International Journal of Engineering Science Invention
ISSN (Online): 2319 – 6734, ISSN (Print): 2319 – 6726
www.ijesi.org Volume 2 Issue 7 ǁ July. 2013 ǁ PP.01-08

Multidimensional Clustering Methods of Data Mining for Industrial Applications

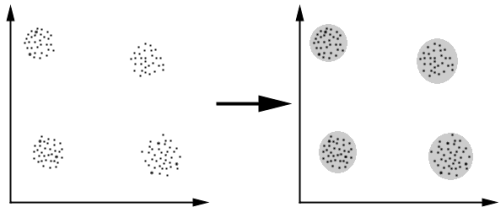
Ashok Kumar Thavani Andu[1], Dr. Antony Selvdoss Thanamani[2]
[1]*Assistant Professor, CMS College, Coimabtore.Tamilnadu.INDIA*
[2]*Professor & Head, NGM College, Pollachi. Tamilnadu.India*

ABSTRACT: *Clustering is the unsupervised classification of patterns (observations, data items, or feature vectors) into groups (clusters). The clustering problem has been addressed in many contexts and by researchers in many disciplines; this reflects its broad appeal and usefulness as one of the steps in exploratory data analysis. However, clustering is a difficult problem combinatorially, and differences in assumptions and contexts in different communities have made the transfer of useful generic concepts and methodologies slow to occur. This paper presents an overview of pattern clustering methods from a statistical pattern recognition perspective, with a goal of providing useful advice and references to fundamental concepts accessible to the broad community of clustering practitioners. We present a taxonomy of clustering techniques, and identify cross-cutting themes and recent advances. We also describe some important applications of clustering algorithms such as image segmentation, object recognition, and information retrieval.*

I. INTRODUCTION

Clustering can be considered the most important *unsupervised, earning* problem; so, as every other problem of this kind, it deals with finding a *structure* in a collection of unlabeled data. A loose definition of clustering could be "the process of organizing objects into groups whose members are similar in some way". A *cluster* is therefore a collection of objects which are "similar" between them and are "dissimilar" to the objects belonging to other clusters.

We can show this with a simple graphical example:



In this case we easily identify the 4 clusters into which the data can be divided; the similarity criterion is *distance*: two or more objects belong to the same cluster if they are "close" according to a given distance (in this case geometrical distance). This is called *distance-based clustering.*

Another kind of clustering is *conceptual clustering*: two or more objects belong to the same cluster if this one defines a concept *common* to all that objects. In other words, objects are grouped according to their fit to descriptive concepts, not according to simple similarity measures.

II. THE GOALS OF CLUSTERING

The goal of clustering is to determine the intrinsic grouping in a set of unlabeled data. But how to decide what constitutes a good clustering? It can be shown that there is no absolute "best" criterion which would be independent of the final aim of the clustering. Consequently, it is the user which must supply this criterion, in such a way that the result of the clustering will suit their needs

Possible Applications

clustering algorithms can be applied in many fields, for instance:

- Marketing: finding groups of customers with similar behavior given a large database of customer data containing their properties and past buying records;
- Biology: classification of plants and animals given their features;
- Libraries: book ordering;

- Insurance: identifying groups of motor insurance policy holders with a high average claim cost; identifying frauds;
- City-planning: identifying groups of houses according to their house type, value and geographical location;
- Earthquake studies: clustering observed earthquake epicenters to identify dangerous zones;
- WWW: document classification; clustering weblog data to discover groups of similar access patterns.

Requirements

the main requirements that a clustering algorithm should satisfy are:

- scalability;
- dealing with different types of attributes;
- discovering clusters with arbitrary shape;
- minimal requirements for domain knowledge to determine input parameters;
- ability to deal with noise and outliers;
- insensitivity to order of input records;
- high dimensionality;
- Interpretability and usability.

Problems

There are a number of problems with clustering. Among them:

- current clustering techniques do not address all the requirements adequately (and concurrently);
- dealing with large number of dimensions and large number of data items can be problematic because of time complexity;
- the effectiveness of the method depends on the definition of "distance" (for distance-based clustering);
- if an *obvious* distance measure doesn't exist we must "define" it, which is not always easy, especially in multi-dimensional spaces;
- the result of the clustering algorithm (that in many cases can be arbitrary itself) can be interpreted in different ways.

