A maximum variance cluster algorithm

Article *in* IEEE Transactions on Pattern Analysis and Machine Intelligence · October 2002

DOI: 10.1109/TPAMI.2002.1033218 · Source: IEEE Xplore

3 authors, including:

Cor J. Veenman
Leiden University
69 PUBLICATIONS **3,040** CITATIONS
SEE PROFILE

Marcel J T Reinders
Delft University of Technology
634 PUBLICATIONS **13,917** CITATIONS
SEE PROFILE

A Maximum Variance Cluster Algorithm

C.J. Veenman, M.J.T. Reinders, and E. Backer *

Abstract We present a partitional cluster algorithm that minimizes the sum-of-squared-error criterion while imposing a hard constraint on the cluster variance. Conceptually, hypothesized clusters act in parallel and cooperate with their neighboring clusters in order to minimize the criterion and to satisfy the variance constraint. In order to enable the demarcation of the cluster neighborhood without crucial parameters, we introduce the notion of *foreign* cluster samples. Finally, we demonstrate a new method for cluster tendency assessment based on varying the variance constraint parameter.

Keywords: cluster analysis, partitional clustering, cluster tendency assessment, cluster validity.

1 Introduction

Data clustering is an extensively investigated problem for which many algorithms have been reported [18], [26]. Roughly, cluster algorithms can be categorized in hierarchical and partitional algorithms. Hierarchical algorithms deliver a hierarchy of possible clusterings, while partitional cluster algorithms divide the data up into a number of subsets. In partitional cluster analysis most algorithms assume the number of clusters to be known a priori. Because in many cases the number of clusters is not known in advance, additional validation studies are used to find the optimal partitioning of the data [6], [8], [11], [16].

In this paper, we propose an algorithm for partitional clustering that minimizes the within cluster scatter with a constraint on the cluster variance. Accordingly, in contrast to many other cluster algorithms, this method finds the number of clusters automatically. Clearly, a proper value for the variance constraint parameter has to be selected. We present a way to discover cluster tendencies to find significant values for this variance parameter in case this information is not available from the problem domain. We first formally define the cluster problem.

*The authors are with the Department of Mediamatics, Faculty of Information Technology and Systems, Delft University of Technology, P.O.Box 5031, 2600 GA, Delft, The Netherlands. E-mail: {C.J.Veenman, M.J.T.Reinders, E.Backer}@its.tudelft.nl

Let $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$ be a data set of $N = |X|$ feature vectors in a p-dimensional metric space. Then, the cluster problem is to find a clustering of X in a set of clusters $C = \{C_1, C_2, ..., C_M\}$, where M is the number of clusters, such that the clusters C_i are homogeneous and the union of clusters is inhomogeneous.

The most widely used criterion to quantify cluster homogeneity is the sum-of-squared-error criterion or simply the square-error criterion[1]:

$$J_e = \frac{\sum_{i=1}^{M} H(C_i)}{N}, \tag{1}$$

where

$$H(Y) = \sum_{\boldsymbol{x} \in Y} \|\boldsymbol{x} - \mu(Y)\|^2 \tag{2}$$

expresses the cluster homogeneity and

$$\mu(Y) = \frac{1}{|Y|} \sum_{\boldsymbol{x} \in Y} \boldsymbol{x} \tag{3}$$

is the cluster mean.

Straight minimization of (1) leads to a trivial clustering with N clusters; one for each sample. Therefore, additional constraints are imperative. For instance, one could fix M, the number of clusters, to an a priori known number like among others the widely used K-means model [23]. In the image segmentation domain, a maximum variance per cluster is sometimes used in addition to a spatial connectivity constraint, e.g. [1], [15]. In this paper, we present an algorithm that is based on a model proposed for intensity-based image segmentation [28]. The constraint that is imposed on the square-error criterion (1) within this model states that the variance of the union of two clusters must be higher than a given limit σ^2_{max}, i.e.:

$$\forall C_i, C_j, i \neq j : Var(C_i \cup C_j) \geq \sigma^2_{max}, \quad \text{where } Var(Y) = \frac{H(Y)}{|Y|} \tag{4}$$

A consequence of this model is that the variance of each resulting cluster is generally *below* σ^2_{max} [28]. This does, however, not imply that we could impose a maximum variance constraint on each individual cluster instead. That is, if we would replace the joint variance constraint (4) with a constraint for individual clusters ($\forall C_i : Var(C_i) \leq \sigma^2_{max}$) the minimization of (1) would lead to a trivial solution with one sample per cluster.

Clearly, since the model imposes a variance constraint instead of fixing the number of clusters, the resulting optimal clustering can be different from the K-means result, even if the final number of clusters is the same.

[1] Usually the sum-of-squared-error criterion is not averaged over the whole data set. As defined here, J_e expresses the average distance to the cluster centroids instead of the total distance.

