software for Convex Programming named CVX [5]. The code ran successfully and the outputs

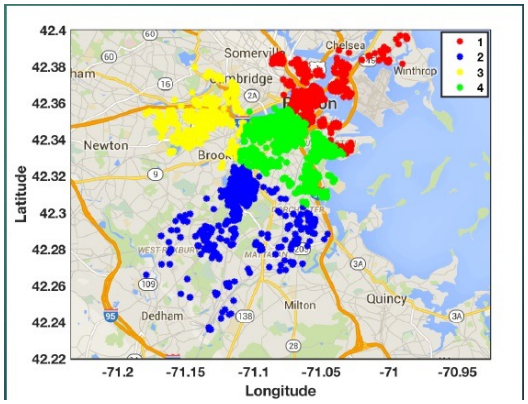

Figure 7: Boston Airbnb data with 4 clusters

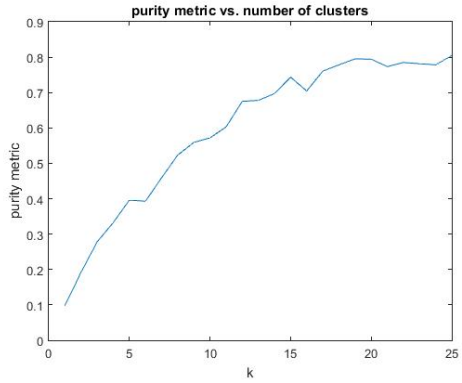

Figure 8: Purity metric vs number of clusters for Boston Airbnb data

seemed reasonable for many different datasets. Although the objective function has been modified to become computationally tractable, but it is a convex problem with a lot of constraints which requires a lot of computer resources like memory as well as processing power. Due to this problem we were only able to run the algorithm for a random subset of the data with max data points equal to 700.

As it can be seen in Fig. 9 the number of clusters is perfect with not a high merging in the data. Sets if parameters that we used for our model implementation was regularization parameter $\lambda = 0.001$.

6.3 DBSCAN

We also implemented DBSCAN on this data set. We varied the tuning parameters as illustrated in Fig. 10. It is observed that while the purity metric is maximum at low density values, such a tuning results in a very large number of clusters. The result with density parameters $\epsilon = 0.02$ and Minimum Points $\gamma = 10$ is shown in Fig. 11. As it can be observed, DBSCAN, which is not an optimization based algorithm, tends to cluster based on the sparsity and density of the data rather than a performance

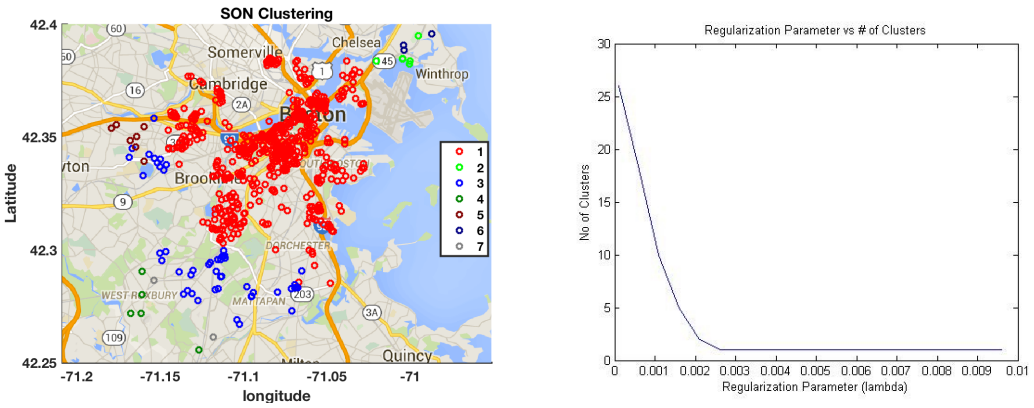

Figure 9: SON Clustering Results on AirBnB data set. (left) Clusters (right) As expected the number of clusters decrease as we increase the value of regularization parameter which is the penalty on the number of clusters.

metric.

