Fuzzifying clustering algorithms: The case study of MajorClust

Eugene Levner[1], David Pinto[(2,3)], Paolo Rosso[2],
David Alcaide[4], R.R.K. Sharma[5]

[1]Holon Institute of Technology, Holon, Israel
[2]Department of Information Systems and Computation, UPV, Spain
[3]Faculty of Computer Science, BUAP, Mexico
[4]Universidad de La Laguna, Tenerife, Spain
[5]Indian Institute of Technology, Kanpur, India
levner@hit.ac.il, {dpinto, prosso}@dsic.upv.es,
dalcaide@ull.es, rrks@iitk.ac.in

Abstract. Among various document clustering algorithms that have been proposed so far, the most useful are those that automatically reveal the number of clusters and assign each target document to exactly one cluster. However, in many real situations, there not exists an exact boundary between different clusters. In this work, we introduce a fuzzy version of the MajorClust algorithm. The proposed clustering method assigns documents to more than one category by taking into account a membership function for both, edges and nodes of the corresponding underlying graph. Thus, the clustering problem is formulated in terms of weighted fuzzy graphs. The fuzzy approach permits to decrease some negative effects which appear in clustering of large-sized corpora with noisy data.

1 Introduction

Clustering analysis refers to the partitioning of a data set into subsets (clusters), so that the data in each subset (ideally) share some common trait, often proximity, according to some defined distance measure [1,2,3]. Clustering methods are usually classified with respect to their underlying algorithmic approaches: hierarchical, iterative (or partitional) and density based are some instances belonging to this classification. Hierarchical algorithms find successive clusters using previously established ones, whereas partitional algorithms determine all clusters at once. Hierarchical algorithms can be agglomerative ("bottom-up") or divisive ("top-down"); agglomerative algorithms begin with each element as a separate cluster and merge them into successively larger clusters. Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters. Iterative algorithms start with some initial clusters (their number either being unknown in advance or given a priori) and intend to successively improve the existing cluster set by changing their "representatives" ("centers of gravity", "centroids") , like in K-Means [3] or by iterative node-exchanging (like in [4]).

An interesting density-based algorithm is MajorClust [5], which automatically reveals the number of clusters, unknown in advance, and successively increases the total "strength" or "connectivity" of the cluster set by cumulative attraction of nodes between the clusters.

MajorClust [5] is one of the most promising and successful algorithms for unsupervised document clustering. This graph theory based algorithm assigns each document to that cluster the majority of its neighbours belong to. The node neighbourhood is calculated by using a specific similarity measure which is assumed to be the weight of each edge (similarity) between the nodes (documents) of the graph (corpus). MajorClust automatically reveals the number of clusters and assigns each target document to exactly one cluster. However, in many real situations, there not exists an exact boundary between different categories. Therefore, a different approach is needed in order to determine how to assign a document to more than one category. The traditional MajorClust algorithm deals with crisp (hard) data, whereas the proposed version, called F-MajorClust, will use fuzzy data. We suggest to calculate a weigthed fuzzy graph with edges between any pairs of nodes that are supplied with fuzzy weights which may be either fuzzy numbers or linguistic variables. Thereafter, a fuzzy membership function will be used in order to determine the possible cluster a node belongs to. The main feature of the new algorithm, F-MajorClust, which differs from MajorClust, is that all the items (for example, the documents to be grouped) are allowed to belong to two and more clusters.

The rest of this paper is structured as follows. The following section recalls the case study of the K-means algorithm and its fuzzy version. In Section 3 the traditional MajorClust algorithm is described whereas its fuzzy version is discussed in Section 4. Finally we give some conclusions and further work.

2 K-means and fuzzy K-means clustering

The widely known K-means algorithm assigns each point to the cluster whose center is nearest. The center is the average of all the points of the cluster. That is, its coordinates are the arithmetic mean for each dimension separately over all the points in the cluster. The algorithm steps are ([3]):

1. Choose the number k of clusters.
2. Randomly generate k clusters and determine the cluster centers, or directly generate k random points as cluster centers.
3. Assign each point to the nearest cluster center.
4. Recompute the new cluster centers.
5. Repeat the two previous steps until some convergence criterion is met (usually that the assignment has not changed).