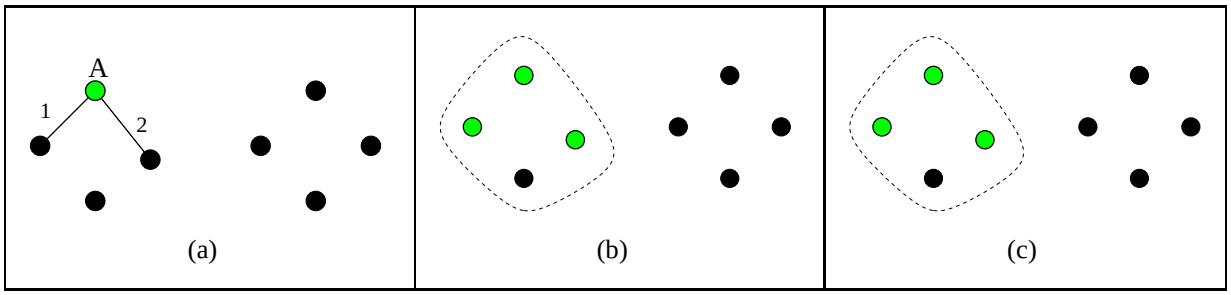


Figure 1: Illustration of the cluster neighborhood with the 2 nearest-neighbors method. In (a) the neighbor ranking for sample *A* is shown. (b) and (c) display the neighborhood of the grey colored cluster with respectively 3 and 4 samples, where in (c) the set of expansion candidates is empty.

2 Algorithm

For the optimization of the cluster model, we propose a stochastic optimization algorithm. In the literature other stochastic clustering algorithms have been reported that generally optimize the K-means model or fuzzy C-means model either using simulated annealing techniques [7], [20], [25] or using evolutionary computation techniques [10], [13], [22], [29]. Accordingly, these stochastic approaches focus on the optimization of known cluster models. The algorithm we propose, however, shows more resemblance with the distributed genetic algorithm (DGA) for image segmentation as introduced by Andrey and Tarroux [2], [3]. We also dynamically apply local operators to gradually improve a set of hypothesized clusters, but, in contrast with the DGA approach, we consider the statistics of the whole cluster in the optimization process. Before describing the algorithm itself, we first elaborate on the neighborhood relationships of samples, which play a crucial role in the proposed algorithm.

Both for effectiveness and efficiency the algorithm exploits *locality* in the feature space. Namely, the most promising candidates for cluster expansion are in clusters that are close in the feature space. Similarly, the most distant cluster members are the least reliable, hence, they are the first candidates for removal. The computational performance can also profit from feature locality because cluster update operations can be executed in *parallel* when the optimization process applies locally. For these reasons, we consider the optimization process from the individual clusters' point of view, i.e. each cluster can execute a number of actions in order to contribute to an improvement of the criterion as well as to satisfy the variance constraint.

In order to collect the expansion candidates of a cluster, we need to find neighboring samples of that cluster. A common way to define the neighborhood of a sample is to collect its k nearest neighbors using the Eucledian distance measure, where k is a predefined number. In this way, the neighborhood of a cluster would be the union of the neighbors of all samples in the cluster. Accordingly, the set of expansion candidates of a cluster consists of the samples from its neighborhood, excluding the

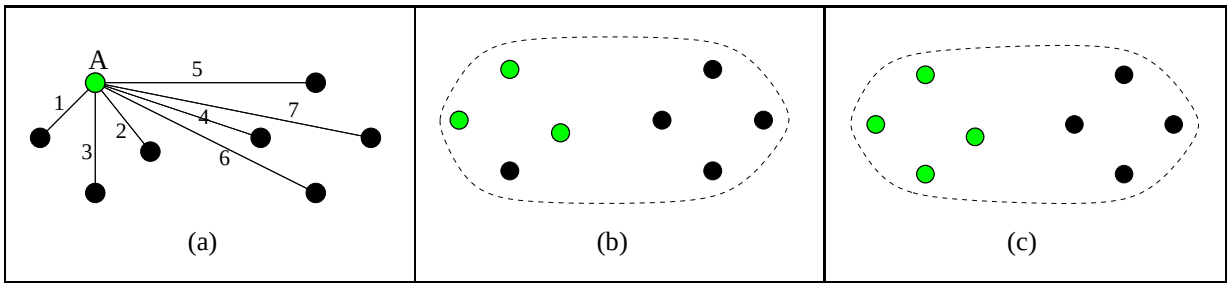


Figure 2: Illustration of the cluster neighborhood with the 8 nearest-neighbors method. In (a) the neighbor ranking for sample A is shown. (b) and (c) display the neighborhood of the grey colored cluster with respectively 3 and 4 samples.

samples from the cluster itself. The problem with this approach is that the value of k becomes an integral part of the cluster model, e.g. [12], [19], [24]. If k is set too low, then even for small clusters all k nearest neighbors are in the cluster itself, so there are no expansion candidates left, see Fig. 1. On the other hand, if k is set too high, then the neighborhood is always large, so all clusters have a major part of the samples as expansion candidates, which clearly violates the idea of locality. As a consequence, the set of expansion candidates will be a mix of good and bad candidate samples without preference, see Fig. 2.