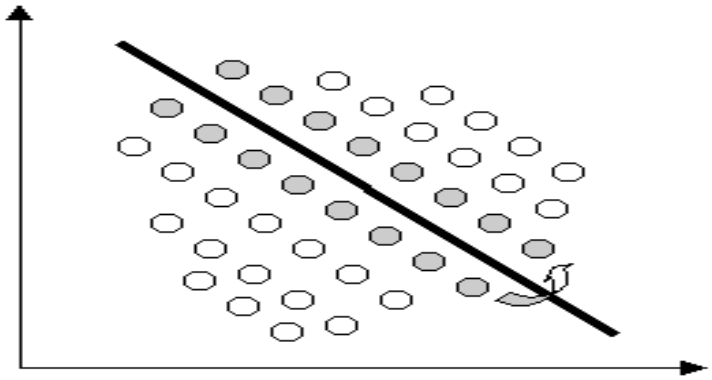
Clustering Algorithms

Classification

clustering algorithms may be classified as listed below:

- Exclusive Clustering
- Overlapping Clustering
- Hierarchical Clustering
- Probabilistic Clustering

In the first case data are grouped in an exclusive way, so that if a certain datum belongs to a definite cluster then it could not be included in another cluster. A simple example of that is shown in the figure below, where the separation of points is achieved by a straight line. On contrary the second type, the overlapping clustering, uses fuzzy sets to cluster data, so that each point may belong to two or more clusters with different degrees of membership. In this case, data will be associated to an appropriate membership value.



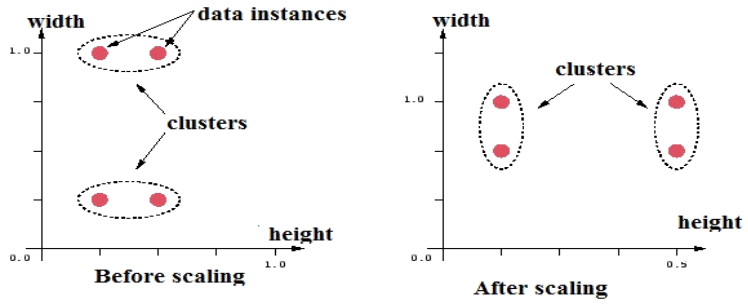
Instead, a hierarchical clustering algorithm is based on the union between the two nearest clusters. The beginning condition is realized by setting every datum as a cluster. After a few iterations it reaches the final clusters wanted.

Finally, the last kind of clustering uses a completely probabilistic approach.
In this tutorial we propose four of the most used clustering algorithms:

- K-means
- Fuzzy C-means
- Hierarchical clustering
- Mixture of Gaussians

III. DISTANCE MEASURE

An important component of a clustering algorithm is the distance measure between data points. If the components of the data instance vectors are all in the same physical units then it is possible that the simple Euclidean distance metric is sufficient to successfully group similar data instances. However, even in this case the Euclidean distance can sometimes be misleading. Figure shown below illustrates this with an example of the width and height measurements of an object. Despite both measurements being taken in the same physical units, an informed decision has to be made as to the relative scaling. As the figure shows, different scaling can lead to different clustering.



Notice however that this is not only a graphic issue: the problem arises from the mathematical formula used to combine the distances between the single components of the data feature vectors into a unique distance measure that can be used for clustering purposes: different formulas leads to different clusterings. Again, domain knowledge must be used to guide the formulation of a suitable distance measure for each particular application.

IV. MINKOWSKI METRIC

For higher dimensional data, a popular measure is the Minkowski metric,

$$d_p(x_i, x_j) = \left(\sum_{k=1}^{d} \left| x_{i,k} - x_{j,k} \right|^p \right)^{\frac{1}{p}}$$

Where *d* is the dimensionality of the data. The *Euclidean* distance is a special case where *p*=2, while *Manhattan* metric has *p*=1. However, there are no general theoretical guidelines for selecting a measure for any given application.

