Genomic Data Mining

Lei Yu[1]

Abstract— **Gene expression data contain much information about gene. By analyzing gene expression matrix, we could get information about function of some genes. Data mining is a great and common technique to deal with gene expression data in biological analysis. Gene expression analysis has a wide variety of applications, including cancer studies. Comparing tumor samples with healthy samples through genomic data mining in this project help us predict if a new patient has a tumor. Through analyzing many kinds of techniques and methods, Fuzzy C-means clustering algorithm could provide a relatively better and more reasonable result.**

I. INTRODUCTION

Gene expression matrices[1] are pretty common in biological analysis. For example, if the expression vector of a newly sequenced gene is similar to the expression vector of a gene with known function, a biologist may suspect that these genes perform related functions. Also, genes with similar expression vectors may imply that the genes are co-regulated, meaning that their expression is controlled by the same transcription factor. This suggests a "guilt by association" strategy for inferring gene functions by starting from a few genes with known functions and potentially propagating the functions of these genes to other genes with similar expression vectors. Finally, gene expression analysis is important in biomedical studies such as analyzing tissues before and after a drug is administered or contrasting cancerous and non-cancerous cells. For example, expression analysis led to MammaPrint, a diagnostic test that determines the likelihood of breast cancer recurrence based on the expression analysis of 70 human genes associated with tumor activation and suppression.

II. DATA USED

In 1999, Uri Alon analyzed gene expression data for 2,000 genes from 40 colon tumor tissues and compared them with data from colon tissues belonging to 21 healthy individuals. We can represent his data as a 61 $\times$ 2,000 gene expression matrix, where the first 40 columns describe tumor samples and the last 21 columns describe normal samples.

Now, we performed a gene expression experiment with a colon sample from a new patient, corresponding to a 62nd row in an augmented gene expression matrix.

Our goal is to predict whether this patient has a colon tumor. Since the partition of tissues into two clusters (tumor vs. healthy) is known in advance, it may seem that classifying the sample from a new patient is easy. Indeed, since each patient corresponds to a point in 2,000-dimensional space, we can compute the center of gravity of these points for the tumor sample and for the healthy sample. Afterwards, we can simply check which of the two centers of gravity is closer to the new tissue.

Alternatively, we could perform a blind analysis, pretending that we do not already know the classification of samples into cancerous vs. healthy, and analyze the resulting 62 $\times$ 2,000 expression matrix to divide the 62 samples into two clusters. If we obtain a cluster consisting predominantly of cancer tissues, this cluster may help us diagnose colon cancer.

Given Alons 2,000 $\times$ 61 gene expression matrix and gene data from a new patient, derive a superior approach to evaluate whether this patient is likely to have a colon tumor

The data from Uri Alon, shown in Figure 1, contain 62 rows, among of them are 40 rows for cancer samples and 21 rows for healthy samples, and the last row is for unknown sample to be detected. Each row has 2,000 features about gene expression. And the data file is from UC San Diego online course.

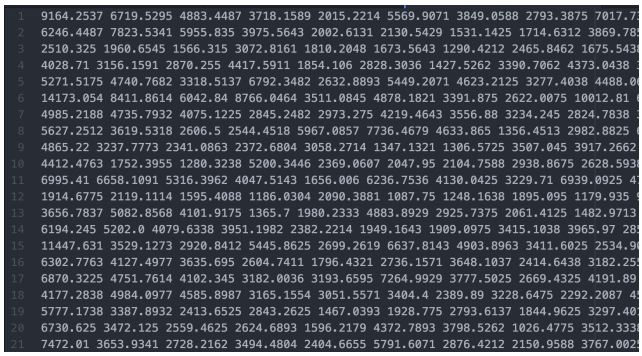

Fig. 1. Fragment of 61 $\times$ 2,000 gene expression matrix

III. TOOLS USED

The tool, which is a python code, is implemented by author. It applies Fuzzy C-means Clustering Algorithm on the gene expression data.

To identify groups of genes with similar expression patterns, we will think of an expression vector of length m as a point in m-dimensional space; genes with similar expression vectors will therefore form clusters of nearby points. Ideally, clusters should satisfy the following common-sense principle, which is illustrated in the figure below.

A. Good Clustering Principle

Every pair of points from the same cluster should be closer to each other than any pair of points from different clusters.