We take another approach to collect the expansion candidates. First, we call the set of expansion candidates of cluster C_a the *outer border* B_a of cluster C_a. Further, we introduce the notion of *foreign* samples, which we define as neighboring samples that are not in the cluster itself. Accordingly, the k-th order outer border B_a of cluster C_a is the union of the k *nearest foreigners* of all samples in C_a, leading to:

$$B_a(k) = \bigcup_{\boldsymbol{x} \in C_a} F(\boldsymbol{x}, k, C_a, X), \tag{5}$$

where $F(\boldsymbol{x}, k, C_a, X)$ is the set of k nearest foreigners of $\boldsymbol{x}$ according to:

$$F(\boldsymbol{x}, k, C_a, Y) = \begin{cases} \{nf(\boldsymbol{x}, C_a, Y)\} \cup F(\boldsymbol{x}, k-1, C_a, Y - \{nf(\boldsymbol{x}, C_a, Y)\}), & \text{if } k > 0 \\ \emptyset, & \text{if } k = 0 \end{cases} \tag{6}$$

and $nf(\boldsymbol{x}, C_a, Y)$ is the nearest foreigner of sample $\boldsymbol{x} \in C_a$ in X defined as:

$$nf(\boldsymbol{x}, C_a, Y) = \arg \min_{\boldsymbol{y} \in Y - C_a} \|\boldsymbol{y} - \boldsymbol{x}\|^2 \tag{7}$$

Consequently, the outer border of a cluster always has a limited number of samples and it never becomes empty (unless there is only one cluster left). In Fig. 3 we illustrate how a second-order outer border evolves with the growing of a cluster. An appropriate value for the order of the outer border depends on the constellation of the clusters and the actual data.

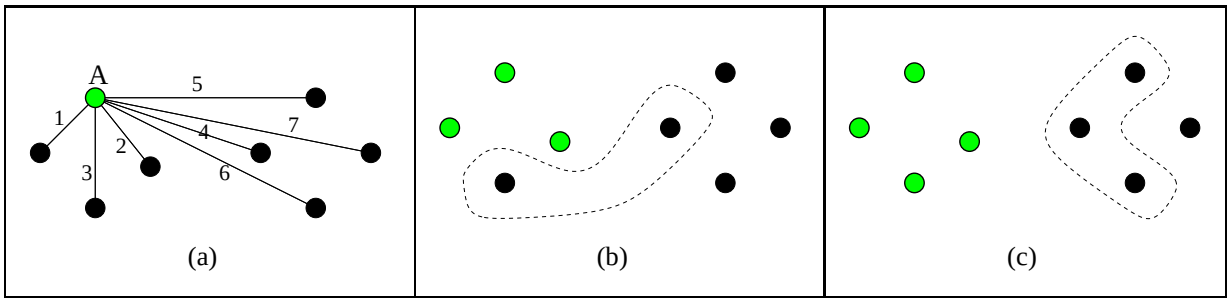


Figure 3: The figures illustrate the cluster border construction with respect to sample A and its cluster. In (a) the distance ranking for sample A is shown and (b) and (c) display the second-order border of the grey colored cluster with respectively 3 and 4 samples.

Besides the expansion candidates, we also need to collect candidates for removal from the cluster in order to impose the variance constraint. To this end, we introduce the q-th order inner border I_a of cluster C_a. The inner border I_a consists of those samples that are the furthest cluster mates of the samples in C_a. Accordingly, the q-th order inner border can be expressed as follows:

$$I_a(q) = \bigcup_{\boldsymbol{x} \in C_a} G(\boldsymbol{x}, q, C_a), \tag{8}$$

where $G(\boldsymbol{x}, q, C_a)$ is the set of q furthest cluster mates of $\boldsymbol{x}$, or, in other words the q furthest neighbors of $\boldsymbol{x}$ in C_a according to:

$$G(\boldsymbol{x}, q, Y) = \begin{cases} \{fn(\boldsymbol{x}, Y)\} \cup G(\boldsymbol{x}, q-1, Y - \{fn(\boldsymbol{x}, Y)\}), & \text{if } q > 0 \\ \emptyset, & \text{if } q = 0 \end{cases} \tag{9}$$

and $fn(\boldsymbol{x}, Y)$ is the furthest neighbor of $\boldsymbol{x}$ in Y as in:

$$fn(\boldsymbol{x}, Y) = \arg\max_{\boldsymbol{y} \in Y} \|\boldsymbol{y} - \boldsymbol{x}\|^2 \tag{10}$$

Since the set of foreigners of a sample changes every time the cluster is updated, for efficiency reasons we introduce a rank list R_i per sample $\boldsymbol{x}_i$, containing indices to all other samples in X in order of their distance to the given sample. The rank list R_i is an N-tuple defined as $R_i = Rank(\boldsymbol{x}_i, X)$ according to:

$$Rank(\boldsymbol{x}, Y) = \langle nn(\boldsymbol{x}, Y) \circ Rank(\boldsymbol{x}, Y - \{nn(\boldsymbol{x}, Y)\}) \rangle, \tag{11}$$

where $\circ$ is the concatenate operator and $nn(\boldsymbol{x}, Y)$ is the nearest neighbor of $\boldsymbol{x}$ in Y as in:

$$nn(\boldsymbol{x}, Y) = \arg\min_{\boldsymbol{y} \in Y} \|\boldsymbol{y} - \boldsymbol{x}\|^2 \tag{12}$$

The k nearest foreigners of a cluster sample can now easily be found by scanning the rank list, starting at the head while skipping those elements that are already in the cluster. To this end, some bookkeeping is needed for both the clusters and the samples.