The main advantages of this algorithm are its simplicity and speed which allows it to run on large datasets. This explains its wide applications in very many areas. However, a number of questions arise of which some examples follow. Since the choice of centers and assignment are random, and, hence, resulting clusters

can be different with each run, how to yield the best possible result? How can we choose the "best" distance d among very many possible options? How can we choose the "best" center if we have many possible competing variants? How do we choose the number K of clusters, especially in large-scale data bases? Having found multiple alternative clusterings for a given K, how can we then choose among them?

Moreover, K-means has several pathological properties and limitations. The algorithm takes account only of the distance between the centers and the data points; it has no representation of the weight or size of each cluster. Consequently, K-means behaves badly if several data bases strongly differ in size; indeed, data points that actually belong to the broad cluster have a tendency to be incorrectly assigned to the smaller cluster (see [1], for examples and details). Further, the K-means algorithm has no way of representing the size or shape of a cluster. In [1], there given an example when the data naturally fall into two narrow and elongated clusters. However, the only stable state of the K-means algorithm is that the two clusters will be erroneously sliced in half. One more criticism of K-means is that it is a 'crisp' rather than a 'soft' algorithm: points are assigned to exactly one cluster and all points assigned to a cluster are equals in that cluster. However, points located near the border between two or more clusters should, apparently, play a partial role in several bordering clusters. The latter disadvantage is overcome by Fuzzy K-means described below, whereas another mentioned-above disability of the K-means associated with the size and shape of clusters, is treated by the algorithm F-MajorClust presented in Section 4. In fuzzy clustering, data elements may belong to more than one cluster, and associated with each element we have a set of membership levels. These indicate a degree of belonging to clusters, or the "strength" of the association between that data element and a particular cluster. Fuzzy clustering is a process of assigning these membership levels, and then using them to assign data elements to one or more clusters. Thus, border nodes of a cluster, may be in the cluster to a lesser degree than inner nodes. For each point x we have a coefficient giving the degree of being in the k-th cluster $u_k(x)$. Usually, the sum of those coefficients is defined to be 1:

$$\forall x \quad \sum_{k=1}^{NumClusters} u_k(x) = 1 \tag{1}$$

In the fuzzy C-Means (developed by Dunn in 1973 [6] and improved by Bezdek in 1981 [7]), the center (called "centroid") of a cluster is the mean of all points, weighted by their degree of belonging to the cluster:

$$\text{center}_k = \frac{\sum_x u_k(x)^m x}{\sum_x u_k(x)^m} \tag{2}$$

where m is a fuzzification exponent; the larger the value of m the fuzzier the solution. At $m = 1$, fuzzy C-Means collapses to the crisp K-means, whereas at very large values of m, all the points will have equal membership with all the

clusters. The degree of belonging is related to the inverse of the distance to the cluster:

$$u_k(x) = \frac{1}{d(\text{center}_k, x)}, \tag{3}$$

then the coefficients are normalised and fuzzyfied with a real parameter $m > 1$ so that their sum is 1. Therefore,

$$u_k(x) = \frac{1}{\sum_j d(\text{center}_k, x) d(\text{center}_j, x)}, \tag{4}$$

for m equal to 2, this is equivalent to linearly normalise the coefficient to make the sum 1. When m is close to 1, then the closest cluster center to the point is given a greater weight than the others. The fuzzy C-Means algorithm is very similar to the K-means algorithm: Its steps are the following ones:

1. Choose a number of clusters.
2. Assign randomly to each point coefficients $u_k(x)$ for being in the clusters.
3. Repeat until the algorithm has converged (that is, the coefficients' change between two iterations is no more than ϵ, the given "sensitivity threshold"):
   (a) Compute the centroid for each cluster, using the formula above.
   (b) For each point, compute its coefficients $u_k(x)$ of being in the clusters, using the formula above.

The fuzzy algorithm has the same problems as K-means: the results depend on the initial choice of centers, assignments and weights, and it has no way of taking the shape or size of the clusters into account. The algorithm MajorClust presented below is intended to overcome the latter disadvantage.