It is often the case that the components of the data feature vectors are not immediately comparable. It can be that the components are not continuous variables, like length, but nominal categories, such as the days of the week. In these cases again, domain knowledge must be used to formulate an appropriate measure.

The Algorithm K- means (MacQueen, 1967) is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. 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 main idea is to define k centroids, one for each cluster. These centroids should be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest centroid. When no point is pending, the first step is completed and an early groupage is done. At this point we need to re-calculate k new centroids as barycenters of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new centroid. A loop has been generated. As a result of this change their location step by step until no more changes are done. In other words centroids do not move any more.

Finally, this algorithm aims at minimizing an *objective function*, in this case a squared error function. The objective function

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2$$

,

Where $\left\| x_i^{(j)} - c_j \right\|^2$ is a chosen distance measure between a data point $x_i^{(j)}$ and the cluster centre c_j, is an indicator of the distance of the *n* data points from their respective cluster centers.

The algorithm is composed of the following steps:

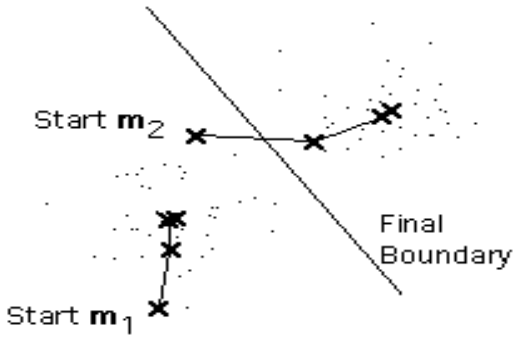
Although it can be proved that the procedure will always terminate, the k-means algorithm does not necessarily find the most optimal configuration, corresponding to the global objective function minimum. The algorithm is also significantly sensitive to the initial randomly selected cluster centres. The k-means algorithm can be run multiple times to reduce this effect.

K-means is a simple algorithm that has been adapted to many problem domains. As we are going to see, it is a good candidate for extension to work with fuzzy feature vectors.

Suppose that we have n sample feature vectors $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$ all from the same class, and we know that they fall into k compact clusters, k < n. Let $\mathbf{m}_i$ be the mean of the vectors in cluster i. If the clusters are well separated, we can use a minimum-distance classifier to separate them. That is, we can say that **x** is in cluster i if $\| \mathbf{x} - \mathbf{m}_i \|$ is the minimum of all the k distances. This suggests the following procedure for finding the k means:

- Make initial guesses for the means $\mathbf{m}_1, \mathbf{m}_2, ..., \mathbf{m}_k$
- Until there are no changes in any mean
- Use the estimated means to classify the samples into clusters
- For i from 1 to k
- Replace $\mathbf{m}_i$ with the mean of all of the samples for cluster i
- end_for
- end_until

Here is an example showing how the means $\mathbf{m}_1$ and $\mathbf{m}_2$ move into the centers of two clusters.

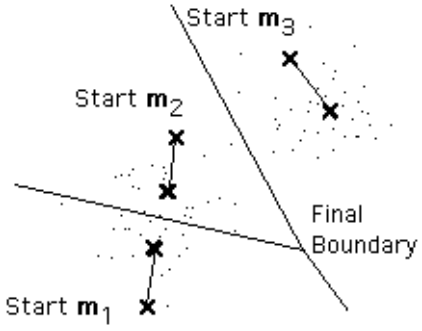


Remarks

This is a simple version of the k-means procedure. It can be viewed as a greedy algorithm for partitioning the n samples into k clusters so as to minimize the sum of the squared distances to the cluster centers. It does have some weaknesses:

- The way to initialize the means was not specified. One popular way to start is to randomly choose k of the samples.
- The results produced depend on the initial values for the means, and it frequently happens that suboptimal partitions are found. The standard solution is to try a number of different starting points.
- It can happen that the set of samples closest to $\mathbf{m}_i$ is empty, so that $\mathbf{m}_i$ cannot be updated. This is an annoyance that must be handled in an implementation, but that we shall ignore.
- The results depend on the metric used to measure $\| \mathbf{x} - \mathbf{m}_i \|$. A popular solution is to normalize each variable by its standard deviation, though this is not always desirable.
- The results depend on the value of k.