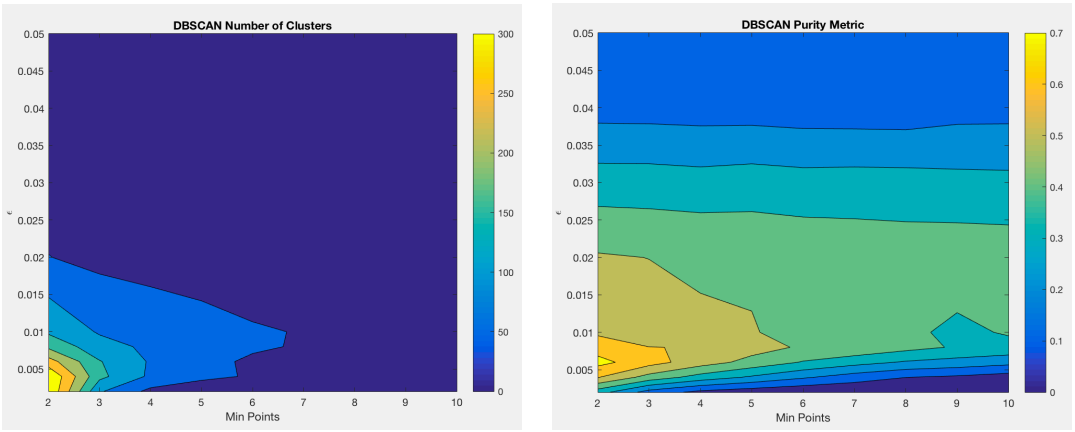

Figure 10: DBSCAN results on AirBnB data set: Number of Clusters (Left) and Purity Metric (Right) vs Tuning Parameters

7 Protein Docking

This data is derived from the professor Vajda's lab in BME Department in BU. This data is used for the problem of Protein Docking. The ultimate goal of protein docking is to predict the structure of a complex of two protein structure as it happens in nature. There is a benchmark for a number of protein-protein complexes with known complexed structures, which has been developed for testing docking methods. In this project we use a data for a complex 1acb. In protein docking one protein is fixed and in order to find the conformation of the complex, the other protein (see figure 12) is moved at each orientation such that the position of the protein in which the complex

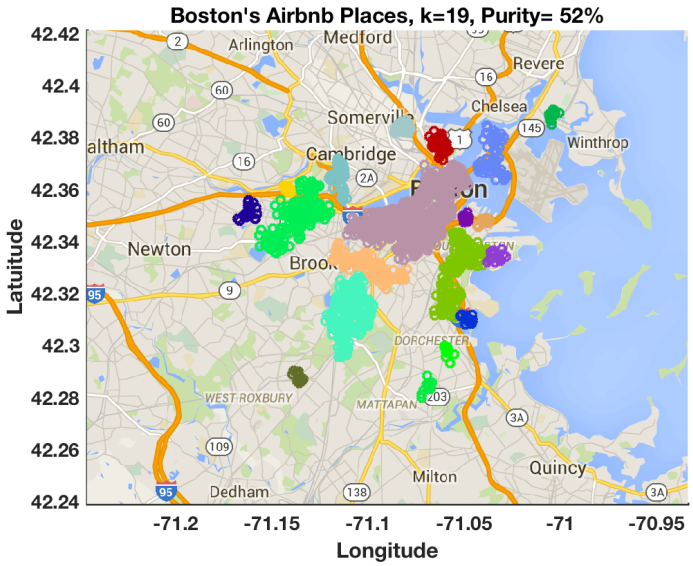

Figure 11: DBSCAN result with density parameters $\epsilon = 0.02$ and Minimum Points $\gamma = 10$ which yield 19 clusters.

has the minimum energy is found. The data file that we use has 1500 instances with 6 features. Each instance corresponds to a possible position of the second protein which is not fixed. The first three components correspond to the position of the center of the mass of the protein and the last three components correspond to the orientation of the protein. In computational biology and specifically protein docking, clustering actually plays an important role. The reason is that, events that are happened in a cluster are probably not random. Also, people who work in this area believe that the largest cluster could be the cluster of the actual position of the protein in nature (the data in the largest cluster is corresponding to the position of the second protein. Such a position is the most possible position of the second protein in the nature). In Fig. 13 we can see the conformation of two proteins. This data is from a benchmark and we know the actual position of the two proteins relative to each other. We explain the problem in figures below.

