Report Series

Fuzzy Voting in Clustering

Evgenia Dimitriadou
Andreas Weingessel
Kurt Hornik

Report No. 29
January 1999

Report Series

January 1999

SFB
'Adaptive Information Systems and Modelling in Economics and Management Science'

Vienna University of Economics
and Business Administration
Augasse 2–6, 1090 Wien, Austria

in cooperation with
University of Vienna
Vienna University of Technology

http://www.wu-wien.ac.at/am

This paper was accepted for publication in:
Proceedings of the "6th International Workshop Fuzzy-Neuro Systems'99", Leipzig, Germany, March 18-19, 1999.

This piece of research was supported by the Austrian Science Foundation (FWF) under grant SFB#010 ('Adaptive Information Systems and Modelling in Economics and Management Science').

Fuzzy Voting in Clustering

Evgenia Dimitriadou **Andreas Weingessel**
Kurt Hornik
Institut für Statistik, Wahrscheinlichkeitstheorie
und Versicherungsmathematik
Technische Universität Wien
Wiedner Hauptstraße 8–10/1071
A-1040 Wien, Austria
Email: *firstname.lastname*@ci.tuwien.ac.at

Abstract

In this paper we present a fuzzy voting scheme for cluster algorithms. This fuzzy voting method allows us to combine several runs of cluster algorithms resulting in a common fuzzy partition. This helps us to overcome instabilities of the cluster algorithms and results in a better clustering.

Keywords: Cluster Algorithms, Fuzzy Partition, Perturbation, Fuzzy Data Sets.

1 Introduction

Partitioning a given population of individuals into "similarity" groups has many applications in science and business. When partitioning individuals into plausible subgroups usually the following main problems are encountered: First, neural network and classical models generally deal with the ideal condition where an input feature belongs to either one class or another. They do not consider cases where an input point has a structure with a certain degree of confidence and may belong to more than one cluster with a certain degree of "belongingness". Second, different cluster algorithms and even multiple replications of the same algorithm result in different solutions

due to random initializations and stochastic learning methods. Every cluster algorithm tries to optimize some criterion, like minimizing the mean-square error. If the task is to find structures in the data, these optimization criteria might not be the appropriate ones. Especially, if the probability distribution of the given data set is completely unknown (generally non-Gaussian), it is not clear which criterion to use. Moreover, the true structure, especially the number and shapes of the clusters, is unknown. Every cluster algorithm is more or less expert finding specific cluster structures, (e.g., k-means creates round clusters) which results to wrong clustering if the data set does not comply with the structure that an algorithm tends to create.

We develop an algorithm which allows a fuzzy decision between several results of a cluster algorithm. Specifically, we introduce an artificial, not random, disturbance to the data set, according every time to the result of the voting part of the algorithm. The perturbed data set can influence the clustering algorithm to create more different cluster structures that also explain adequately the structure of the original data set, but due to the already mentioned problems, a simple clustering algorithm is incapable of producing them. In this way, we notice that the fuzzy voting algorithm results in an improved result. The result of our voting is a fuzzy partition of the data.

The paper is organized as follows. In Section 2 we present the fuzzy voting scheme. Section 3 demonstrates our experimental results and some comments on them. A conclusion of the paper is given in Section 4.

2 Fuzzy Voting Scheme

2.1 Fuzzy Voting

We developed the following algorithm [1]. Our input data x is clustered several times. Let C_{ij} denote the jth cluster in the ith run and D_{ij} denote the jth cluster in the combination of the first i runs. The first two runs are combined in the following way. First a mapping between the clusters of the two runs is defined. To do this we compute for each cluster C_{2j} how many percent of its points have been assigned to which cluster C_{1k}. Then, the two clusters with the highest percentage of common points are assigned towards each other. Of the remaining clusters, again the two with the highest similarity are matched and so on. After renumbering the clusters of the second run so that C_{2j} corresponds to $C_{1j}, \forall j$, we assign the points to the common clusters D_{2j} in the following way. If a data point x has been

assigned to both C_{1j} and C_{2j} it will be assigned to D_{2j}. If x has been assigned to C_{1j} in the first run and to C_{2k} with $j \neq k$ in the second run then it will be assigned to both, D_{2j} and D_{2k}, with weight 0.5.

If we have already combined the first n runs in a common clustering D_{nj} we add an additional run $C_{(n+1)j}$ by combining it with D_{nj} in the same way as for two runs, but give weight $n/(n+1)$ to the common clusters of the first n runs and weight $1/(n+1)$ to the new cluster.