*This work was not supported by any organization

[1]Lei Yu is a graduate student with the Department of Computer Science, University of Nebraska at Lincoln yuleinku at gmail.com

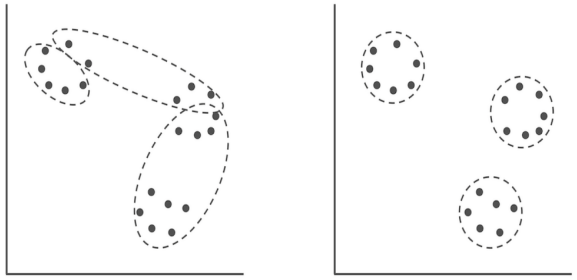

Fig. 2. (Left) A partition of twenty points into three clusters that do not satisfy the Good Clustering Principle. (Right) A different partition of these points that does satisfy the Good Clustering Principle.

B. *k-Center Clustering[2]*

Rather than thinking about clustering[8] as dividing data points Data into k clusters, we will instead try to select a set Centers of k points that will serve as the centers of these clusters. We would like to choose Centers so that they minimize some distance function between Centers and Data over all possible choices of centers.

We now define the distance between all data points Data and centers Centers. This distance, denoted as MaxDistance(Data, Centers), is the maximum of d(DataPoint, Centers) among all data points DataPoint, MaxDistance(Data, Centers) = max all points DataPoint from Data d(DataPoints,Centers).

Given a set of data points, find k centers minimizing the maximum distance between these data points and centers.

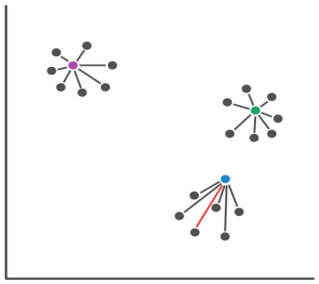

Fig. 3. The collection of points Data (shown as black points) along with three centers forming a set Centers (shown as colored points). For each point DataPoint in Data, d(DataPoint, Centers) is equal to the length of the segment connecting it to its nearest center. MaxDistance(Data, Centers) is equal to the length of the longest such segment, which is shown in red.

C. *Farthest First Traversal*

Although the k-Center Clustering Problem is easy to state, it is NP-Hard[5]. The Farthest First Traversal[3] heuristic, selects centers from the points in Data (instead of from all possible points in m-dimensional space). It begins by selecting an arbitrary point in Data as the first center and iteratively adds a new center as the point in Data that is farthest from the centers chosen so far, with ties broken arbitrarily.

- Weakness of Farthest First Traversal

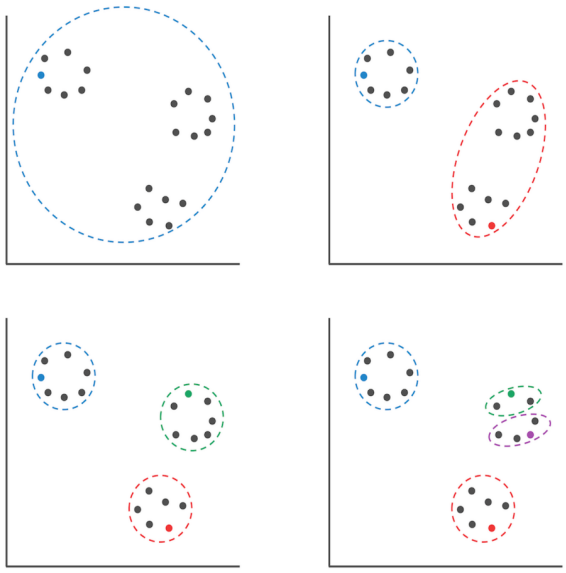

Fig. 4. Applying Farthest First Traversal. (Top left) An arbitrary point from the dataset (shown in blue) is selected as the first center. All points belong to a single cluster. (Top right) The red point is selected as the second center, since it is the farthest from the blue point. (Bottom left) After computing each data points minimum distance to each of the first two centers, we find that the point with the largest such distance is the green point, which becomes the third center. (Bottom right) The fourth center is shown in purple.

Farthest First Traversal is fast, and according to the preceding exercise, its solution approximates the optimal solution of the k-Center Clustering Problem; however, this algorithm is rarely used for gene expression analysis. In k-Center Clustering, we selected Centers so that these points would minimize MaxDistance(Data, Centers), the maximum distance between any point in Data and its nearest center. But biologists are usually interested in analyzing typical rather than maximum deviations, since the latter may correspond to outliers representing experimental errors.

