Working Paper Series

A Voting-Merging Clustering Algorithm

Evgenia Dimitriadou
Andreas Weingessel
Kurt Hornik

Working Paper No. 31
April 1999

Working Paper Series

April 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 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').

A Voting-Merging Clustering Algorithm

Evgenia Dimitriadou **Andreas Weingessel**
Kurt Hornik
Institut für Statistik,
Wahrscheinlichkeitstheorie & 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 propose an unsupervised voting-merging scheme that is capable of clustering data sets, and also of finding the number of clusters existing in them. The voting part of the algorithm allows us to combine several runs of clustering algorithms resulting in a common partition. This helps us to overcome instabilities of the clustering algorithms and to improve the ability to find structures in a data set. Moreover, we develop a strategy to understand, analyze and interpret these results. In the second part of the scheme, a merging procedure starts on the clusters resulting by voting, in order to find the number of clusters in the data set.

Keywords: Cluster Algorithms, Unsupervised Learning, Partition, Number of Clusters.

1 Introduction

Clustering is the partitioning of a set of objects into groups so that objects within a group are "similar" and objects in different groups are "dissimilar". Thus, the purpose of clustering is to identify "natural" structures in a data set. In real life clustering situations usually the following main problems are encountered: First, the true structure, especially the number and shapes of the clusters, is unknown. Second, different cluster algorithms and even multiple replications of the same algorithm result in different solutions due to random initializations and stochastic learning methods. Moreover, there is no clear indication which of the different solutions of the replications of the algorithm is the best one.

Every cluster algorithm tries to optimize some criterion, like minimizing the mean-square error. If the task of clustering is to compress the data by mapping every data point to a prototype, then the minimization of an appropriate error measure is the right thing to do, but if the task is to find structures in the data, these optimization criterion might help to find a good solution, but they are not necessarily the right measures to optimize. Especially, if the (generally non-Gaussian) probability distribution of the given data set is completely unknown, it is not clear which criterion to use.

To tackle these problems, we handle results of various runs by using the existing idea of voting in classification (Breiman, 1996; Freund and Schapire, 1995). We develop an algorithm which allows a combination between several results of a cluster algorithm (voting) resulting in a common partition. The idea is that the voting procedure can be applied to any existing algorithm that has instable results. The result of our voting is a "fuzzy" partition of the data. We propose the idea that there are 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". With this method the output of several single classifiers can be combined so as to reduce the variance of the error between the different runs and to get an overall decision made by the combined classifiers. Additionally, voting can react to the tendency of every algorithm to create clusters with a specific kind of structure (e.g., k-means is creating round clusters) and may result also to non convex ones.

In the following steps, we take advantage of that "fuzzy" partition of the data to introduce some new measures for handling and understanding these results, and to develop a method for finding the right number of clusters in a data set. This technique is followed by a merging procedure, where clusters resulting from voting are merged according to the highest probability of their data points to belong to another cluster. This procedure continues until every cluster is merged, and then a decision according to some criteria is taken in order to specify the optimal number of clusters in the data set.

Consequently, our Voting-Merging Algorithm (VMA) represents a complete scheme of a clustering algorithm, able to partition data points of a set in clusters and also to find the optimal number of classes existing in the set.

The paper is organized as follows. In Sections 2 and 3 we present the Voting-Merging Algorithm (VMA) and its implementation. Section 4 demonstrates our experimental results and some comments on them. A comparison with other clustering algorithms follows in Section 5, and finally a conclusion of the paper is given in Section 6.

2 Description of the VMA

Generally, the VMA is a scheme consisting of 3 procedures, namely the repeated runs of a clustering algorithm, the voting procedure receiving these results, and finally the merging procedure which concludes finding the number of clusters in a data set. The 3 levels of the algorithm are applied sequentially. They do not interfere with each other, but they just receive the results from the previous levels. No feedback process happens, and the algorithm terminates after the completion of all procedures.

2.1 Voting Procedure

In classification, there is a fixed set of labels which are assigned to the data. Therefore, we can compare for every input x the results of the various classifiers, i.e., the labels assigned to x and apply a voting procedure between these results. Things are different in clustering, because different runs of a cluster algorithm can result in different clusters, which might partition the input data in totally different ways. Thus, there is the problem to decide which cluster of one run corresponds to which in another run.