We cluster the data using DP-means and DBSCAN. DP-means provides relatively good clusters but the problem is that number of clusters is very large. Besides, for the problem of initialization we randomize the order of data in ten different ways and pick the one that gives us the minimum objective function. However, the answers are not similar to each other; also, we get different answer by all the way possible that we can randomize the order of data. This is a reason why DP-means is not an appropriate method for some kind of data especially large data. As it is shown in figure 14, the largest cluster of DP-means is exactly in the place that it should be (since we know the answer we can compare it with real position). For obtaining this result, we tuned $\lambda = 15$ and the resulting silhouette is 0.5413.

Also we apply DBSCAN on the same data. The largest cluster made by DBSCAN

is also near the position that it should actually be (see Fig. 15). However, as it is shown in Fig 16, DBSCAN considers a lot of points as noise which is not appropriate. The results for DBSCAN using $\epsilon = 3$, $\gamma = 6$ is 24 clusters and 80 points are marked as noise. As we can see, the number of DBSCAN is clusters is far smaller than the number of clusters in DP-means.

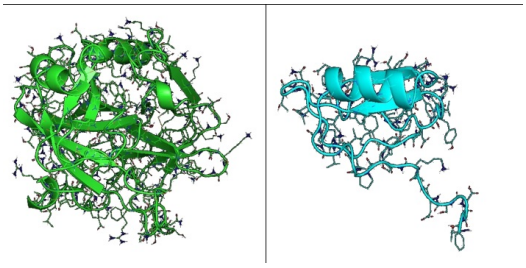

Figure 12: Green protein (left figure) is the fixed one and we move the blue protein(right picture) to find local minimum of the complex energy. Each data that we have is position of the blue protein relative to the green one.

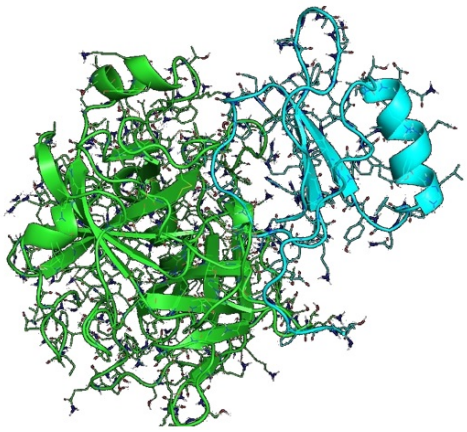

Figure 13: This is the actual conformation of the complex 1acb in nature.

8 Earthquake Data

We have implemented DBSCAN algorithm on an earthquake data set. The data set includes the longitude (l) and latitude (ϕ) of 2178 earthquakes [1], which is shown in Fig. 17. We use the geodesic distance between two points as the underlying metric, where the distance between two points is given by the following equation:

$$d_{i,j} = \arccos\left(\sin\phi_i \sin\phi_j + \cos\phi_i \cos\phi_j \cos(l_i - l_j)\right).$$

[1] `http://sci2s.ugr.es/keel/dataset.php?cod=1297`

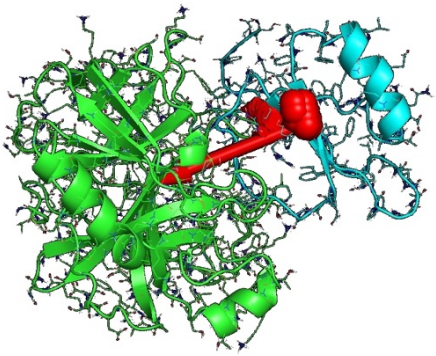

Figure 14: the largest cluster with DP-means method, as we can see the largest cluster is exactly in the place of the second protein. so clustering method works for our data.

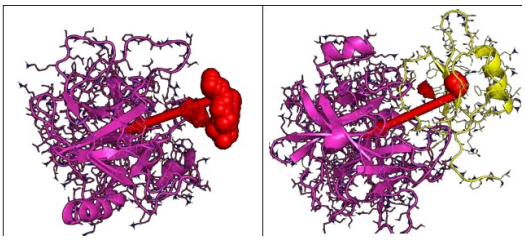

Figure 15: Largest cluster with DBSCAN, the largest cluster is in the exactly place that it should be.

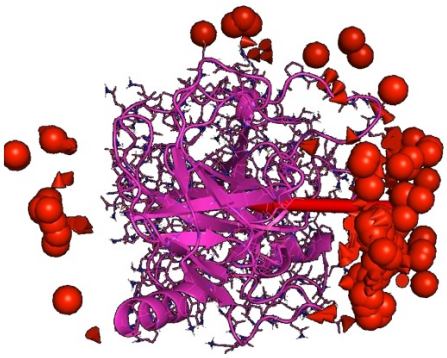

Figure 16: DPSCAN consider a lot of position as noise. Which is not good because in this way we miss a lot of good conformations.