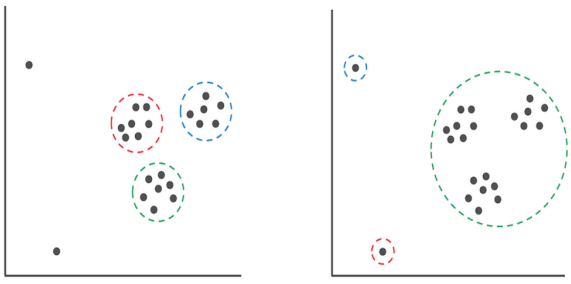

Fig. 5. (Left) A set of data points with three clearly seen clusters and two outliers. (Right) Because Farthest First Traversal relies on MaxDistance to compute new centers, if we attempt to cluster the data points into three clusters, then regardless of which point is selected as the first center, the two outliers on the left will be selected as centers of single-element clusters, with all remaining data points assigned to a single cluster.

D. *k-Means Clustering*

Note that whereas MaxDistance(Data, Centers) only accounts for the length of the single red segment in the Figure

3, the squared error distortion accounts for the length of all segments in this Figure 3.

To address limitations of MaxDistance, we will introduce a new scoring function. Given a set Data of n data points and a set Centers of k centers, the squared error distortion of Data and Centers, denoted Distortion(Data, Centers), is defined as the mean squared distance from each data point to its nearest center,

Distortion(Data,Centers) = (1/n) Sum of all points DataPoint in $Datad(DataPoint, Centers)^2$.

Given a set of data points, find k center points minimizing the squared error distortion.

The key difference between the k-Centers and k-Means Clustering Problems[7] is that in the latter, the placement of a center is far less affected by outliers, which is shown in Figure 6.

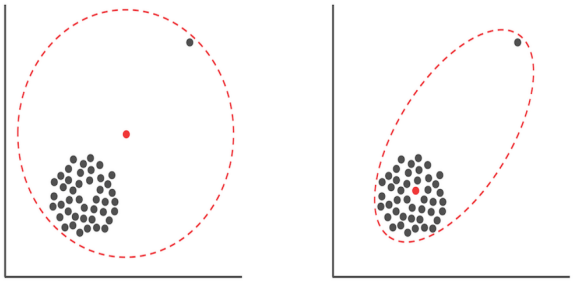

Fig. 6. Center placement varies in different clustering problem formulations. (Left) In the k-Center Clustering Problem, a clusters center is chosen so that the maximum distance between the center and any point in the cluster is minimized. As a result, the position of the center can be greatly influenced by outliers. (Right) In the k-Means Clustering Problem[9], the outliers influence over the placement of the center is much smaller. This is preferable when analyzing biological datasets, in which outliers often correspond to erroneous data.

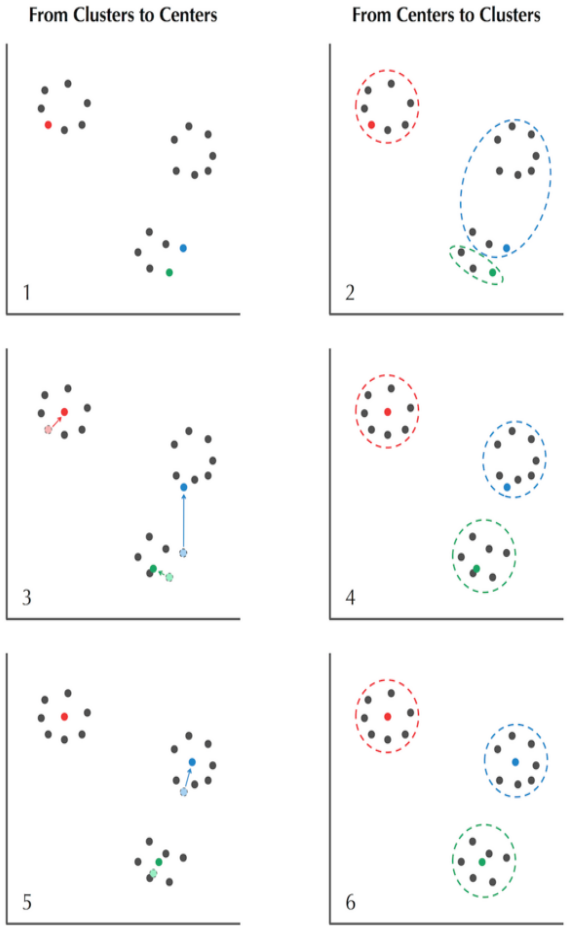

Fig. 7. The Lloyd algorithm in action for k = 3. In the top left panel, we select three arbitrary data points as centers, shown as differently colored points. In subsequent panels, we iterate the "Centers to Clusters" step followed by the "Clusters to Centers" step. In the bottom right panel, the Lloyd algorithm has converged.