As an example, suppose that we have the results of several runs i of cluster algorithms which partition our data into 3 clusters $C_{ij}, j = 1, 2, 3$. When combining the first two runs we have to decide which of the clusters C_{11}, C_{12}, C_{13} of the first run is similar to which cluster C_{21}, C_{22}, C_{23} of the second run, if there is any similarity at all. Note that as opposed to classification the numbers $1, 2, 3$ are arbitrarily assigned to the clusters and their order can be interchanged. If we combine more than 2 runs, the additional difficulty arises that the similarity relation is not transitive. That is, cluster C_{11} of the first run might be similar to cluster C_{22} of the second run which might again be similar to cluster C_{31} of the third run and so on, but this does not mean that C_{31} is again similar to C_{11}. It might even turn out that C_{31} is more similar to C_{12} for example.

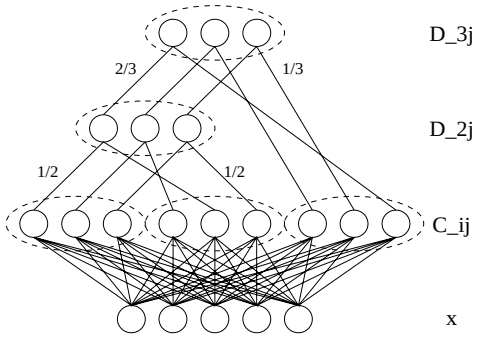


Figure 1: The Voting Procedure

Since there seems to be no obvious way for combining more than two runs at once, we developed the following procedure, which is depicted by the network in Figure 1. Our input data x is clustered several times, the second layer (C_{ij}) of Figure 1 symbolizes three runs with three clusters each. 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. Our procedure works on an iterative basis. 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. This step is shown in the connection between the second layer (C_{ij}) and third layer (D_{2j}) in Figure 1. Every cluster of D_{2j} receives input from exactly one cluster of the first run and one of the second run. The weights of these connections is set to $\frac{1}{2}$.

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.

Note that this voting procedure gives equal weights to all repetitions of the cluster algorithm and there is no additional gating network (Weigend et al., 1995) which learns to decide between the different results. The reason is that, as opposed to supervised learning, we can not decide which of the various results is the best.

2.2 The Partition Resulting from the Voting Procedure

For the results of the voting procedure 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 $D_{Nj}(x)$ which gives the fraction of times x has been assigned to cluster j. For interpreting the final result we can either accept this fuzzy decision or assign every data point x to that cluster $k = \text{argmax}_j\, D_{Nj}(x)$ where it has been assigned most often. We define the *sureness* of a data point as the percentage of times it has been assigned to its "winning" cluster k, that is $sureness(x) = \max_j D_{Nj}(x)$. Then we can not only see how strong 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, i.e., $avesure(k) = \text{mean}_{x \in k}\, D_{Nk}(x)$. In this way we can notice which clusters have a clear data structure and which not.

2.3 Merging Procedure

As already mentioned in the previous section, after voting every data point x belongs to more than one cluster j with a certain degree of "belongingness" $D_{Nj}(x)$. If we set $n(k,j) := \text{mean}_{x \in k}\, D_{Nj}(x)$ we get a measure of how strong the points of cluster k belong to cluster j. Thus, $n(k,j)$ defines a (non-symmetric) neighborhood relation between two clusters. So, we can say that a cluster l is the closest cluster to cluster k, if $n(k,l) = \max_{j \neq k}\, n(k,j)$[1].

We can use this neighborhood relation to develop a merging procedure that starts with many clusters and merges clusters which are closest to each other. More specifically, two clusters k and l are merged together, if k is the closest cluster to l and l is the closest cluster to k. Additionally, we merge "chains" of clusters. That is, a set of clusters $k_1, k_2, \ldots, k_n$ is merged if k_{i+1} is the closest cluster to k_i $(i = 1, \ldots, (n-1))$ and k_1 is the closest cluster to k_n.

After the merging step, the *sureness* of the points and *avesure* of the clusters are recomputed by adding up the values of the merged clusters. Then, the merging step repeats until some stopping criterion is met.

