Report Series

Voting in Clustering and Finding the Number of Clusters

Evgenia Dimitriadou
Andreas Weingessel
Kurt Hornik

Report No. 30
March 1999

Report Series

March 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 "International ICSC Symposium on Advances in Intelligent Data Analysis (AIDA 99)" ("International Congress on Computational Intelligence: Methods and Applications (CIMA 99)", ICSC Academic Press)

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').

Voting in Clustering and Finding the Number of Clusters

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 present an unsupervised algorithm which performs clustering given a data set and which can also find the number of clusters existing in it. This algorithm consists of two techniques. The first, the voting technique, allows us to combine several runs of clustering algorithms, with the number of clusters predefined, resulting in a common partition. We introduce 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". The second part consists of an index measure which receives the results of every voting process for different number of clusters and makes the decision in favor of one. This algorithm is a complete clustering scheme which can be applied to any clustering method and to any type of data set. Moreover, it helps us to overcome instabilities of the clustering algorithms and to improve the ability of a clustering algorithm to find structures in a data set.

Keywords: Clustering Algorithms, Unsupervised Learning, Stability, Number of Clusters

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, different cluster algorithms and even multiple replications of the same algorithm result in different solutions due to random initializations and stochastic learning methods. Every clustering 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. Second, there is very little indication about the correct choice of the number of clusters to look for.

To tackle the first problem, we handle different results of various runs by using the existing idea of voting in classification (Breiman, 1996; Freund & Schapire, 1995), where the output of several single classifiers can be combined to reduce the variance of the error and to get an overall decision made by the combined classifiers. We develop a process which allows voting between several results of a clustering algorithm. The idea is that the voting process can be applied to any existing algorithm that has instable results.

In the next step of the algorithm, we take advantage of the "fuzzy" partition that voting results to, in order to find a measure to specify the "right" number of clusters existing in the data set. We introduce some measures that help to explain and handle the voting results and also an index measure, which we compute for different number of clusters. This shows us the "importance" of every cluster solution in order to make the decision in favor of one.

2 Proposed Algorithm

We describe analytically the proposed algorithm as a two stage algorithm starting with the voting process and continuing with the work out of the voting

results together with the index measure for finding the number of clusters.

2.1 Voting

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 clustering 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.

We developed the following algorithm. 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 Finding the Number of Clusters

For the results of the voting algorithm 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 has been 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 has been assigned to its "winning" cluster k, that is $sureness(x) = \max_j(D_{Nj})$. 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. In this way we can notice which clusters have a clear data structure and which not.

We can now compute the "sureness of the result", as the mean value of the sureness of all data points. To find the right number of clusters we apply the voting algorithm to an increasing number n of clusters and we compute for each n the sureness of this result, which we call *numsure(n)*.

On one hand, there is the tendency that *numsure(n)* decreases starting from the minimum number of clusters to the maximum, because the decision has to be made between more and more clusters. On the other hand, we expect an increase of *numsure(n)*, if we reach the right number of clusters. To take both effects into account, we propose the following measure $devsure(n)$ to find the right number of clusters:

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

This second order difference shows how much the solution for n clusters deviates from the general decrease of the sureness. The solution with the maximum value for $devsure(n)$ is the one whose cluster structure found by voting is the clearest.

3 Experimental Results

3.1 Implementation

We apply our voting algorithm on the results of several off- and on-line clustering algorithms (Linde et al., 1980; Xu et al., 1993; Martinetz et al., 1993). We use artificial and real world data sets, where the true cluster structure is known.

All of the above algorithms are not able to indicate or propose the appropriate solution in the case of unstable 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 is to study how far our algorithm can overcome these instabilities.

The experiments presented here consist of 100 runs of a clustering algorithm and a voting between these runs (see also Weingessel et al., 1998). This procedure is repeated for different prespecified number of clusters. Cluster solutions are computed starting with 2 cluster centers up to 13 centers. The range is chosen so that it contains at least twice the number of clusters that there is in the data sets, so that the solution where every existing cluster might be split into two parts is still contained in the range of considered centers. After every voting the proposed measure is calculated for every number of prespecified clusters, in order to find in the end the number of clusters in the 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.

In Tables 1, 2, 4 and 5 we show the *avesure* of every cluster in the data set for a typical run of voting, and also the mean value and standard deviation of the mean average sureness of all clusters in 100 voting runs. The results given in Table 6 are the mean value and standard deviation 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 100 voting ones. Each voting run has been performed with the results of 100 new cluster runs.