This last problem is particularly troublesome, since we often have no way of knowing how many clusters exist. In the example shown above, the same algorithm applied to the same data produces the following 3-means clustering. Is it better or worse than the 2-means clustering?



Unfortunately there is no general theoretical solution to find the optimal number of clusters for any given data set. A simple approach is to compare the results of multiple runs with different k classes and choose the best one according to a given criterion (for instance the Schwarz Criterion - see Moore's slides), but we need to be careful because increasing k results in smaller error function values by definition, but also an increasing risk of overfitting.

V. FUZZY C-MEANS CLUSTERING

The Algorithm Fuzzy c-means (FCM) is a method of clustering which allows one piece of data to belong to two or more clusters. This method (developed by Dunn in 1973 and improved by Bezdek in 1981) is frequently used in pattern recognition. It is based on minimization of the following objective function:

$$J_m = \sum_{i=1}^{N}\sum_{j=1}^{C} u_{ij}^{m}\left\|x_i - c_j\right\|^2 \quad , \quad 1 \le m < \infty$$

where m is any real number greater than 1, u_{ij} is the degree of membership of x_i in the cluster j, x_i is the ith of d-dimensional measured data, c_j is the d-dimension center of the cluster, and ||*|| is any norm expressing the similarity between any measured data and the center.

Fuzzy partitioning is carried out through an iterative optimization of the objective function shown above, with the update of membership u_{ij} and the cluster centers c_j by:

$$u_{ij} = \frac{1}{\sum_{k=1}^{C}\left(\frac{\left\|x_i - c_j\right\|}{\left\|x_i - c_k\right\|}\right)^{\frac{2}{m-1}}} \quad , \quad c_j = \frac{\sum_{i=1}^{N} u_{ij}^{m} \cdot x_i}{\sum_{i=1}^{N} u_{ij}^{m}}$$

This iteration will stop when $\max_{ij}\left\{\left|u_{ij}^{(k+1)} - u_{ij}^{(k)}\right|\right\} < \varepsilon$, where ε is a termination criterion between 0 and 1, whereas k are the iteration steps. This procedure converges to a local minimum or a saddle point of J_m. The algorithm is composed of the following steps:

- *Initialize* $U=[u_{ij}]$ *matrix,* $U^{(0)}$
- *At k-step: calculate the centers vectors* $C^{(k)}=[c_j]$ *with* $U^{(k)}$

$$c_j = \frac{\sum_{i=1}^{N} u_{ij}^{m} \cdot x_i}{\sum_{i=1}^{N} u_{ij}^{m}}$$

- *Update* $U^{(k)}$, $U^{(k+1)}$

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{\|x_i - c_j\|}{\|x_i - c_k\|} \right)^{\frac{2}{m-1}}}$$

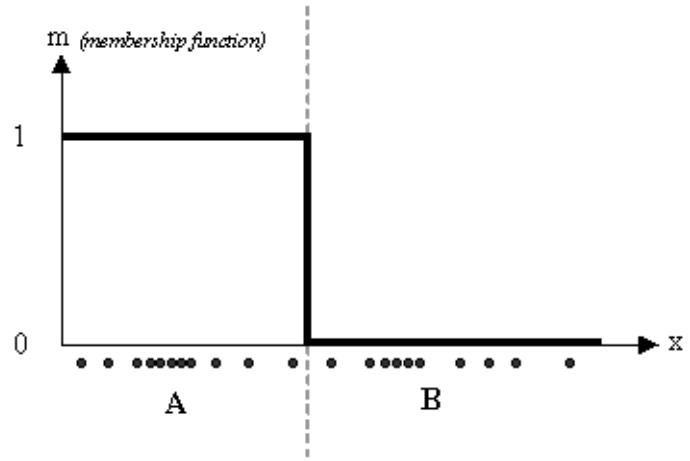
- *If* $|| U^{(k+1)} - U^{(k)}|| < \varepsilon$ *then STOP; otherwise return to step 2.*