3 The MajorClust clustering algorithm

The algorithm is designed to find the cluster set maximizing the total cluster connectivity $\Lambda(C)$, which is defined in [5] as follows:

$$\Lambda(C) = \sum_{k=1}^{K} |C_k| \lambda_k, \tag{5}$$

where C denotes the decomposition of the given graph G into clusters, $C_1, C_2, \cdots, C_k$ are clusters in the decomposition C, λ_k designates the edge connectivity of cluster $G(C_k)$, that is, the mininum number of edges that must be removed to make graph $G(C_k)$ disconnected.

MajorClust operationalises iterative propagation of nodes into clusters according to the "maximum attraction wins" principle [8]. The algorithm starts by assigning each point in the initial set its own cluster. Within the following relabelling steps, a point adopts the same cluster label as the "weighted majority of its neighbours". If several such clusters exist, one of them is chosen randomly. The algorithm terminates if no point changes its cluster membership.

The MajorClust algorithm
Input: object set D, similarity measure $\varphi : D \times D \to [0; 1]$, similarity threshold τ.
Output: function $\delta : D \to N$, which assigns a cluster label to each point.

1. $i := 0$, ready := false
2. for all p from D do $i := i + 1$, $\delta(p) := i$ enddo
3. while ready = false do
   (a) ready := true
   (b) for all q from D do
      i. $\delta^* := i$ if $\Sigma\{\varphi(p, q)|\varphi(p; q) \geq t$ and $\delta(p) = i\}$ is maximum.
      ii. if $\delta(q) \neq \delta^*$ then $\delta(q) := \delta^*$, ready := false
   (c) enddo
4. enddo

Remark. The similarity thershold τ is not a problem-specific parameter but a constant that serves for noise filtering purposes. Its typical value is 0.3.

The MajorClust is a relatively new clustering algorithm with respect to other methods. However, its characteristic of automatically discovering the target number of clusters make it even more and more attractive [9,10,11,12], and hence the motivation of fuzzifying it.

In the following section, we will describe in detail the proposed fuzzy approach for the traditional MajorClust clustering algorithm.

4 Fuzzification of MajorClust

4.1 Fuzzy weights of edges and nodes in *F*-MajorClust

The measure of membership of any edge i in a cluster k is presented by a membership function μ_{ik}, where $0 \leq \mu_{ik} \leq 1$, and $\sum_k \mu_{ik} = 1$ for any i.

In order to understand the way it is emplyed, we will need the following definitions. A node j is called *inner* if all its neighbours belong to the same cluster as the node j. If an edge i connects nodes x and y, we will say that x and y are the *end* nodes of the edge i. A node j is called *boundary* if some of its neighbours belong to a cluster (or several clusters) other than the cluster containing the node j itself.

The main ideas behind the above concept of the fuzzy membership function μ_{ik} is that the edges connecting the inner nodes in a cluster may have a larger "degree of belonging" to a cluster than the "peripheral" edges (which, in a sense, reflects a greater "strength of connectivity" between a pair of nodes). For instance, the edges (indexed i) connecting the *inner nodes* in a cluster (indexed k) are assigned $\mu_{ik} = 1$ whereas the edges linking the *boundary nodes* in a cluster have $\mu_{ik} < 1$. The latter dependence reflects the fact that in the forthcoming algorithm the boundary nodes have more chances to leave a current cluster than the inner ones, therefore, the "strength of connectivity" of a corresponding edge

in the current cluster is smaller. As a simple instance case, we define $\mu_{ik} = \frac{a_{ik}}{b_i}$, where a_{ik} is the number of those neighbours of the end nodes of i that belong to the same cluster k as the end nodes of i, and b_i is the number of all neighbours to the end nodes of i. In a more advanced case, we define $\mu_{ik} = \frac{A_{ik}}{B_i}$, where A_{ik} is the sum of the weights of edges linking the end nodes of i with those neighbours of the end nodes of i that belong to the same cluster k as the end nodes of i, and B_i is the total sum of the weights of the edges adjacent to the edge i.