The criterion that derives directly from this procedure is that merging will stop when all clusters end up to have an *avesure* of 100%. Since in practice this condition is not always possible to be realized, for example due to number precision problems, we decide to stop the merging procedure when the *avesure* of every cluster is greater than 99%. That is, on average every point is assigned less than once to a "wrong" cluster, if voting has been applied with 100 runs. Thus, we have in that case a significant decision that every point belongs to the "right" cluster. In this case the probability of a cluster to be merged with another is extinguished, which means that the procedure terminates normally. Unfortunately, this happens only in

[1] Note that cluster k is not necessarily the closest cluster to cluster l.

the case where a simple and clear structure, especially a non overlapping one, exists in a data set. In other situations clusters are able to reach a "very sure" structure (of more than 90%) but not the one of 100% *avesure*. Note that clusters with one data point are not considered as clusters but the point is partitioned to the next cluster with the highest probability to "win" it. That is due to the fact that, obviously, one point clusters can become very easily 100% "sure" and stop merging.

Since it is non realistic in real-world data sets to reach a 100% "sure" solution, it is still probable to happen that the "real" clusters existing in a data set will increase all the time their *avesure* during their construction by the merging procedure and the rest will stay quite "unsure" in comparison. The problem logically arising is that after reaching by merging the "real" clusters existing in the data set, every merging step will also continue increasing the mean *avesure* of all the clusters and consequently it is not always clear when the merging should be stopped. However, this increase that takes place is not significant, because the clusters are already to that point quite "sure".

Thus, to tackle these problems we suggest the following criterion (see Dimitriadou et al., 1999):

$$devsure(i) := [numsure(i) - numsure(i-1)] - [numsure(i+1) - numsure(i)]$$

where $numsure(i)$ is the sum of the $sureness(x)$ of all points in all the n clusters in the ith step of the merging. This second order difference shows how much the solution for the n clusters existing in the i step of merging deviates from the general increase (when n decreases) of the *sureness*. The solution with the maximum value for $devsure(i)$ is the one whose cluster structure found by voting and merging is the optimal for the algorithm.

3 Pseudo-algorithm and Implementation

3.1 Pseudo-algorithm

STEP 1: Voting

- i=1; Run cluster algorithm to obtain clusters C_{1j}; $D_{1j} = C_{1j}$.
- for (i in 2:N)
  - Run cluster algorithm to obtain clusters C_{ij}
  - Find matching between C_{ij} and $D_{(i-1)j}$.
  - $D_{ij} = \frac{(i-1)}{i} D_{(i-1)j} + \frac{1}{i} C_{ij}$
- Assign to cluster $\text{argmax}_j(D_{Nj})$
- *Sureness* of a point $\max_j D_{Nj}$
- Average sureness (*Avesure*) of a cluster is the mean sureness of its points

STEP 2: Merging

- Repeat
  - Compute $n(k,j) := \text{mean}_{x \in k} D_{Nj}(x)$ for all pairs of clusters k and j.
  - Find pairs and chains of clusters $k_1, \ldots, k_n$ to merge
  - Merge clusters $k_1, \ldots, k_n$ to cluster k, $D_{Nk} = \sum_{k_i} D_{Nk_i}$.
- Until stopping criterion is met

3.2 Implementation

We used different clustering algorithms in our experiments, like k-means (also known as LBG algorithm, Linde et al. (1980)) as an example of an off-line algorithm, hard competitive learning as an online version of k-means (see for example Xu et al., 1993) and neural gas (Martinetz et al., 1993).

The experiments presented here consist of 100 runs of one clustering algorithm with a big number of clusters specified in the beginning. There is no formula to determine the value of this number, but experimental results show that it should be bigger proportional to the size of the data set. This is logical, in the way that if a small data set is clustered with a big number of clusters then it will be unavoidable that we will have many clusters of few points (for example 2 points) and also many empty ones. In the case of too many clusters we have the problem of having in the beginning of the algorithm structures being 100% "sure" with only very few points (for example 2), something that does not correspond to reality, when in the case of many empty clusters, it makes simply the decision of clustering with a big number of clusters meaningless. Moreover, in the case of big data sets it is proposed to cluster them with many centers, to avoid points belonging to different clusters but being close neighbors to be clustered together. For the same reason that the number of the clusters, with which a clustering algorithm runs, should be also big the more overlapping the structures of the data sets appear to be, for example generally the real-world data sets.

After this step a voting between these runs follows and then a merging according to voting results. Merging stops when we reach the 2 clusters in a data set. 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, (Mucha, 1992, page 202), 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.

All our experiments have been performed in R, a system for statistical computation and graphics, which implements to well-known S-language for statistics. R runs under a variety of Unix platforms (such as Linux) and under Windows95/98/NT. It is available freely via CRAN, the Comprehensive R Archive Network, whose master site is at `http://www.ci.tuwien.ac.at/R`.