E. Lloyd Algorithm

The Lloyd algorithm[4] is one of the most popular clustering heuristics for the k-Means Clustering Problem. It first chooses k arbitrary points Centers from Data as centers and then iteratively performs the following two steps:

- Centers to Clusters: After centers have been selected, assign each data point to the cluster corresponding to its nearest center; ties are broken arbitrarily.
- Clusters to Centers: After data points have been assigned to clusters, assign each clusters center of gravity to be the cluster's new center.

In the Figure 7, the centers appear to be moving less and less between iterations. We say that the Lloyd algorithm has converged if the centers (and therefore their clusters) stop changing between iterations.

- Weakness of Lloyd Algorithm's Initialization Step

The Figure 8 illustrates that things can go horribly wrong if we do not pay attention to the Lloyd algorithm's initialization step. In the top figure, we select no centers from clump 1, two centers from clump 3, and one center from each of clumps 2, 4, and 5. As shown in the bottom figure, after the first iteration of the Lloyd algorithm, all points in clumps 1 and 2 will be assigned to the red center, which will move approximately halfway between clumps 1 and 2. The two centers in clump 3 will divide the points in that clump into two clusters. And the centers in clumps 4 and 5 will move toward the middle of these clumps. The Lloyd algorithm will then quickly converge, resulting in an incorrect clustering.

F. k-means++ Initializer

We have thus far not paid much attention to how initial centers are chosen in the Lloyd algorithm, which selects them randomly. Similarly to Fartherst First Traversal, k-Means++ Initializer[10] picks k centers one at a time, but instead of choosing the point farthest from those picked so far, it chooses each point at random in such a way that distant points are more likely to be chosen than nearby points. Specifically, the probability of selecting a center DataPoint from Data is proportional to the squared distance of DataPoint from the centers already chosen, i.e., to d(DataPoint, Centers)2.

For a simple example, say that we have just three data

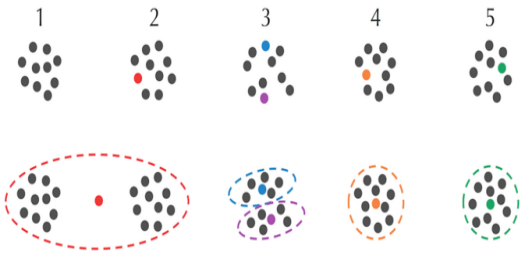

Fig. 8. (Top) Five clumps of ten points in two-dimensional space. The Lloyd algorithm is initialized so that clump 1 contains no centers, clump 3 contains two centers (blue and purple), and each of the other three clumps contains one center (red, orange, and green). (Bottom) Points are clustered according to the center to which they are assigned. The Lloyd algorithm has combined the points in clumps 1 and 2 into a single cluster and split clump 3 into two clusters.

points, and that the squared distances from these points to the existing centers Centers are equal to 1, 4, and 5. Then the probability of k-Means++Initializer selecting each of these points as the next center is 1/10, 4/10, and 5/10, respectively.

- Weakness of k-means Clustering

One weakness with our formulation of the k-means Clustering Problem is that it forces us to make a "hard" assignment of each point to only one cluster. This strategy makes little sense for midpoints, or points that are approximately equidistant from two centers, like the situation shown in Figure 9.

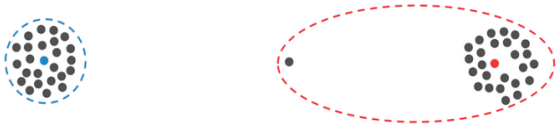

Fig. 9. The Lloyd algorithm rigidly assigns the midpoint between the two clusters to a single cluster, rather than giving it approximately equal membership in each cluster.

G. Fuzzy C-means Clustering[11]

To solve the limitation of k-means clustering algorithm, our goal is to transition away from a rigid assignment of a data point to a single cluster and toward a "soft" assignment, as shown in Figure 10.