Moreover, we show in Table 7 how many times our algorithm finds the right number of clusters in a set. Also, to give an idea of how clear the decision of the measure is, the absolute difference between the winning number of clusters and the second number is computed and scaled by the range of the numbers for all clusters. The mean value and standard deviation of this in 100 runs (both multiplied by 100) is given in the columns *Sureness of Decision* and *Sd*. Consequently, if the measure finds the right number of clusters (almost) every time, it is optimal for the *Sureness of Decision* to be high, if the measure is wrong, it is favorable to be small to indicate an unsure decision. In Figure 1 we see the mean average sureness of a typical result of a voting run for every number of clusters, where as in Figure 2 we see the respective values of the decision measure for every data.

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

3.2 Data Sets

Artificial data sets as well as real-world sets are used to demonstrate the performance of our algorithm. 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.

3.2.1 4 Gaussian Balls

This data set, which is an extension of the one of Mao & Jain (1995) consists of 4 normally distributed clusters in a 10-dimensional space with mean values $\mu_1 = (1,1,1,1,1,1,1,1,1,1)^T$, $\mu_2 = -\mu_1$, $\mu_3 = (1,1,1,1,1,-1,-1,-1,-1,-1)^T$, and $\mu_4 = -\mu_3$ and the identity matrix as covariance matrix $\Sigma_1 = \Sigma_2 = \Sigma_3 = \Sigma_4 = I$. Each cluster consists of 500 points.

In this data set every voting run is applied to 100 k-means results. Since this data set is well separated, the *numsure* of the voting result with 4 clusters is 100% in 98 out of 100 repetitions (see Table 1). In the 2 other repetitions the *numsure* is 99.6%. Thus, it is not difficult for our algorithm to find always that this data set consists of 4 clusters (see Table 7, also Figure 2). We also have to mention that since in this data set the k-means clustering algorithm yields always stable results (see Table 6), it is consequent that the voting algorithm yields an identical result since there are no different results to "vote" in between.

Cluster	1	2	3	4	mean	sd
Avesure	100	100	100	100	99.99	0.06

Table 1: Gaussian 4 Clusters: Average Sureness

3.2.2 3 Gaussian Balls

This data set consists of 3 Gaussian Balls. The first two of them correspond to the first two of the previous example, the third cluster has only 100 points and lies in between with zero mean zero and covariance matrix $\Sigma_3 = 0.01I$. The difficulty of this data set is the small cluster in the middle. The k-mean clustering algorithm, with the number of 3 clusters specified, suffers in this data set by not being many

times able to find the correct structure but it splits the external big ball into 2 clusters and shares the data points of the middle one to all the 3 clusters. Nevertheless, our algorithm finds the correct 3 clusters in 95 out of the 100 runs; only twice the result is 2 clusters (thus ignoring the small cluster), three times the algorithm suggests 10 clusters (see Table 7, also Figure 2). Moreover, the classification rate improves from 87% to 93% and the high standard deviation (almost 11) is decreased reaching almost the zero value (see Table 6).

Cluster	1	2	3	mean	sd
Avesure	96.69	94.34	79.48	91.23	1.47

Table 2: Gaussian 3 Clusters: Average Sureness

3.2.3 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 (see Table 3) that has been designed in the following way: we have 12 binary variables belonging to 4 groups (G1, G2, G3, G4) 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 probability (H stands for 0.8) to have a 1 there and for the other variables a low probability (L for 0.2). In the experiment the sizes of the 6 types (and therefore the sizes of the expected clusters) are 1000 each. The clusters in this example 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 82.98% for this data set. For a more detailed description of this (and other) artificial binary data sets, see Dolnicar et al. (1998).

Type	G1	G2	G3	G4	n
1	H	H	L	L	1000
2	L	L	H	H	1000
3	L	H	H	L	1000
4	H	L	L	H	1000
5	L	H	L	H	1000
6	H	L	H	L	1000

Table 3: Binary Data Set

Table 6 shows that voting can improve the classification rate from 81% to 83% which is close to the Bayes rate. Also the standard deviation between different voting runs is close to 0. As seen in Table 7 the voting algorithm finds all the time in all 100 runs the right number of clusters. The single runs of a clustering algorithm for this data set are performed by hard competitive learning method.

Cluster	1	2	3
Avesure	90.78	90.93	88.48
Cluster	4	5	6
Avesure	90.34	92.59	88.72
	mean	sd	
Avesure	88.72	1.00	

Table 4: Binary Data: Average Sureness

3.2.4 Iris Data Set

We apply a clustering algorithm to the well known Iris data set (see for example Mucha, 1992, page 131). The reason of using this data set is to prove the competence of the voting algorithm to a real and known data set. Basically, the Iris data set is considered as a classification problem because the true classes are known. But because there are no well known benchmark data sets for cluster problems, we treat the problem as a clustering one (Mucha, 1992, page 202).We apply the hard competitive learning algorithm because it proved to have less stable results than k-means and then we compare it with voting. In Table 6 we can see that voting improves the performance from 82% to 89% with almost 0 standard deviation (13 by the hard competitive learning algorithm). Table 5 shows that the "setosa cluster", which is linearly separable from the others, is the surest.