4 Experimental Results

Artificial data sets as well as real-world sets are used to demonstrate the performance of the VMA. A description of these data sets, some figures and tables, as well as some comments on

the results, follow in order to make clear and demonstrate the performance of the algorithm.

In Tables 1, 2, 4, 5 and 6 we show the *Avesure* of every cluster in the data set for a typical run of voting. The results given in Table 7 are the mean value for the classification rate of the results of 100 runs of the cluster algorithm (for the right number of clusters in the data sets) and of a typical VMA one. Each voting run has been performed with the results of 100 new cluster runs.

Furthermore, in Figure 3(a) we see the mean average sureness of a typical result of the VMA for every merging step (and respectively for every number of clusters in every step), where as in Figure 3(b) we see the respective values of the decision measure for some data sets.

We present only the results of voting with one particular cluster algorithm, because the results for the other algorithms are similar.

4.1 A 2-Dimensional Example

This data set is also a 2-dimensional one, and it is a simplification of the one described and used in Mao and Jain (1995). It consists of 2000 data points belonging to 2 elongated, curved clusters with opposite curvature (see Figure 2(a)). The two curves with 1000 data points each are generated in the following way; one according to $y = 2(x-1)^2 + \epsilon$ where the x are uniformly distributed in $[0, 2]$ and ϵ is a normally distributed noise term with zero mean and standard deviation 0.1; the other is $y = 3 - 2(x-2)^2 + \epsilon$ with x uniformly distributed in $[1, 3]$ and ϵ as above. This data set is not linearly separable.

After voting between 100 results of hard competitive learning runs (starting with 30 clusters), merging also results partitioning perfectly the data points to their clusters (100% classification rate, see Table 7). Here, since we reach 100% "avesure" clusters merging stops without the help of the criterion (see Table 1).

Also by this data set, hard competitive single runs, with the number of the 2 clusters specified, can never find the two curved clusters by misclassifying 10% of the data points on average (see a typical result of a hard competitive run in Figure 2(b)).

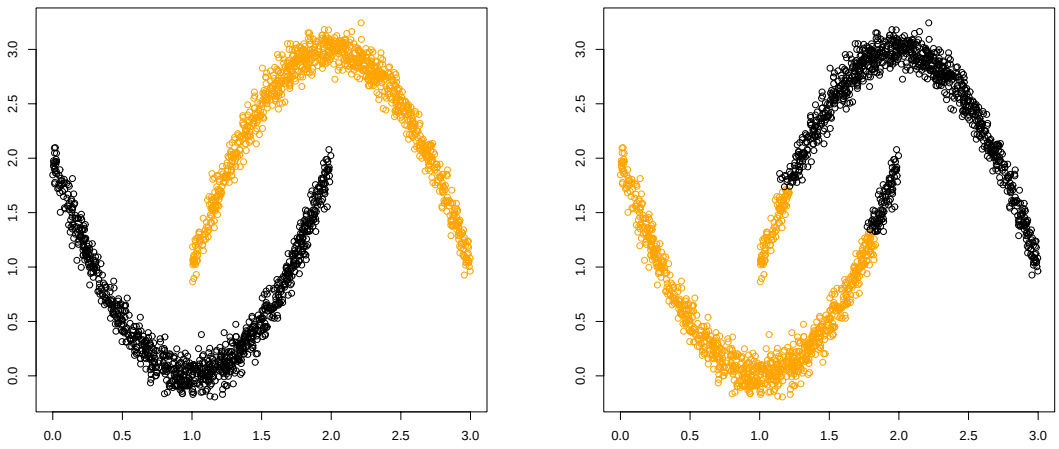
No.clusters	1	2
Size.clusters	1000	1000
Avesure	99.76	99.27

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

4.2 A 3-Dimensional Example

This data set is a 3-dimensional one with data points uniformly distributed within a small cube in the middle and 3 big cuboids surrounding it. The cube in the middle consists of 200 data points and all the other of 500 points. The difficulty of this set is introduced by the significantly smaller shape of the cluster in the middle surrounded by the other.

After voting between 100 results of k-means runs (starting with 30 clusters), merging also results partitioning perfectly the data points to their clusters (100% classification rate, see



(a) 2-Dimensional Data Set

(b) Clustering Result

Figure 2: Elongated Curves

Table 7). Here, since we reach 100% "avesure" clusters merging stops without the help of the criterion (see Table 2).