After the definitions of the inner and outer border of a cluster, we now describe the maximum variance cluster (MVC) algorithm. Since the optimization of the model defined in (1) – (4) is certainly an intractable problem, exhaustive search of all alternatives is out of the question. In order to prevent early convergence of the consequent approximate optimization process, we introduce sources of non-determinism in the algorithm [4].

The algorithm starts with as many clusters as samples. Then in a sequence of epochs every cluster has the possibility to update its content. Conceptually, in each epoch the clusters act in parallel or alternatively sequentially in random order. During the update process, cluster C_a performs a series of tests each of which causes a different update action for that cluster.

1. *Isolation*

   First C_a checks whether its variance exceeds the predefined maximum σ^2_{max}. If so, it randomly takes a number of candidates i_a from its inner border I_a proportional to the total number of samples in I_a. It *isolates* the candidate that is the furthest from the cluster mean $\mu(C_a)$. It takes a restricted number of candidates to control the greed of this operation. Then the isolated sample forms a new cluster (resulting in an increase of the number of clusters).

2. *Union*

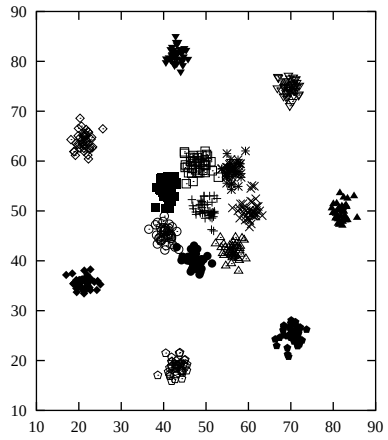
   If on the other hand, C_a is homogeneous (its variance is below σ^2_{max}), then it checks if it can *unite* with a *neighboring cluster*, where a neighboring cluster is a cluster that contains a foreign sample of C_a. To this end, it computes the joint variance with its neighbors. If the lowest joint variance remains under σ^2_{max}, then the corresponding neighbor merges with C_a (resulting in a decrease of the number of clusters).

3. *Perturbation*

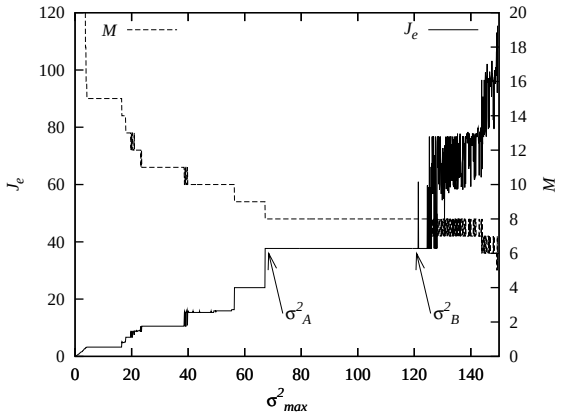
   Finally, if none of the other actions applies, the cluster C_a attempts to *improve the criterion* by randomly collecting a number of candidates b_a from its outer border B_a. Again, to control the greed, a restricted number of candidates is selected proportional to the size of the border. Then C_a ranks these candidates with respect to the gain in the square-error criterion when moving them from the neighboring cluster C_b to C_a.

   We define the criterion gain between C_a and C_b with respect to $\boldsymbol{x} \in C_b$ as:

$$G_{ab} = H(C_a) + H(C_b) - H(C_a \cup \{\boldsymbol{x}\}) - H(C_b - \{\boldsymbol{x}\}) \tag{13}$$



(a)



(b)

Figure 4: In (a) the R15 data set is shown, which is generated as 15 similar 2-D Gaussian distributions. In (b) J_e and M are displayed as a function of σ^2_{max} for the R15 data set.

If the best candidate has a positive gain then this candidate moves from the neighbor to C_a. Otherwise, there is a small probability P_d of *occasional defect*, which forces the best candidate to move to C_a irrespective the criterion contribution.

Because of the occasional defect, no true convergence of the algorithm exists. Therefore, after a certain number of epochs E_{max}, we set $P_d = 0$. Further, since it is possible that at the minimum of the constrained optimization problem (1) – (4) the variance of some clusters exceeds σ^2_{max} (exceptions to the general rule mentioned in Section 1), after E_{max} epochs also isolation is no longer allowed in order to prevent algorithm oscillations. With these precautions the algorithm will certainly converge, since the overall homogeneity criterion only decreases and it is always greater than or equal to zero. In case two clusters unite, the criterion may increase, but the number of clusters is finite. Still, we have to wait for a number of epochs in which the clusters have not changed due to the stochastic sampling of the border.

3 Experiments

In this section we demonstrate the effectiveness of the proposed maximum variance cluster (MVC) algorithm with some artificial and real data sets. First, however, we show that the maximum variance constraint parameter can be used for cluster tendency assessment.