We are now ready to use the expectation maximization algorithm[6] to modify the Lloyd algorithm into a Fuzzy C-means clustering algorithm. This algorithm starts from randomly chosen centers and iterates the following two steps:

- Centers to Soft Clusters: After centers have been selected, assign each data point a "responsibility" value for each cluster, where higher values correspond to stronger cluster membership.
- Soft Clusters to Centers: After data points have been assigned to soft clusters, compute new centers.

We begin with the "Centers to Soft Clusters" step. We have already used the term "center of gravity" when computing centers; if we think about the centers as stars and the data points as planets, then the closer a point is to a center, the

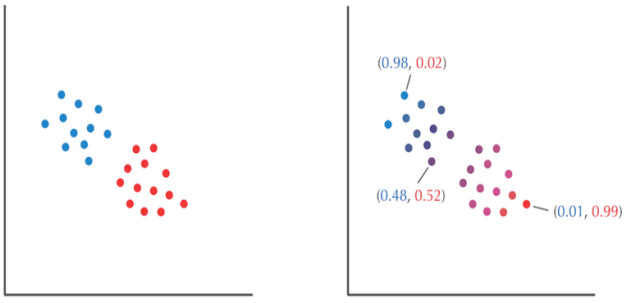

Fig. 10. (Left) Points partitioned into two clusters by the Lloyd algorithm; points are colored red or blue depending on their cluster membership. (Right) We can visualize a soft clustering of the same data into two clusters as assigning each point a pair of numbers representing the points percentage of "blue" and "red" based on each clusters "responsibility" for this point. The colors mix to form a color from the red-blue spectrum.

stronger that center's "pull" should be on the point. Given k centers Centers = (x1, ..., xk) and n points Data = (Data1, ... , Datan), we therefore need to construct a k $\times$ n responsibility matrix HiddenMatrix for which $HiddenMatrix_{i,j}$ is the pull of center i on data point j. This pull can be computed according to the Newtonian inverse-square law of gravitation,

$$HiddenMatrix_{i,j} = \frac{1/d(Data_j, x_i)^2}{\sum_{\text{all centers } x_i} 1/d(Data_j, x_i)^2}$$

Fig. 11.

Actually, the partition function from statistical physics often works better in practice:

$$HiddenMatrix_{i,j} = \frac{e^{-\beta \cdot d(Data_j,\, x_i)}}{\sum_{\text{all centers } x_i} e^{-\beta \cdot d(Data_j,\, x_i)}}$$

Fig. 12.

In this formula, e is the base of the natural logarithm ($e \approx 2.72$), and β is a parameter reflecting the amount of flexibility in our soft assignment and called-appropriately enough-the stiffness parameter.

In soft k-means clustering, if we let $HiddenMatrix_i$ denote the i-th row of HiddenMatrix, then we can update center x_i using an analogue of the above formulas. Specifically, we will define the j-th coordinate of center x_i, denoted $x_{i,j}$, as

$$x_{i,j} = \frac{HiddenMatrix_i \cdot Data^j}{HiddenMatrix_i \cdot \vec{1}}$$

Fig. 13.

Here, $Data_j$ is the n-dimensional vector holding the j-th coordinates of the n points in Data. The updated center x_i is called a weighted center of gravity of the points Data.

All information above provides the concepts and methods we used in this project. Actually, we apply Fuzzy C-means clustering algorithm on the gene expression data. The code is

implemented by myself, which would read data and convert the data into 62 × 2,000 expression matrix. The first 40 rows indicate the cancer samples, and the next 61 rows denotes 21 healthy samples. The 62th row, which is the last row in the data, is gene expression for a new unknown patient. Each sample in one row has 2,000 features. The algorithm would partition all 62 samples into 2 clustering. The color of the bar would be more likely to be a tumor sample if it is close to red, while it has bigger probability to be healthy if the color is close to blue. Since the color from red to blue is changing continuously instead of suddenly, we could assign each sample a pair of numbers representing the sample percentage of blue and red based on each cluster responsibility for this sample, instead of hard assigning each sample to only one cluster, which would provides a more accurate prediction and a more reasonable result.