Also by this data set, k-means single runs, with the number of the 4 clusters specified, have difficulties finding the structure by misclassifying 15% of the data points on average (for more details see Weingessel et al. (1999b)).

No.clusters	1	2	3	4
Size.clusters	500	500	500	200
Avesure	100.00	100.00	100.00	100.00

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

4.3 A Binary Data Set

One of our current research projects is the analysis of binary marketing data. Therefore, we experiment also with a 12-dimensional artificial binary scenario that is designed in the following way (see Table 3): we have 12 binary variables belonging to 4 groups of 3 variables each. There are 6 types of data points in the data set. Each type has for the variables of 2 of the 4 variable groups a high (H) probability (0.8) to have a 1 there and for the other variables a low (L) one (0.2). In the experiment the sizes of the 6 types (and therefore the sizes of the expected clusters) are 3000, 1000, 700, 500, 500, and 300 respectively. This scenario has an important difficulty level since the sizes of the clusters differ significantly and moreover

the clusters are overlapping. Thus a 100%-correct classification is impossible. The Bayes classification rate, which is the performance of the theoretically best classifier for a given probability distribution, is 88.55% for this data set.

We perform the voting procedure between 100 clustering results of the hard competitive clustering algorithm (starting with 70 clusters). The VMA succeeds in detecting the right number of clusters in the scenario reaching a classification rate of 86.17%, when hard competitive learning reaches a 79% classification rate (see Table 7) on average in 100 runs with the right number of 6 clusters being specified. For a more detailed description of this (and other) artificial binary data sets, see Dolnicar et al. (1998). Moreover, we see in Table 4 that the most "unsure" clusters appear to be the smallest (300 points) and the biggest (3000 points), which is expected since due to their sizes the small tends to disappear and the big to "win" all the points. In Figures 3(a) we see the mean average sureness of all clusters of a typical result of a voting run for every number of clusters, where as in Figure 3(b) we see the respective values of the decision measure for every data.

	G1			G2			G3			G4			
Type	I1	I2	I3	I1	I2	I3	I1	I2	I3	I1	I2	I3	n
1	H	H	H	H	H	H	L	L	L	L	L	L	1000
2	L	L	L	L	L	L	H	H	H	H	H	H	300
3	L	L	L	H	H	H	H	H	H	L	L	L	700
4	H	H	H	L	L	L	L	L	L	H	H	H	3000
5	L	L	L	H	H	H	L	L	L	H	H	H	500
6	H	H	H	L	L	L	H	H	H	L	L	L	500

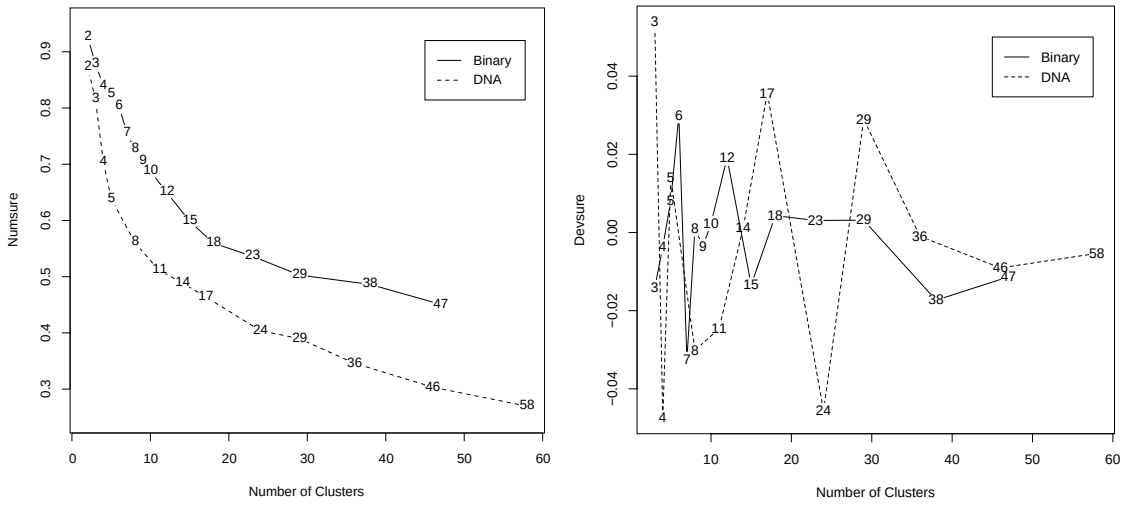
Table 3: Binary Data Set

No.clusters	1	2	3	4	5	6
Size.clusters	1000	300	700	3000	500	500
Avesure	0.91	0.74	0.93	0.81	0.91	0.92