In 86 out of 100 runs our algorithm find the right number of 3 clusters, the other decisions are 4 clusters once, 6 clusters 4 times, and 8 to 12 clusters the remaining 9 repetitions (see Table 7, also Figure 2).

Cluster	setosa	versicolor	virginica
Avesure	100.00	93.87	95.58
	mean	sd	
Avesure	96.45	0.39	

Table 5: Iris: Average Sureness

	clust.alg.		voting	
	mean	std	mean	std
Gauss 4 Cl.	97.75	0.00	97.75	0.00
Gauss 3 Cl.	86.74	10.97	92.74	0.64
Binary	81.43	2.88	82.78	0.15
Iris	82.73	13.03	89.00	0.38

Table 6: Data Sets: Correct Classification

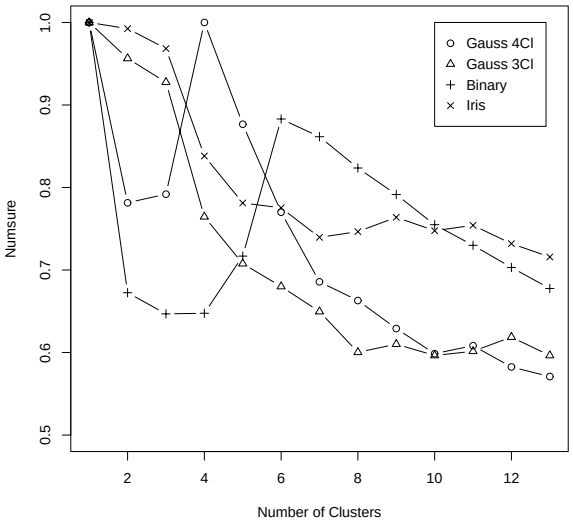

Figure 1: Mean Average Sureness

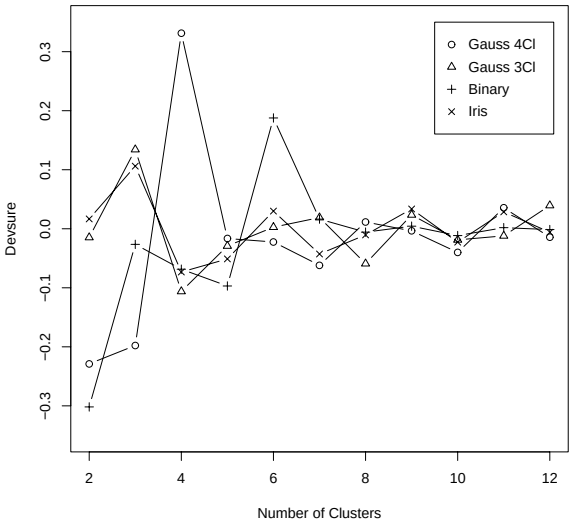

Figure 2: Measure values

Data Set	Right Number Found	Sureness of Decision	Sd
Gauss 4Cl.	100	53.69	3.19
Gauss 3Cl.	95	37.75	11.11
Binary	100	45.22	6.81
Iris	86	19.39	0.11

Table 7: Data Sets: Right Number of Clusters Found

4 Conclusions

In this paper we present a method to combine the results of several independent clustering runs by voting between their results. This allows us to deal with the problem of local minima of clustering algorithms and to find a partition of the data which is supported by repeated applications of the clustering algorithm, without being influenced by the randomness of initialization or the cluster process itself. Furthermore, we introduce some measures that help us to understand and to analyze the results of the voting process and moreover to take advantage of these results by defining an index measure that can be used to find the right number of clusters in a data set. Therefore, this algorithm helps to overcome the problems of instable cluster results and of an unknown number of clusters.

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

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

Dolnicar, S., Leisch, F., Weingessel, A., Buchta, C., & 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.

Freund, Y. & 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., & Gray, R. M. (1980). An algorithm for vector quantizer design. *IEEE Transactions on Communications*, **COM-28**(1), 84–95.

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

Martinetz, T. M., Berkovich, S. G., & 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.

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

Weingessel, A., Dimitriadou, E., & Hornik, K. (1998). *A Voting Scheme for Cluster Algorithms*. Working Paper 23, SFB "Adaptive Information Systems and Modeling in Economics and Management Science", http://www.wu-wien.ac.at/am. Accepted for publication in: Proceedings of the "Fourth International Workshop Neural Networks in Applications '99", Magdeburg, Germany.

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