2.2 Partition Resulting from the Fuzzy Voting

For the results of the voting algorithms the data points are typically not uniquely assigned to one cluster, but there is a fuzzy partition. That is, after voting of N runs we get for every data point x and every cluster j a value D_{Nj} which gives the fraction of times this data point was assigned to this cluster. For the final result we assign every data point to that cluster $k = \text{argmax}_j(D_{Nj})$ where it has been assigned most often. We define the *sureness* of a data point as the percentage of times it was assigned to its "winning" cluster k, that is $sureness(x) = \max_j(D_{Nj})$. Then we can not only see how "strongly" a certain point belongs to a cluster but we can also compute the average sureness of a cluster ($Avesure$) as the average sureness of all the points of a cluster that belong to it.

2.3 Perturbation

In order to find the data structure with the help of the voting algorithm, we need several different results of various runs of the cluster algorithm. Experimenting with the voting part of the algorithm, we noticed that voting can not overcome some problems connected with the process of the simple clustering methods and also it is appearing some of its own. That means that if the cluster algorithm is too stable, or a bit unstable, voting always yields the same result and gives exactly the same one. It may also happens, due to random initializations and to the cluster process itself, that a clustering result may never appear due to close decisions being made against it. Moreover, in the same way, in the case of an unstable clustering method that produces just a few clustering results, it exists always the risk that the voting decision for the result can be on the limit.

To overcome these problems we introduce artificial instabilities in the data set by perturbing the points. These perturbations are not at random,

but we use the current result of the voting steps to decide which points lie in between the clusters and need "help" to find the cluster they belong to. The idea is that the "polyphony" of the produced results by perturbing the data set makes the voting more flexible and the results more optimized.

Perturbation of the data points is taking place in the following way[1]. After every voting step every data point has a fuzzy membership to each cluster. We perturb a point from its original place towards all clusters except for its "winning" one, analogous to its probability to belong to a specific cluster, and to the minimum distance of the point from every fuzzy cluster center. This vector product result is also multiplied by the difference of the mean *Avesure* of all the clusters subtracted from 1. This stands for a stable learning rate.

3 Experimental Results

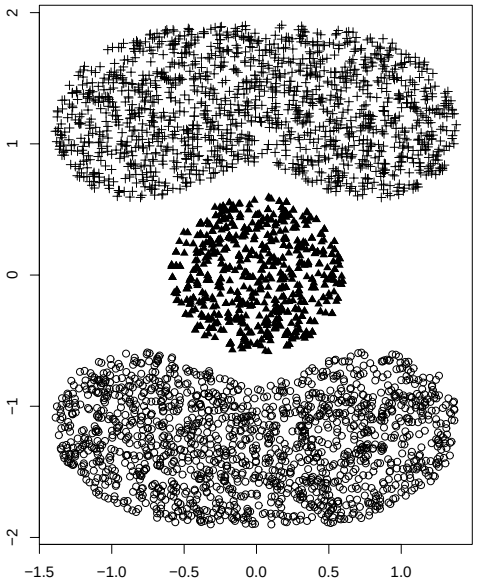
We used different cluster algorithms in our experiments, like k-means (also known as LBG algorithm [2]) as an example of an off-line algorithm, hard competitive learning as an online version of k-means (see for example [3]), and neural gas [4]. All of the above algorithms are not able to indicate or propose the appropriate solution in the case of instable results, and moreover they show local minima problems due to dependencies on the initial conditions of the simulations or on the choice of the learning rate. Our purpose was to study up to what extend the fuzzy scheme can overcome these instabilities. Note that no comparison of our scheme is being made with already existing fuzzy clustering methods. The reason is that we do not introduce a fuzzy clustering method, but fuzzy decision (voting) between several clustering results of a data set. Moreover, we can extend our voting scheme also to voting between the results of several fuzzy cluster algorithms.

Due to space constraints we only present the results of voting with the k-means algorithm. The results for the other algorithms are similar. The experiments presented here consist of 100 runs of one algorithm and a voting between these runs. Since we are, for our data sets, aware of the true cluster structure, we can treat the result of a cluster algorithm as a classification problem (although we do not use the class information during clustering). That is, we can compute how many points have been assigned by the cluster algorithm to the right cluster. The results given in Tab. 3[2] are the mean value

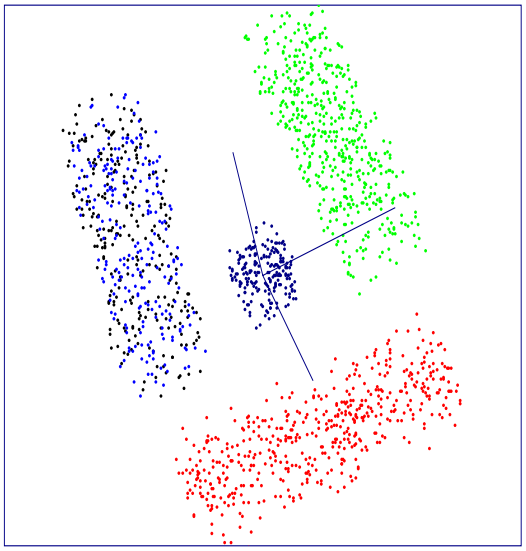
[1]Note that depending on the data set various versions of this perturbation may yield a better result. The one we present here, performs well independently from a data set.