Table 4: Binary Data: Average Sureness of the Clusters

4.4 Real-World Data Set: DNA

We apply a clustering algorithm to the DNA data set (see Michie et al., 1994). Basically, the DNA set is considered as a classification problem because the true classes are known. But because there are no well established benchmark data sets for cluster problems, we treat the problem as a clustering one. It consists of 2000 data points (splice junctions) on a DNA sequence at which "superfluous" DNA is removed during protein creation. The data points are described by 60 indicator binary variables and the problem is to recognize the 3 classes (ei, ie, neither), i.e., the boundaries between exons (the parts of the DNA sequence retained after splicing) and introns (the parts of the DNA sequence that are spliced out).



(a) Mean Average Sureness

(b) Measure Values

Figure 3: Finding the Number of Clusters

VMA also here succeeds in detecting the right number of clusters in the data set with a classification rate of 85.56% (see Table 7). The voting procedure has been performed on 100 runs of the hard competitive algorithm (starting with 90 clusters). In Figures 3(a) we see the mean average sureness of all clusters of a typical result of a voting run for every number of clusters, where as in Figure 3(b) we see the respective values of the decision measure for every data.

Name.clusters	ei	ie	neither
Size.clusters	465	1051	464
Avesure	82.66	83.58	76.06

Table 5: DNA: Average Sureness of the Clusters

4.5 Iris Data Set

We apply a cluster algorithm to the Iris data set (see for example Mucha, 1992, page 131). The reason of using this data set is to apply the VMA to a real and especially very well known data set. Basically, the Iris set is also considered as a classification problem because the true classes are known. We also treat this problem as a clustering one (Mucha, 1992, page 202).

VMA finds the right number of clusters with a misclassification rate of 9.33% (see Table 7), which is inevitable since the two classes in the data set (versicolor and virginica) are overlapping. Due to the same reason these two clusters conclude not having 100% *avesure*, see Table 6. Voting has been performed from 100 runs of the hard competitive algorithm.

It is important to note that single hard competitive results have on average a classification rate of 76% with high (round 14) standard deviation in 100 runs on average, with the number of the 3 clusters being specified from the beginning (for more details Weingessel et al. (1999b)).

Name.cluster	setosa	versicolor	virginica
Size.cluster	50	50	50
Avesure	100.00	99.57	98.28

Table 6: Iris: Average Sureness of the Clusters

	clust.alg.	VMA
Data Sets	class.rate	class.rate
2-Dimensional	90.20	100.00
3-Dimensional	85.18	100.00
Binary Scenario	78.10	86.17
DNA	76.02	85.56
Iris	76.20	90.67

Table 7: Data Sets: Classification Rate

5 Comparative Performance

In real life clustering situations a researcher in the beginning is confronted with crucial decisions like choosing the appropriate clustering method and selecting the number of clusters in the final solution. Both are considered to be unsolved problems of great significance as the result and the success of a research depends on these decisions. Numerous strategies have been proposed for clustering and finding the "right" number of clusters.

First, we refer to some index measures known in the literature as stopping criteria, that can be applied on every clustering algorithm in order to find the number of clusters in a data set. Monte Carlo evaluations of these indexes have been conducted by researchers in order to analyze and compare their performance However, it has been shown that also the best of them are not able of finding the "right" number of clusters in the majority of the experimental runs (see Milligan and Cooper, 1985; Milligan, 1981, 1980; Weingessel et al., 1999a).