Clearly, since the clustering result depends on the setting of σ^2_{max}, the square-error criterion J_e also changes as a function of σ^2_{max}. Accordingly, cluster tendencies can be read from trends in J_e. Consider for instance the data set shown Fig. 4(a). The corresponding square-error curve resulting from varying

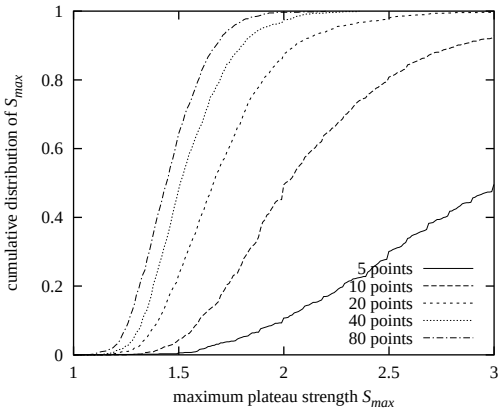


Figure 5: Cumulative distribution of the maximum plateau strength for differently sized random 2-D data sets. The distributions have been calculated from 1000 independent draws.

σ^2_{max} can be seen in Fig. 4(b). The figure shows some prominent plateaus in the square-error criterion. Clearly, these plateaus can both be caused by hierarchical cluster structures and random sample patterns. If there is real structure in the data then the variance constraint can be increased up to the moment that true clusters are lumped together. On the other hand, if the resulting clustering is random in character, then the clusters will easily be rearranged when the variance constraint is increased.

Let σ^2_A be the starting point of a J_e plateau and σ^2_B the end point of that plateau, as for example in Fig. 4(b). Then, we define the strength S of a plateau as the ratio:

$$S(\sigma^2_A, \sigma^2_B) = \frac{\sigma^2_B}{\sigma^2_A} \tag{14}$$

Accordingly, the strength of a plateau gives an indication of whether or not the corresponding clustering represents real structure in the data. Intuitively, we expect that two real clusters will be lumped together if the variance constraint is higher than roughly twice their individual variance. This implies that the strength of a plateau in J_e should be greater than 2 in order to be a *significant plateau*, that is, to represent real structure. To test this hypothesis, we did experiments with uniform random data and various numbers of samples. For each data set, we measured the maximum plateau strength S_{max} and subsequently computed the distribution of S_{max}. In Fig. 5 we show the cumulative distribution of S_{max} for different sizes of the random data sets. Only when N was very low ($N < 50$), significant plateaus were occasionally found, which is to be expected with low numbers of samples. On the other hand, the experiments with structured data, among which the ones that we describe in this section, indeed resulted in significant J_e plateaus.

The advantage of this cluster tendency assessment approach is that the same model is used for the clustering as for the cluster tendency detection. In our view the usual approach, where different criteria are used for the clustering and the detection of the cluster tendency, is undesirable, like for

instance in [6], [8], [11], [16]. That is, the cluster algorithm may not be able to find the clustering corresponding to the local minimum or knee in the cluster tendency function[2].

Some additional remarks need to be made about the significance of J_e plateaus. First, it has to be noted that because of fractal like effects, σ_A^2 must be higher than a certain value in order to rule out extremely small 'significant' plateaus. Second, in case there are *multiple* significant plateaus, these plateaus represent the scales at which the data can be considered. Then, the user can select the appropriate scale and corresponding plateau. In our view, there is no best scale in these cases, so the selection is fully subjective.

In all experiments, we compared the performance of the MVC algorithm to the K-means algorithm [23], and the Gaussian mixtures modeling (GMM) method with likelihood maximisation [18] using the EM algorithm [9]. For both the K-means and the GMM method the numbers of clusters is set to the resulting number (M) found by the MVC algorithm. Further, since both the MVC and the K-means algorithm prefer circular shaped clusters we constrained the Gaussian models of the GMM to be circular too in order to reduce its number of parameters. For the MVC algorithm we set $P_d = 0.001$, $E_{max} = 100$, $k = 3$, $q = 1$, $i_a = \lfloor\sqrt{|I_a|}\rfloor$, and $b_a = \lfloor\sqrt{|B_a|}\rfloor$. It has to be noted that these parameter values appeared to be not critical (experiments not included). They merely serve to tune the convergence behavior similar as in other non-deterministic optimization algorithms, like for instance the mutation rate and population size parameters in genetic algorithms [14]. Since all algorithms have non-deterministic components[3], we ran them 100 times on each data set and display the result with the lowest square-error (MVC and K-means) or highest likelihood (GMM), i.e. the best solution found. Further, we measured the average computation time, and the number of times the best solution was found (hit rate). For the MVC algorithm we did not add the computation time of the rank lists, since this time only depends on the size of the data set and not on the structure. Moreover, these lists have to be computed only once for a data set and can then be used for tendency assessment and the subsequent runs to find the optimal clustering.

We start with the already mentioned data set from Fig. 4(a) consisting of 15 similar 2-D Gaussian clusters that are positioned in rings (R15). Though we know an estimate of the variance of the clusters, we first varied the σ_{max}^2 constraint in order to discover the cluster tendency. Fig. 4(b) shows the resulting curves for J_e and the number of found clusters M. The figure shows a number of prominent plateaus in J_e, from which the first [4.20...16.5] has strength $S = 3.93$. This significant plateau corresponds to the originating structure of 15 clusters. Further, there is a large plateau [67.5...122] with strength $S = 1.80$ which corresponds to the clustering where all inner clusters are merged into one cluster. This plateau is, however, not significant according to our definition. This is because

[2]When additional criteria are used to discover cluster tendencies, they are usually called cluster validity functions or indices.

[3]The K-means and GMM algorithm are initialized with randomly chosen cluster models.