[2]Note that significance tests have been made between the perturbed and non perturbed

and standard deviation of the results of 100 k-mean runs and 100 voting, with and without perturbation, ones. Additionally, we show the classification rate for the k-means run which yields the minimum error.



(a) 2-Dimensional Data Set



(b) 3-Dimensional Data Set

Figure 1: Artificial Data Sets

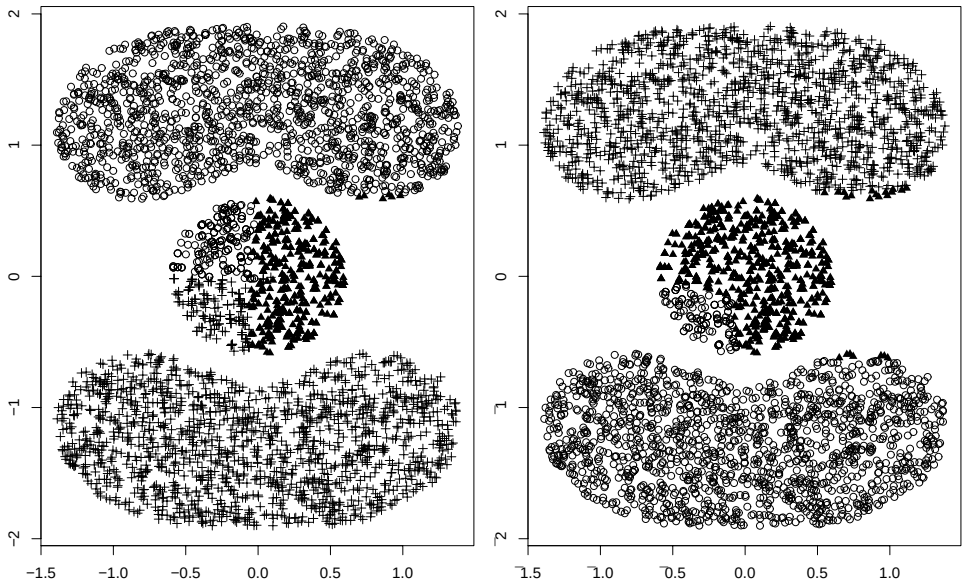
3.1 A 2-Dimensional Example

This data set is 2-dimensional with data points uniformly distributed within the structures seen in Fig. 1(a). The middle structure consists of 900 data points whereas both the external ones consist of 1500 points. Typical results of k-means clustering and the result which yielded the minimum error in all 100 runs show that k-means divides one of the two big structures into two clusters, thus ignoring the small cluster in the middle.

In only about 10 times in every 100 runs k-means had the optimal clustering result, but there is no way to detect these results, because they do not have the minimum error. The result of the voting algorithm can be seen in Fig. 2(a). We see that the two big structures have been found, the third

fuzzy voting results and it was proved that they are significantly different.

cluster has been placed in the middle structure. Voting with perturbation resulted to an almost perfect clustering result in particular for the middle structure (see Fig. 2(b)).



(a) Voting Result

(b) Result of Voting with Perturbation

Figure 2: 2-D Results

In Tab. 3 we show the average number of correctly classified points for single runs of k-means and for the voting algorithm. We see that on average, k-means misclassifies 26% of the points, whereas voting can almost halve the misclassification rate to 15%. The standard deviation of the repetitions of the k-mean runs is more than 2 times higher than the standard deviation between the voting runs. This shows that voting yields more stable results. Voting with perturbation yields a better classification rate being more than 3% increased. Tab. 1 shows the average sureness of the clusters. Clusters 1 and 2 correspond to the big structures, which are more "sure" than the third cluster in the middle.

Table 1: 2-D: Average Sureness of the Clusters

No.clusters	1	2	3
Voting	0.75	0.81	0.61
Pert.Voting	0.78	0.80	0.66

3.2 A 3-Dimensional Example

This data set is a 3-dimensional one with data points points uniformly distributed within four cuboids, see Fig. 1(b). The small cube in the middle consists of 200 data points and all the others of 400 to 500 points. In Tab. 3 we see that k-means again has difficulties in finding the right clusters, in particular the small cube, on average it classifies only 87% of the points correctly. Voting, however, finds the cluster structure. In Tab. 2 we show the *Avesure* of the 4 clusters. The first 3 clusters correspond to the 3 big structures, their *Avesure* is between 84 and 85%. The small cluster in the middle is often split into several parts, therefore its *Avesure* is significantly smaller. Voting with perturbation almost perfectly finds the data structure (see Fig. 3).