Remarks

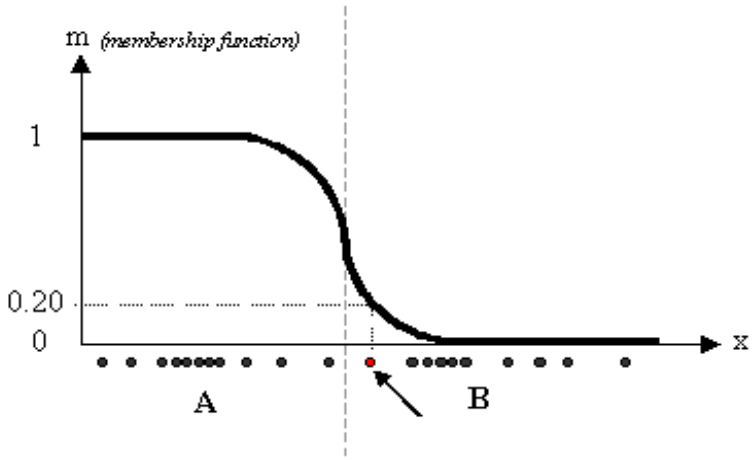
As already told, data are bound to each cluster by means of a Membership Function, which represents the fuzzy behavior of this algorithm. To do that, we simply have to build an appropriate matrix named U whose factors are numbers between 0 and 1, and represent the degree of membership between data and centers of clusters

For a better understanding, we may consider this simple mono-dimensional example. Given a certain data set, suppose to represent it as distributed on an axis.

Looking at the picture, we may identify two clusters in proximity of the two data concentrations. We will refer to them using 'A' and 'B'. In the first approach shown in this tutorial - the k-means algorithm - we associated each datum to a specific cancroids; therefore, this membership function looked like this:



In the FCM approach, instead, the same given datum does not belong exclusively to a well defined cluster, but it can be placed in a middle way. In this case, the membership function follows a smoother line to indicate that every datum may belong to several clusters with different values of the membership coefficient.



In the figure above, the datum shown as a red marked spot belongs more to the B cluster rather than the A cluster. The value 0.2 of 'm' indicates the degree of membership to A for such datum. Now, instead of using a graphical representation, we introduce a matrix U whose factors are the ones taken from the membership functions:

$$U_{MxC} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ .. & .. \\ 0 & 1 \end{bmatrix} \qquad U_{MxC} = \begin{bmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \\ 0.6 & 0.4 \\ .. & .. \\ 0.9 & 0.1 \end{bmatrix}$$

(a) (b)

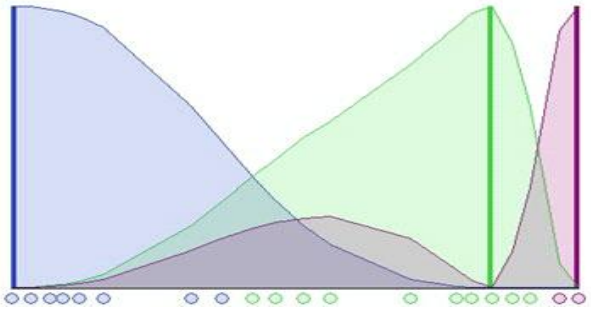
The number of rows and columns depends on how many data and clusters we are considering. More exactly we have C = 2 columns (C = 2 clusters) and N rows, where C is the total number of clusters and N is the total number of data. The generic element is so indicated: u_{ij}.