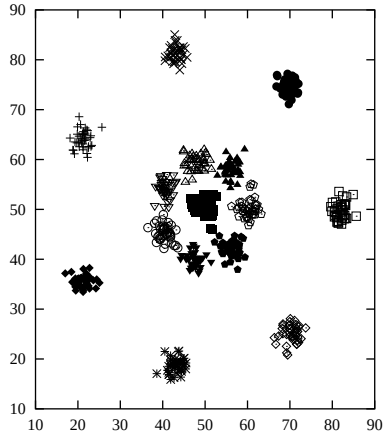


Figure 6: Results of applying the clustering algorithms to the R15 data set. In (a) the results of the MVC and K-means algorithm with 15 clusters is shown and in (b) the results of the GMM method with 15 clusters is shown.

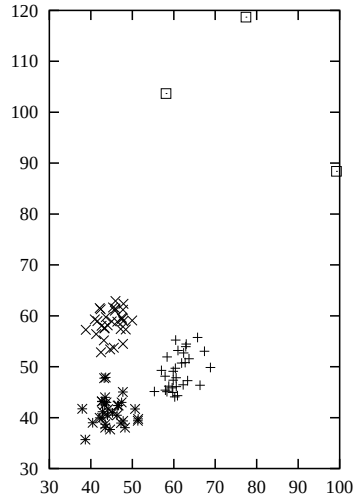
method	parameter	hit	M	time (ms)
MVC	$\sigma^2_{max} = 10$	100	15	122
MVC	$\sigma^2_{max} = 100$	100	8	99
K-means	$M = 15$	3	–	20
K-means	$M = 8$	10	–	12
GMM	$M = 15$	4	–	280
GMM	$M = 8$	10	–	160

Table 1: Statistical results of applying the algorithms to the R15 data set.

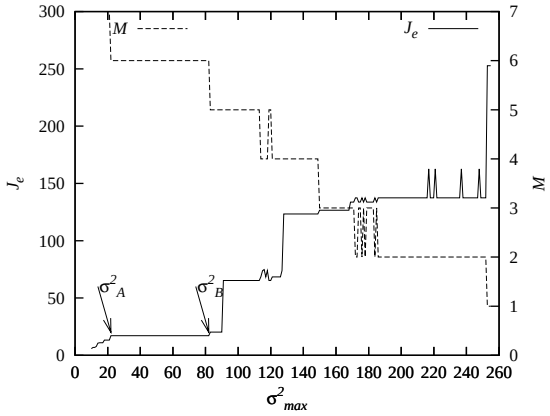
the total variance of the clusters lumped in the center is much higher than the variance of the outer clusters. The resulting clusterings for MVC ($\sigma^2_{max} = 10$), K-means ($M = 15$) and GMM ($M = 15$) were the same (Fig. 6), and also for MVC ($\sigma^2_{max} = 100$), K-means ($M = 8$) and GMM ($M = 8$). Table 1 shows that the MVC algorithm is clearly more robust in converging towards the (possibly local) minimum of its criterion. That is, the hit rate for the MVC algorithm is much higher than for the K-means and the GMM algorithm. Further, the K-means algorithm that is known to be efficient is indeed the fastest.

The next artificial data set consists of three clusters with some additional outliers (O3), see Fig. 7(a). Again, we first varied the σ^2_{max} parameter for the MVC algorithm in order to discover cluster tendencies. Although we roughly know the variance of the clusters, in this case it is certainly useful to search for the proper σ^2_{max} value, since the outliers may disrupt the original cluster variances.

Fig. 7(b) clearly shows only one prominent plateau [22.0...82.0]. This plateau is significant, because its strength is $S = 3.73$. In the corresponding clustering result all three outliers are put in separate clusters leading to a total of six clusters, as is shown in Fig. 8(a). Because the K-means



(a)



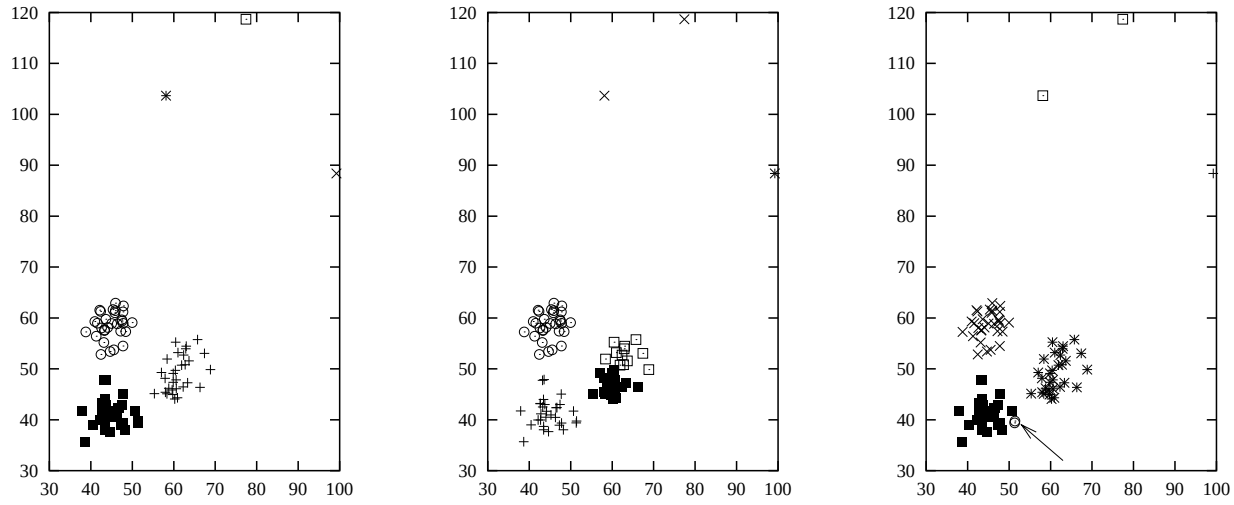
(b)