Furthermore, we introduce the measure of membership of any item (node) j in any cluster k, which is presented by the membership function γ_{jk}, where $0 \leq \gamma_{jk} \leq 1$, and $\sum_k \gamma_{jk} = 1$ for any j. Notice that these weights are assigned to nodes, rather than to the edges: this specific feature being absent in all previous algorithms of MajorClust type. The value of the membership function γ_{jk} reflects the *semantic correspondence* of node j to cluster k, and is defined according to the "fitness" of node j to cluster k as defined in [13]. The idea behind this concept is to increase the role of the nodes having a larger fitness to their clusters. In the formula (6) and the text below, γ_{jk} is a function of a cluster C_k containing the node j: $\gamma_{jk} = \gamma_{jk}(C_k)$ which may dynamically change in the algorithm suggested below as soon as C_k changes. The objective function in the clustering problem becomes more general than that in [5] so that the weights of nodes are being taken into account, as follows:

$$\textbf{Maximize} \quad \Lambda(C) = \sum_{k=1}^{K} \left(\sum_{j=1}^{|C_k|} \mu_{i(j),k} \lambda_k + \sum_{j=1}^{n} \gamma_{jk}(Ck) \right), \tag{6}$$

where C denotes the decomposition of the given graph G into clusters, $C_1, \ldots, C_K$ are not-necessarily disjoint clusters in the decomposition C, $\Lambda(C)$ denotes the total weighted connectivity of $G(C)$, λ_k designates, as in MajorClust, the edge connectivity of cluster $G(C_k)$, the weight $\mu_{i(j),k}$ is the membership degree of arc $i(j)$ containing node j in cluster k, and finally, $\gamma_{jk}(C_k)$ is the fitness of node j to cluster k. λ_k is calculated according to [5], meaning the cardinality of the set of edges of minimum total weight $\sum_i \mu_{ik}$ that must be removed in order to make the graph $G(C_k)$ disconnected.

4.2 Limitations of MajorClust and *if-then* rules

The fuzzy weights of edges and nodes (that is, in informal terms, the fuzzy semantic correlations between the documents and the fuzzy fitness of documents to categories) can be presented not only in the form of fuzzy numbers defined between 0 and 1 reflecting a flexible (fuzzy) measure of fitness (which sometimes is called "responsibility"). Moreover they even may be linguistic variables (e.g. small, medium, large, etc). In the latter case, they are assigned the so-called "grades" introduced in [14,13]. The presence of fuzzy weights on edges and nodes permit us to avoid several limitations of the standard MajorClust. The most important among them are the following ones:

1. When MajorClust runs, it may include nodes with weak links, i.e., with a small number of neighbours which inevitably leads to the decrease of the objective function already achieved (see [5]).
2. MajorClust assigns each node to that cluster the majority of its neighbours belong to, and when doing this, the algorithm does not specify the case when there are several "equivalent" clusters equally matching the node. The recommendation in [5] to make this assignment in an arbitrary manner, may lead to the loss of a neighbouring good solution.
3. MajorClust scans nodes of the original graph in an arbitrary order, which may lead to the loss of good neighbouring solutions.
4. MajorClust takes into account only one local minimum among many others (which may lead to the loss of much better solutions than the one selected). These limitations will be avoided in the fuzzy algorithm suggested, by the price of greater computational efforts (the running time) and a larger required memory.

In the following, we list a set of *if-then* rules which may be added to the proposed fuzzy version of the MajorClust. The order of node scan is defined by the following decision rules R1-R3.

Rule R1: *If* there are several nodes having majority in certain clusters (or the maximum value of the corresponding objective function), *then* choose first the node having the maximal number of neighbours.

Rule R2: *If* there are several nodes having both the majority and the maximum number of neighbours in certain clusters, *then* choose first the node whose inclusion leads to the maximum increase of the objective function.

Rule R2: *If* there are several nodes having both the majority and the maximum number of neighbours in certain clusters, *then* assign the nodes to clusters using the facility layout model taking into account the semantic fitness of nodes to clusters and providing the maximum increase of the objective function (the mathematical model formulation and computational approaches can be found in [15,16]).

Rule R3: *If*, at some iterative step, the inclusion of some node would lead to the decrease of the objective function, *then* this node should be skipped (that is, it will not be allocated into any new cluster at that step).

The algorithm stops when the next iterative step does not change the clustering (this is Rule 4) or any further node move leads to deteriorating of the achieved quality (defined by the formula (6) (this is Rule 5) or according to other stopping rules R6-R7 (see below).

Rule R6: *If* the number of steps exceeds the given threshold, *then* stop.