The performance results for different density tuning parameters is show in Fig. 17. As it is observed, greater density parameters lead to smaller number of clusters, but many data points are thus regarded as noise when the parameter γ is increased. At the same time, the Dunn Index is improved with larger density requirements. This is a drawback of Dunn Index as it is not completely informative.

We finally chose the result for $\gamma = 3$ and $\epsilon = 5°$, which corresponds to roughly

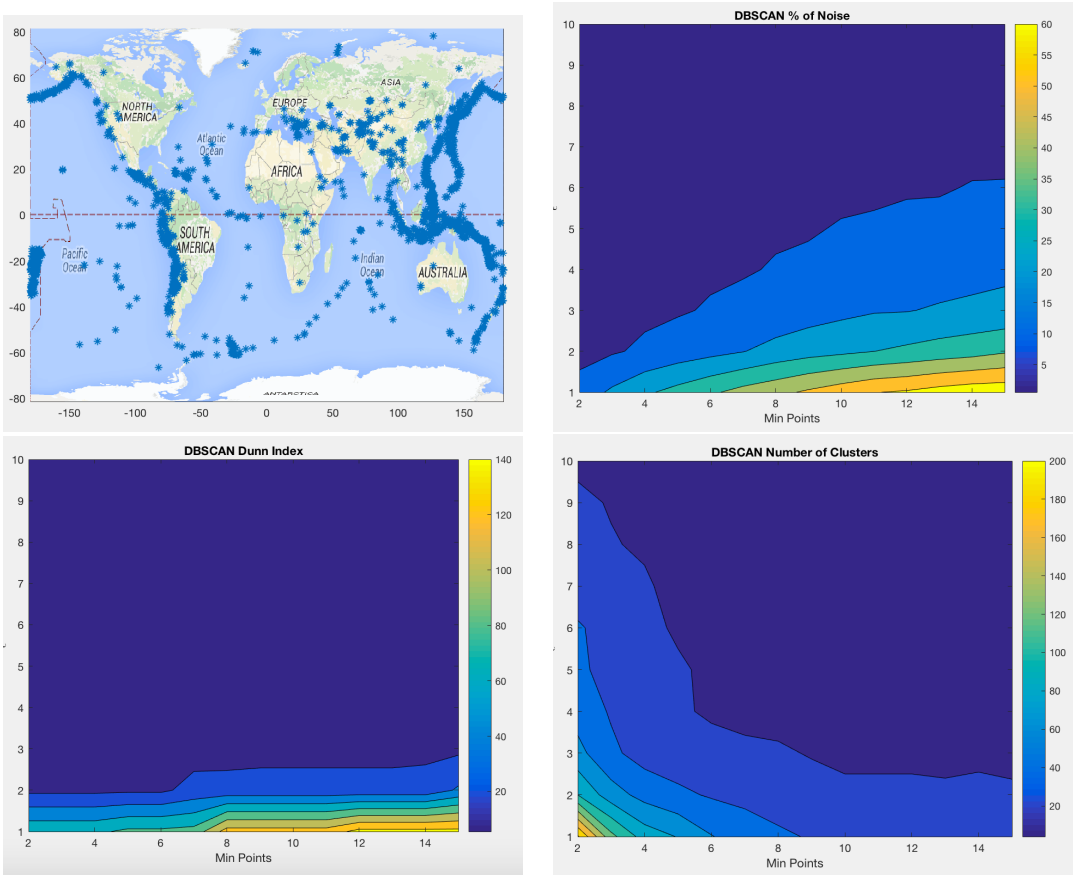

Figure 17: DBSCAN Performance on Earthquake Data.

550 km (340 miles) on the Earth surface, is illustrated in Fig. 18. It is seen that the irregular clusters are consistent with the tectonic plates [2]. We also compared our results with spectral clustering algorithms based on constructing graphs from the ϵ-neighborhoods, which its best Dunn Index performance is shown in Fig. 19. As it is observed, spectral clustering, which in this case can be interpreted as implementation of k-means on this non-flat geometry, demonstrates a poor performance on dividing earthquakes resulting from the same fault planes into different clusters.