Figure 7: In (a) the O3 data set is shown which is generated as 3 similar 2-D Gaussian distributions with some additional outliers. In (b) J_e and M are displayed as a function of the σ^2_{max} constraint parameter.

algorithm does not impose a variance constraint it could find a lower square-error minimum and corresponding clustering with $M = 6$ than the MVC as can be seen in Fig. 8(b). The algorithm split one cluster instead of putting the outliers in separate clusters. This supports the statement that using one model for the detection of cluster tendencies and another for the clustering is undesirable. Also the GMM algorithm was not able to find the MVC solution (see Fig. 8(b)), though the MVC solution indeed had a higher likelihood. When $M = 3$, the K-means and the GMM algorithm merged two true clusters and put the outliers in one clusters. Table 2 shows the statistics of this experiment. Again, the K-means and GMM algorithm were clearly less robust in finding their respective (local) criterion optimum than the MVC and the K-means was the fastest.

We repeated this experiment several times with different generated clusters and outliers. The results were generally the same as described above, i.e. if there was a difference between the cluster results of the algorithms, the MVC handled the outliers better by putting them in separate clusters or it converged more often to its criterion optimum.

For the last synthetic experiment, we used a larger data set (D31) consisting of 31 randomly placed 2-D Gaussian clusters of 100 samples each, see Fig. 9(a). The tendency curve resulting from varying σ^2_{max} for the MVC algorithm shows one significant plateau [0.0030...0.0062] ($S = 2.07$), which corresponds to the original 31 clusters. Remarkably, the K-means and the GMM algorithm were not able to find the originating cluster structure, not even after 10000 trials. The statistical results in Table 3 show that the MVC algorithm consistently found the real structure, while the difference in

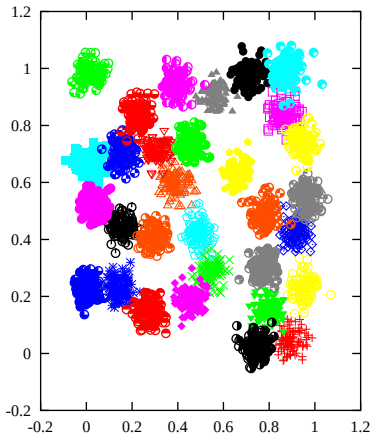


(a) $\sigma^2_{max} = 50$ (b) $M = 6$ (c) $M = 6$

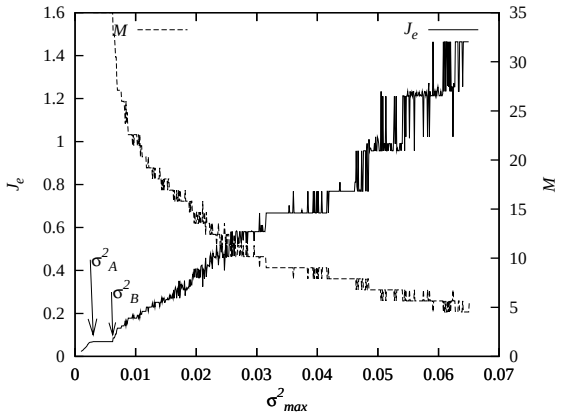
Figure 8: Results of applying the clustering algorithms to the O3 data set. In (a) the results of the MVC algorithm are shown resulting in 6 clusters. (b) and (c) show the results of the K-means and GMM algorithm, respectively. The GMM puts a remote cluster sample in a separate cluster.

method	parameter	hit	M	time (ms)
MVC	$\sigma^2_{max} = 50$	94	6	27
K-means	$M = 6$	1	–	2.5
K-means	$M = 3$	14	–	1.2
GMM	$M = 6$	1	–	17
GMM	$M = 3$	10	–	4.3

Table 2: Statistical results of applying the algorithms to the O3 data set. The hit rate of the GMM method with $M = 6$ certainly refers to a local maximum.



(a)



(b)

Figure 9: In (a) the D31 data set is shown which is generated as 31 similar 2-D Gaussian distributions. In (b) J_e and M are displayed as a function of the σ^2_{max} constraint parameter.