Further on, in the literature, other measures and criteria have been proposed where contrary to the stopping criteria, they depend on a specific clustering algorithm either as being a part of the algorithm or being calculated by specific measures that the algorithm is using. All of these algorithms are able of clustering and finding the structure in a data set, but many of them suffer from several difficulties during their processes; they are highly sensitive to the initializations, different initial conditions lead to different results, like for example RPCL (Xu et al., 1993), FSCL (Marazzi et al., 1996), ISODATA (Ball and Hall, 1965), Begovich algorithm (Begovich and Kane, 1982). They also suffer from inabilities to handle variabilities in cluster shapes (especially problems with non-convex structures, see RPCL, Xu et al. (1993)),

cluster densities (usually the points in a data set are not equally dense considering in this way many times the least dense clusters as noise, (see Yin and Chen, 1994), cluster sizes (especially when a data set consists of very big and very small ones see FSCL, Marazzi et al. (1996)). Additionally, generally all these versions of competitive learning algorithms (known as Winner-Take-All networks), are also not able to indicate or propose the appropriate solution in the case of unstable results, and they show local minima problems due to dependencies on the initial conditions of the simulations or on the choice of the learning rate.

Moreover, most of the algorithms which prove to perform very satisfactory in clustering and finding the number of clusters meet the problem of defining in the beginning of the algorithms some similarity thresholds, see for example Brown et al. (1991), Chaudhuri et al. (1992), also the CDL (Eltoft and DeFigueiredo, 1998), the ART structures (Carpenter and Grossberg, 1986), OI (Optimal Interpolating) and OS (Optimal Smoothing) Networks (DeFigueiredo, 1996).

Further on, one confronts also the problem that many clustering algorithms that find the number of clusters in a data set, presuppose some specific conditions, for example data belonging to specific distributions (see Xu, 1997; Wallace and Kanade, 1990; Biernacki et al., 1997).

Finally, talking about only the performance of clustering algorithms, deterministic cluster algorithms, like hierarchical cluster methods, and generally methods using pairwise distances, usually can not deal with large data sets (i.e., one is confronted with memory and time problems), whereas competitive learning methods like k-means (Linde et al., 1980), hard competitive learning (see for example Xu et al., 1993), neural gas (Martinetz et al., 1993), confront also problems of initializations and of different cluster sizes and structures, while they perform poorly in the case of noisy or overlapping data (for more details Weingessel et al. (1999b)).

Our purpose is to study how far the VMA can overcome these main difficulties and disadvantages of the clustering algorithms. K-means, hard competitive learning and many other existing algorithms produce a big range of different clustering results. As shown in Weingessel et al. (1999b) the voting procedure reduces very significantly the standard deviation between different runs on the same data set and it is stable in the sense that in the repeated partitioning, many patterns always move together. The merging process that follows turns out to be absolutely stable providing the whole algorithm the ability to preserve the stability of the results. Also, we see that voting is able of finding the structures in a data set, no matter if the clusters have different sizes, densities, or are overlapping. Moreover, there are no similarity thresholds, but just a parameter defining in the beginning of the algorithm the starting number of clusters. This is still a disadvantage for the algorithm since different initial number of clusters are possible to lead to different results. Finally, the VMA is not confronting memory problems, but it takes more running time than a simple clustering algorithm due to the repetitions in the beginning of a clustering method and then the voting procedure between these runs.

6 Conclusion

In this paper we present an unsupervised scheme, able to cluster data points in a given data set and also to find the number of clusters existing. The voting procedure of the algorithm combines the results of several independent cluster runs by voting between their results. 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 and is not influenced by the randomness of initialization or the cluster process itself. Moreover, we develop a strategy of analyzing and interpreting the results of the voting algorithm in such a way that existing clusters as well as data points belonging to a cluster can be identified as "sure" or "clear" concerning their structure and their partition respectively. The next part of the algorithm consists of a merging process of the clusters resulting by voting, while a criterion is computed in order to make the decision in favor of a specific number of clusters. Concluding, the VMA does not suffer from thresholds definitions, since only one parameter is needed to be defined is the initial number of clusters, it diminishes to the minimum possible the instabilities of other clustering methods and can support non convex solutions.

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').

References

Ball, G. H. and Hall, D. J. (1965). Isodata, a novel method of data analysis and pattern classification. Technical report, Stanford Research Institute, Menlo Park.

Begovich, C. L. and Kane, V. E. (1982). Estimating the number of groups and group membership using simulation cluster analysis. *Pattern Recognition*, 15(4):335–342.

Biernacki, C., Celeux, G., and Govaert, G. (1997). Assessing a mixture model for clustering with the integrated classification likelihood. Rapport de recherche 3521, Theme 4, Unite de recherche INRIA Lorraine, http://www.inria.fr.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24:123–140.

Brown, D. E., Huntley, C. L., and Garvey, P. J. (1991). Clustering of homogeneous subsets. *Pattern Recognition Letters*, 12:401–408.