In the examples above we have considered the k-means (a) and FCM (b) cases. We can notice that in the first case (a) the coefficients are always unitary. It is so to indicate the fact that each datum can belong only to one cluster. Other properties are shown below:

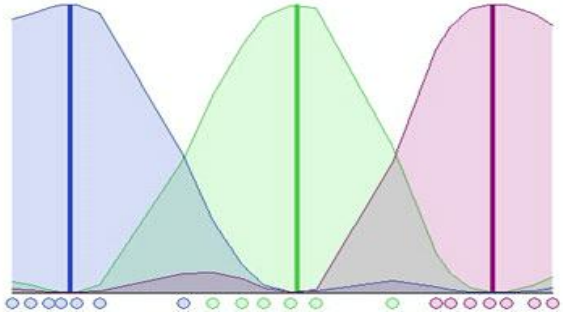
- $u_{ij} \in [0,1] \qquad \forall i, j$
- $\sum_{j=1}^{C} u_{ik} = 1 \qquad \forall i$
- $0 < \sum_{i=1}^{N} u_{ij} < N \quad \forall N$

An Example

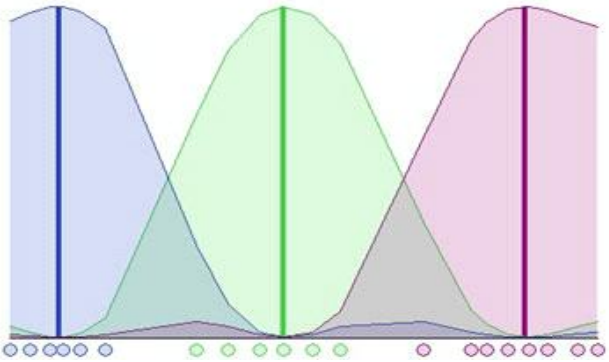
Here, we consider the simple case of a mono-dimensional application of the FCM. Twenty data and three clusters are used to initialize the algorithm and to compute the U matrix. Figures below (taken from our interactive demo) show the membership value for each datum and for each cluster. The color of the data is that of the nearest cluster according to the membership function.



In the simulation shown in the figure above we have used a fuzzyness coefficient m = 2 and we have also imposed to terminate the algorithm when $\max_{ij}\left\{\left|u_{ij}^{(k+1)} - u_{ij}^{(k)}\right|\right\} < 0.3$. The picture shows the initial condition where the fuzzy distribution depends on the particular position of the clusters. No step is performed yet so that clusters are not identified very well. Now we can run the algorithm until the stop condition is verified. The figure below shows the final condition reached at the 8th step with m=2 and ε=0.3:



Is it possible to do better? Certainly, we could use an higher accuracy but we would have also to pay for a bigger computational effort. In the next figure we can see a better result having used the same initial conditions and ε=0.01, but we needed 37 steps!



It is also important to notice that different initializations cause different evolutions of the algorithm. In fact it could converge to the same result but probably with a different number of iteration steps.

REFERENCES

[1] Osmar R. Zaïane: "Principles of Knowledge Discovery in Databases - Chapter 8: Data Clustering"

[2] J. B. MacQueen (1967): "Some Methods for classification and Analysis of Multivariate Observations, *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability"*, Berkeley, University of California Press, 1:281-297

[3] Andrew Moore: "K-means and Hierarchical Clustering - Tutorial Slides"

[4] J. C. Dunn (1973): "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters", *Journal of Cybernetics* 3: 32-57

[5] J. C. Bezdek (2005): "Pattern Recognition with Fuzzy Objective Function Algorithms", Plenum Press, New York

[6] Tariq Rashid: "Clustering".

[7] Hans-Joachim Mucha and Hizir Sofyan: "Nonhierarchical Clustering"

[8] 12 Karp, R. M., Rabin M. O.: Efficient Randomized Pattern-Matching Algorithms. IBM

[9] Journal of Research and Development, vol. 31(2), pp. 249-260 (1987)

[10] 13 Li, B., Xu, S., Zhang, J.: Enhancing Clustering Blog Documents by Utilizing Author/Reader

[11] Comments, ACM Southeast Regional Conference, pp. 94-99 (2007)

[12] [Sholom et al. 2004] Sholom, M., Indurkhya, N., Zhang, T., Damerau, F. Text Mining , Predictive Methods for Analyzing Unstructured Information. Springer, 2004.

[13] [Zhong 2006] Zhong, S. Semi-supervised Model-based Document Clustering: A Comparative Study. Machine Learning, (65) 1, 2006.