Rule R7: *If* the increase in the objective function at H current steps is less than ϵ (H and ϵ are given by the experts and decision makers in advance), *then* stop.

The F-MajorClust algorithm starts by assigning each point in the initial set its own cluster. Within the following re-labelling steps, a point adopts the same

cluster label as the " weighted majority of its neighbours", that is, the node set providing the maximum value to the generalized objective function [13]. If several such clusters exist, say z clusters, then a point adopts all of them and attains the membership function $\omega_{jk} = 1/z$ for all clusters. At each step, if point j belongs to y clusters, then $\omega_{jk} = 1/y$. The algorithm terminates if no point changes its cluster membership.

4.3 The butterfly effect

In Fig. 1 we can observe the so-called "butterfly effect", which appears when some documents (nodes) of the dataset (graph) may belong to more than one cluster, in this case the fuzzy algorithm works better than the crisp one. Fig. 1(a) and 1(b) depict an example when the classical MajorClust algorithm has found two clusters and, then, according to formula (5) this clustering of eight nodes obtains a score of 21 (C=7x3 + 1x0 = 21). Figure 1(b) shows the case when the classical MajorClust algorithm has detected three clusters for the same input data providing a total score of 18 (C=4x3 + 3x2 + 1x0 = 18). On the other hand, Fig. 1(c) and 1(d) demonstrate how the fuzzy algorithm works when some nodes can belong simultaneously to several different clusters. We assume that the algorithm uses formula (6) where, for the simplicity, we take $\gamma_{jk}(C_k) = 0$; even in this simplified case the fuzzy algorithm wins. Two variants are presented: in Figure 1(c) we consider the case when the membership values are shared equally between two cluster with the memebership value 0.5; then the obtained score is 21 (C=2x((3+0.5)x3) + 1x0 = 21). Note that the value of the objective function is here the same as in the case 1(a). However, if the documents (nodes) are highly relevant to the both databases with the membership function values 1 then the fuzzy algorithm yields a better score which is presented in Fig. 1(d): C=2x((3+1)x3) + 1x0=24. It worth noticing that this effect becomes even stronger if $\gamma_{jk}(C_k) > 0$ in (6).

5 Conclusions and further work

Our efforts were devoted to employ fuzzy clustering on text corpora since we have observed a good performance of the MajorClust in this context. We have proposed a fuzzy version of the classical MajorClust clustering algorithm in order to permit overlapping between the obtained clusters. This approach will provide a more flexible use of the mentioned clustering algorithm. We consider that there exist different areas of application for this new clustering algorithm which include not only data analysis but also pattern recognition, spatial databases, production management, etc. in cases when any object can be assigned to more than a unique category.

Special attention in the definition and operation of the so called fuzzifier will be needed, since it controls the amount of overlapping among the obtained clusters and, it is well known that for those corpora with varying data density, noisy data and big number of target clusters, some negative effects may appear

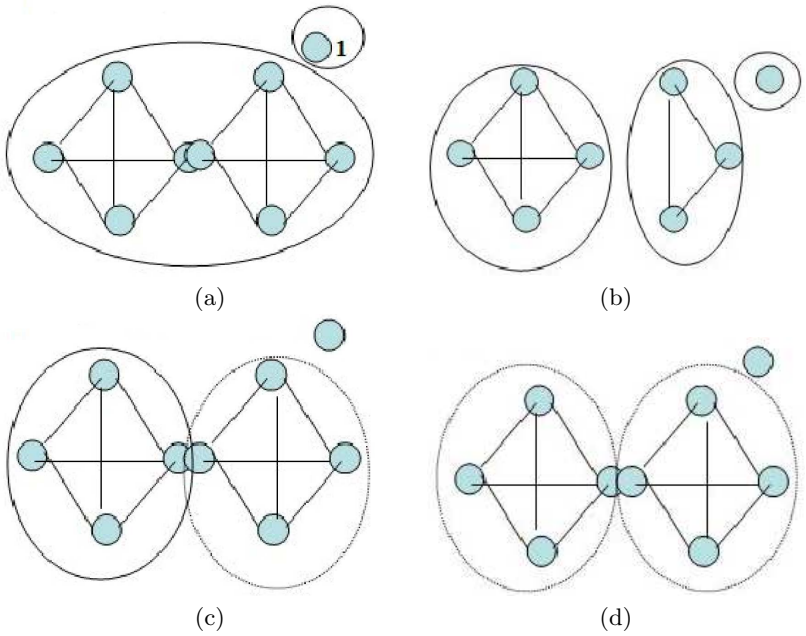

Fig. 1. Butterfly effect in fuzzy clustering. (a) and (b) use classical MajorClust, whereas (c) and (d) use the F-MajorClust approach (with different node membership functions).