Carpenter, G. and Grossberg, S. (1986). Adaptive resonance theory: Stable selforganization of neural recognition codes in response to arbitrary lists of input patterns. In *Proc. 8th Annu. Conf. Cognitive Sci. Soc.*, pages 45–62.

Chaudhuri, D., Chaudhuri, B., and Murthy, C. (1992). A new split-and-merge clustering technique. *Pattern Recognition Letters*, 13:399–409.

DeFigueiredo, R. J. P. (1996). The oi, os, omni and osman networks as best approximations of nonlinear systems under training data constraints. In *Proc. IEEE Int. Symp. Circuits Syst.*, Seattle, WA.

Dimitriadou, E., Weingessel, A., and Hornik, K. (1999). Voting in clustering and finding the number of clusters. In *Proceedings of the "International ICSC Symposium on Advances in Intelligent Data Analysis (AIDA 99)" ("International Congress on Computational Intelligence: Methods and Applications (CIMA 99)"*. ICSC Academic Press. To appear.

Dolnicar, S., Leisch, F., Weingessel, A., Buchta, C., and Dimitriadou, E. (1998). A comparison of several cluster algorithms on artificial binary data scenarios from tourism marketing. Working Paper 7, SFB "Adaptive Information Systems and Modeling in Economics and Management Science", http://www.wu-wien.ac.at/am.

Eltoft, T. and DeFigueiredo, R. J. P. (1998). A new neural network for cluster-detection-and-labeling. *IEEE Transactions on Neural Networks*, 9(5):1021–1034.

Freund, Y. and Schapire, R. E. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. *Lecture Notes in Computer Science*, 904.

Linde, Y., Buzo, A., and Gray, R. M. (1980). An algorithm for vector quantizer design. *IEEE Transactions on Communications*, COM-28(1):84–95.

Mao, J. and Jain, A. K. (1995). Artificial neural networks for feature extraction and multivariate data projection. *IEEE Transactions on Neural Networks*, 6(2):296–317.

Marazzi, A., Gamba, P., Mecocci, A., and Semboloni, A. (1996). Automatic selection of the number of clusters in multidimensional data problems. In *Proceedings of the International Conference on Image Processing*, volume 3, pages 631–634, NY, USA. IEEE.

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

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

Milligan, G. W. (1980). An examination of the effect of six types of error perturbation on fifteen clutering algorithms. *Psychometrika*, 45(3):325–342.

Milligan, G. W. (1981). A monte carlo study of thirty internal criterion measures for cluster analysis. *Psychometrika*, 46(2):187–199.

Milligan, G. W. and Cooper, M. C. (1985). An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179.

Mucha, H.-J. (1992). *Clusteranalyse mit Mikrocomputern*. Akademie Verlag.

Wallace, R. S. and Kanade, T. (1990). Finding natural clusters having minimum description length. In *Proceedings of the 10th International Conference on Pattern Recognition*, volume 1, pages 438–442, Los Alamitos, CA, USA. IEEE Comput. S. Press.

Weigend, A. S., Mangeas, M., and Srivastava, A. N. (1995). Nonlinear gated experts for time series: Discovering regimes and avoiding overfitting. *International Journal of Neural Systems*, 6:373–399.

Weingessel, A., Dimitriadou, E., and Dolnicar, S. (1999a). An examination of indexes for determining the number of clusters in binary data sets. Working Paper 29, SFB "Adaptive Information Systems and Modeling in Economics and Management Science", http://www.wu-wien.ac.at/am.

Weingessel, A., Dimitriadou, E., and Hornik, K. (1999b). A voting scheme for cluster algorithms. In Krell, G., Michaelis, B., Nauck, D., and Kruse, R., editors, *Neural Networks in Applications, Proceedings of the Fourth International Workshop NN'99*, pages 31–37, Otto-von-Guericke University of Magdeburg, Germany.

Xu, L. (1997). Bayesian ying-yang machine, clusterong and number of clusters. *Pattern Recognition Letters*, 18:1167–1178.

Xu, L., Krzyzak, A., and Oja, E. (1993). Rival penalized competitive learning for clustering analysis RBF net and curve detection. *IEEE Transactions on Neural Networks*, 4(4):636–649.

Yin, P.-Y. and Chen, L.-H. (1994). A new non-iterative approach for clustering. *Pattern Recognition Letters*, 15:125–133.