IV. EXPERIMENTAL RESULTS

By implementing code based on Fuzzy C-means clustering algorithm, we could get two main results, one is a list of pairs of numbers, as shown in Figure 15, where the first number is the probability of the sample being tumor sample, and the second number is the probability of the sample being healthy sample. Since there are many pairs of numbers, it is not easy to see the result obviously. To make the final result easier to see, and make it interesting, the code represent the result by visualization technique. The results are presented in terms of spectrum, and bar on the right shows the probability. Red color indicates probability is equal to 1, while blue means probability equal 0.

From the spectral analysis in Figure 14, we could see the color of the unknown sample is closer to blue than red. Blue indicates that it has more probability for healthy sample, while red means the unknown sample is more likely as being cancer sample. In the list of the values of result, as shown in Figure 15, the value of probability for healthy sample is 0.975 (97.5%), while the value of probability for cancer sample is 0.025 (2.5%). So we could predict the new patient is more likely to be a healthy person.

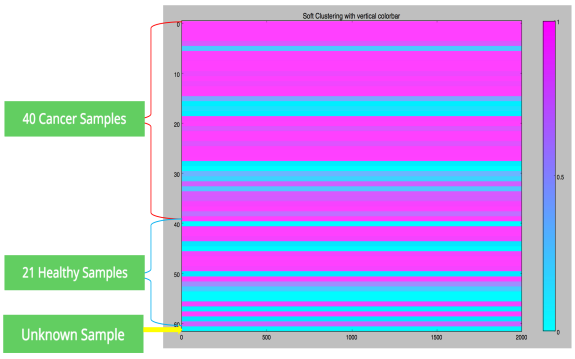

Fig. 14. Final Spectral Analysis

```
[  1.02639100e-01   8.97360900e-01]
[  9.96207051e-01   3.79294924e-03]
[  5.40945278e-01   4.59054722e-01]
[  2.39438828e-01   7.60561172e-01]
[  2.61406348e-03   9.97385937e-01]
[  6.99915701e-03   9.93000843e-01]
[  9.26741469e-01   7.32585310e-02]
[  5.62418181e-02   9.43758182e-01]
[  9.88066625e-01   1.19333749e-02]
[  1.16399512e-01   8.83600488e-01]
[  7.55957514e-01   2.44042486e-01]
[  2.46204475e-02   9.75379552e-01]]
```

Fig. 15. Final Vaules of Result

V. DISCUSSIOIN AND CONCLUSIONS

From the results, we could see data mining could partition the samples into two disjoint clusters, one is cancer samples, and the other is healthy ones. So if we have enough number of samples and information or features of the samples, it is possible to get a good prediction.

VI. FUTURE WORK

There are still a lots of work to do to improve the spectral analysis. Analyze and reduce the dimensions of features to reach better clustering.Need to get more gene expression data to reach better clustering. All these kinds of jobs could separate tumor and healthy samples more obviously. And then the prediction would have more accurate results.

ACKNOWLEDGMENT

The author thanks Prof. Peter Revesz who gave me the opportunity to do this wonderful project on the topic Genomic Data Mining, which also helped me in doing a lot of Research and i came to know about so many new things I am really thankful to them.

REFERENCES

[1] Large Scale Gene Expression Data Analysis I, available at: http://compbio.uthsc.edu/microarray/lecture1.htm
[2] Clustering in metric spaces, available at: http://cseweb.ucsd.edu/ dasgupta/291-geom/kcenter.pdf
[3] Farthest-first traversal, available at: https://en.wikipedia.org/wiki/Farthest-first-traversal
[4] Lloyd's algorithm, available at: https://en.wikipedia.org/wiki/Lloyd27s-algorithm
[5] NP-completeness, available at: https://en.wikipedia.org/wiki/NP-completeness
[6] Center of gravity, available at: https://en.wikipedia.org/wiki/Center_of_gravity
[7] C. Piech, K Means, available at: http://stanford.edu/ cpiech/cs221/handouts/kmeans.html
[8] Introduction to Data Analysis: Clustering, available at: http://www.kosbie.net/cmu/fall-10/15-110/handouts/notes-clustering/notes-clustering.html
[9] Implementation and Use of The K-Means Algorithm, available at: http://www.cs.colostate.edu/ anderson/cs545/index.html/lib/exe/fetch.php?media=assignments:solutions1:five.pdf
[10] k-means++: The Advantages of Careful Seeding, available at: http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf
[11] FUZZY CLUSTERING, available at: https://homes.di.unimi.it/valenti/SlideCorsi/Bioinformatica05/Fuzzy-Clustering-lecture-Babuska.pdf