[17]. These effects can be mitigated by proper calibration of the membership functions in [13] with the help of fuzzy *if-then* rules and the standard Mamdani-type inference scheme (see [18,19]).

As future work, we plan to compare F-MajorClust with C-Means performance by using the Reuters collection. Moreover, as a basic analysis, it will be also interesting to execute the classical MajorClust over the same dataset.

Acknowledgements

This work has been partially supported by the MCyT TIN2006-15265-C06-04 research project, as well as by the grants BUAP-701 PROMEP/103.5/05/1536, FCC-BUAP, DPI2001-2715-C02-02, MTM2004-07550, MTM2006-10170, and SAB2005-0161.

References

1. MacKay, D.J.: Information Theory, Inference and Learning Algorithms. Cambridge University Press (2003)
2. Mirkin, B.: Mathematical Classification and Clustering. Springer (1996)

3. MacQueen, J.B.: Some methods for classification and analysis of multivariate observations. In: Proc. of 5-th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press (1967) 281–297
4. Kernighan, B., Lin, S.: An efficient heuristic procedure for partitioning graphs. Bell Systems Technical Journal **49**(2) (1970) 291–308
5. Stein, B., Nigemman, O.: On the nature of structure and its identification. Lecture Notes in Computer Science, Springer **1665** (1999) 122–134
6. Dunn, J.C.: A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. Journal of Cybernetics **3** (1973) 32–57
7. Bezdek, J.C.: Pattern Recognition with Fuzzy Objective Function Algoritms. Plenum Press, New York (1981)
8. Stein, B., Busch, M.: Density-based cluster algorithms in low-dimensional and high-dimensional applications. In: Proc. of Second International Workshop on Text-Based Information Retrieval, TIR05. (2005) 45–56
9. Stein, B., Meyer, S.: Automatic document categorization. In: KI 2003: Advances in Artificial Intelligence. (2003) 254–266
10. Alexandrov, M., Gelbukh, A., Rosso, P.: An approach to clustering abstracts. In: Proc. of NLDB 2005 Conference, LNCS 3513, Springer, Verlag (2005) 275–285
11. Pinto, D., Rosso, P.: On the relative hardness of clustering corpora. In: Proc. of TSD 2007 Conference, LNCS, Springer, Verlag (2007) To appear.
12. Neville, J., Adler, M., Jensen, D.: Clustering relational data using attribute and link information. In: Proc. of the Text Mining and Link Analysis Workshop, IJCAI03. (2003)
13. Levner, E., Alcaide, D., Sicilia, J.: Text classification using the fuzzy borda method and semantic grades. In: Proc. of WILF-2007 (CLIP-2007). Volume 4578 of LNAI., LNAI, Springer, Verlag (2007) 422–429
14. Levner, E., Alcaide, D.: Environmental risk ranking: Theory and applications for emergency planning. Scientific Israel - Technological Advantages **8**(1-2) (2006) 11–21
15. Koopmans, T., Beckman, M.: Assignment problems and the location of economic activities. Econometrica **25** (1957) 53–76
16. Singh, S., Sharma, R.: A review of different approaches to the facility layout problem. International Journal of Advanced Manufacutring Technology **30**(5–6) (2006) 426–433 http://dx.doi.org/10.1007/s00170-005-0087-9.
17. Klawonn, F., Höpnner, F.: What is fuzzy about fuzzy clustering-understanding and improving the concept of the fuzzifier. Advances in Intelligent Data Analysis V. (2003) 254–264
18. Mamdani, E.H.: Application of fuzzy logic to approximate reasoning using linguistic synthesis. In: Proc. of the sixth international symposium on Multiple-valued logic. (1976) 196–202
19. Zimmermann, H.: Fuzzy Sets, Decision Making and Expert Systems. Kluwer Academic Publishers, Boston (1987)