9 Conclusion

In this work, we investigated several clustering algorithms where the number of clusters was not known in prior. We studied methods that automatically generate the number of clusters and also studied ways to regularize it. The implementation of k-means using different values of k is not computationally efficient. On the other hand, DP means which is an iterative algorithm is highly sensitive to the order of the

[2] `https://en.wikipedia.org/wiki/Plate-tectonics`

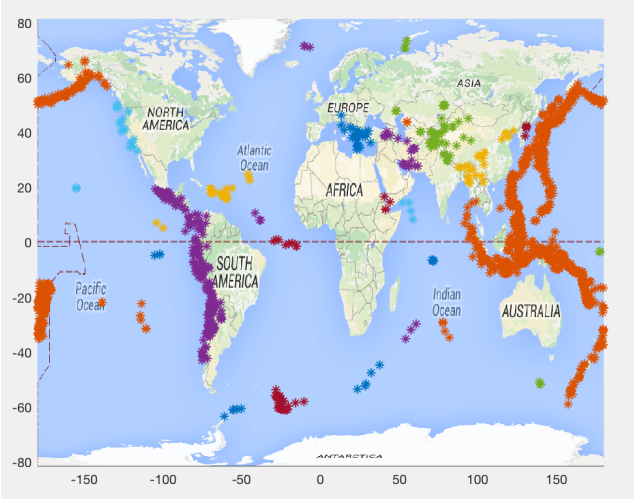

Figure 18: DBSCAN algorithm implemented on Earthquake data with 33 clusters formed.

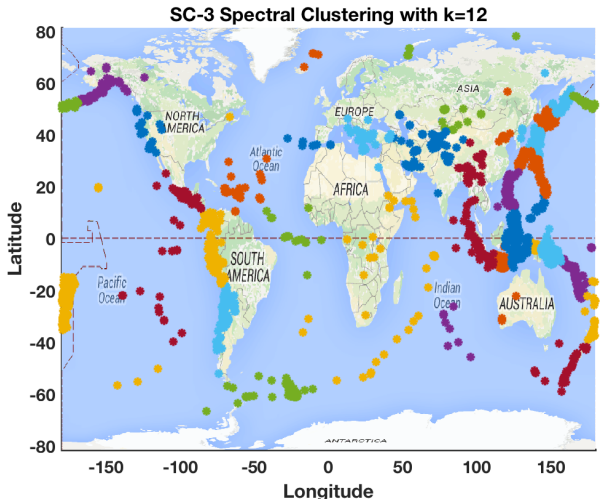

Figure 19: Spectral Clustering algorithm implemented on Earthquake data with 12 clusters formed.

data and also results in a very large number of clusters. Also while the convextifiation of the k-means algorithm is theoretically promising, the number of constraints it generates is very large hence there is not much computational benefit gained from this method. One of the conclusions we reached in this project is that almost all performance metrics are improved with respect to the number of clusters. Therefore, without penalizing the number of clusters, algorithms may tend to generate as many clusters as the number of data instances, which is a trivial optimal case. We also looked at density based algorithm which the number of clusters is dependent on the user defined density parameters. DBSCAN is also poor at identifying clusters with different densities and also can mark many points as noise.

References

[1] M. R. Anderberg, *Cluster analysis for applications: probability and mathematical statistics: a series of monographs and textbooks*, vol. 19. Academic press, 2014.

[2] F. Corpet, "Multiple sequence alignment with hierarchical clustering," *Nucleic acids research*, vol. 16, no. 22, pp. 10881–10890, 1988.

[3] J. C. Dunn, "A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters," 1973.

[4] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *Kdd*, vol. 96, pp. 226–231, 1996.

[5] M. Grant, S. Boyd, and Y. Ye, "Cvx: Matlab software for disciplined convex programming," 2008.

[6] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

[7] B. Kulis and M. I. Jordan, "Revisiting k-means: New algorithms via bayesian nonparametrics," *arXiv preprint arXiv:1111.0352*, 2011.

[8] F. Lindsten, H. Ohlsson, and L. Ljung, "Just relax and come clustering!: A convexification of k-means clustering," 2011.

[9] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[10] H. Steinhaus, "Sur la division des corp materiels en parties," *Bull. Acad. Polon. Sci*, vol. 1, pp. 801–804, 1956.