Table 2: 3-D: Average Sureness of the Clusters

No.clusters	1	2	3	4
Voting	0.84	0.84	0.85	0.58
Pert.Voting	0.92	0.81	0.84	0.65

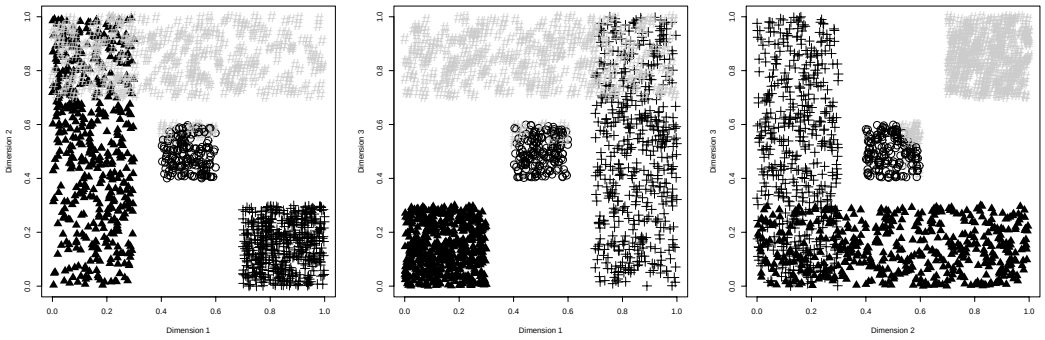


Figure 3: Result of Voting with Perturbation

3.3 Segmentation Data Set

As a real world data set we use an image segmentation dataset [5]. This data set consists of 2310 subimages of outdoor images which are represented by 19 continuous attributes and belong to 7 different classes, like grass, sky, ... Basically, this segmentation data set is considered as a classification problem because the true classes are known. Due to the fact that there are no well known benchmark data sets for cluster problems, we treat the problem as a clustering one ([6], page 202).

The results show a statistically significant improvement from k-means to voting and from voting to fuzzy voting. The improvement, however, is not too big, because k-means yielded not very unstable results and because there are problems with overlapping clusters which can not be separated without the class information.

Table 3: Data Sets: Correct Classification

	k-means			voting		pert.voting	
	mean	std	class (min err)	mean	std	mean	std
2-Dimensional	74.08	10.30	65.21	85.08	4.98	88.23	6.87
3-Dimensional	87.49	4.27	89.56	93.57	3.73	97.24	3.20
Segmentation	55.17	3.20	57.44	57.53	1.05	57.90	0.42

4 Conclusion

In this paper we present a fuzzy method of combining the results of several independent cluster runs by voting between their results. We present a way of perturbing the original data set in order to supply the voting algorithm with a bigger variety of solutions and optimize its performance. This allows us to deal with the problem of local minima of cluster algorithms and to find a partition of the data which is supported by repeated applications of the cluster algorithm not influenced by the randomness of initialization or the cluster process itself. Consequently, this fuzzy perception of the data leads us to further research. We see two possible ways how the results of voting can be used for further improvements of cluster algorithms. First, it can help to handle the difficult problem of finding the right number of clusters in a data

set. Preliminary experiments show that, if we use more clusters in the cluster algorithm as there are in the data set, voting results in almost empty or very unsure clusters thus indicating that fewer clusters are necessary. Secondly, we can identify the "unsure" data points which lie somewhere in between two or more clusters. We can apply additional algorithms which concentrate on those "unsure" points to make the decision more clear.

Acknowledgments

This piece of research was supported by the Austrian Science Foundation (FWF) under grant SFB#010 ('Adaptive Information Systems and Modeling in Economics and Management Science').

Literature

[1] Andreas Weingessel, Evgenia Dimitriadou, and Kurt Hornik. A voting scheme for cluster algorithms. In *Proceedings of the "Fourth International Workshop Neural Networks in Applications'99"*, Otto-von-Guericke University of Magdeburg, Germany, 1999. To appear.

[2] Yoseph Linde, Andrs Buzo, and Robert M. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, COM-28(1):84–95, January 1980.

[3] Bernd Fritzke. Some competitive learning methods, April 5 1997. http://www.neuroinformatik.ruhr-uni-bochum.de/ini/VDM/research/gsn/.

[4] Thomas M. Martinetz, Stanislav G. Berkovich, and Klaus J. Schulten. "Neural-Gas" network for vector quantization and its application to time-series prediction. *IEEE Transactions on Neural Networks*, 4(4):558–569, July 1993.

[5] D. Michie, D. J. Spiegelhalter, and C. C. Taylor, editors. *Machine Learning: Neural Statistical Classification.* Ellis Horwood, 1994.

[6] Hans-Joachim Mucha. *Clusteranalyse mit Mikrocomputern.* Akademie Verlag, 1992.