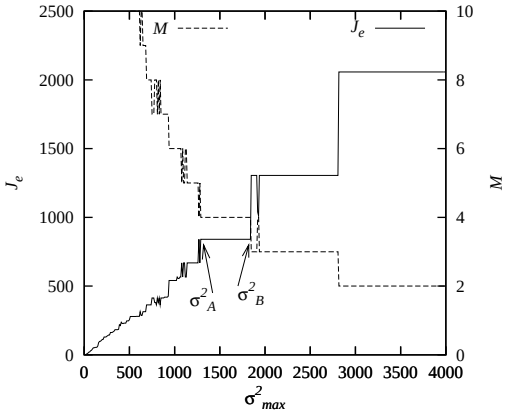
method	parameter	hit	M	time (ms)
MVC	$\sigma^2_{max} = 0.004$	100	31	930
K-means	$M = 31$	1	–	390
GMM	$M = 31$	1	–	220

Table 3: Statistical results of applying the algorithms to the D31 data set. The K-means and GMM algorithm were not able to find the originating structure, so the hit rate refers to a local optimum.

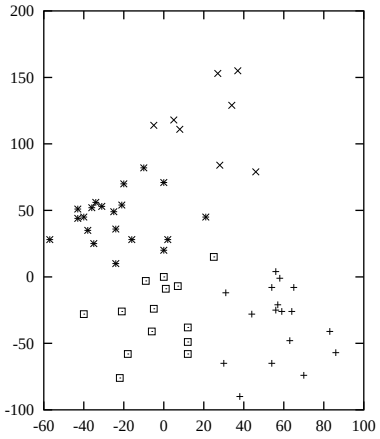
computation time between the algorithms becomes small.

Next, we applied the algorithms to some real data sets. We started with the *German Towns* data set which consists of 2-D coordinates of 59 German towns (pre-'Wende' situation). In order to find a significant clustering result, we again varied the σ^2_{max} parameter for the MVC algorithm. The resulting curves of J_e and M are displayed in Fig. 10(a). The two plateaus [1290...1840] and [1940...2810] have strengths $S = 1.43$ and $S = 1.45$ respectively. Although both plateaus are not significant, we show the clustering results of the first plateau with 4 clusters in Fig. 10(b), which equals the result of the K-means algorithm with $M = 4$. The GMM algorithm came up with a different solution consisting of three main clusters and one cluster containing a single sample. When we visually inspect the data in Fig. 10(b), we can conclude that it is certainly arguable if this data set contains significant structure. Table 4 shows similar hit rates as before and the K-means algorithm was again the fastest.

Finally, we processed the well-known *Iris* data set with both algorithms. The *Iris* data set is actually a labeled data set consisting of three classes of irises each characterized by four features. Fig. 11 illustrates the cluster tendencies resulting from varying σ^2_{max} for the MVC algorithm. The figure displays several plateaus, from which [0.76...1.39] and [1.40...4.53] are the strongest. The



(a)



(b) $\sigma^2_{max} = 1500$, $M = 4$

Figure 10: (a) shows J_e and M as a function of the σ^2_{max} constraint parameter for the *German Towns* data set. In (b) the clustering result of the MVC and the K-means algorithm with 4 clusters is displayed.

method	parameter	hit	M	time (ms)
MVC	$\sigma^2_{max} = 1500$	100	4	27
K-means	$M = 4$	29	–	0.56
GMM	$M = 4$	1	–	12

Table 4: Statistical results of applying the algorithms to the *German Towns* data set.

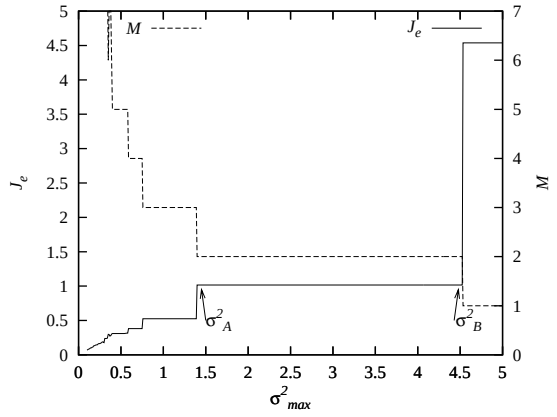


Figure 11: J_e and M as a function of the σ^2_{max} constraint parameter for the *Iris* data set.

method	parameter	hit	M	time (ms)
MVC	$\sigma^2_{max} = 1.0$	100	3	43
MVC	$\sigma^2_{max} = 2.0$	100	2	25
K-means	$M = 3$	36	–	1.6
K-means	$M = 2$	99	–	0.83
GMM	$M = 3$	8	–	4.2
GMM	$M = 2$	99	–	1.6

Table 5: Statistical results of applying the algorithms to the *Iris* data set.

plateaus with strengths $S = 1.83$ and $S = 3.24$ correspond to three and two clusters, respectively. Hence, only the latter is significant. All three algorithms found similar results for the same number of clusters. Since it is known that the three classes cannot be separated based on the given features, it is not surprising that the clustering with $M = 3$ does not correspond to the given labels. However, from the clustering with $M = 2$ (corresponding to the significant plateau), one cluster almost perfectly matches the samples of class I and the other cluster matches the samples of class II+III of the Iris class labels. The statistics in Table 5 show similar differences between the MVC, K-means, and GMM algorithm as in the other experiments.

4 Discussion

We presented a maximum variance cluster algorithm (MVC) for partitional clustering. In contrast to many other algorithms, the MVC algorithm uses a *maximum variance constraint* instead of the number of clusters as parameter. In the experiments, we showed that the method is effective in finding a proper clustering and we compared its results to those of the widely used K-means algorithm and the Gaussian mixtures modeling (GMM) method with likelihood maximisation with the EM algorithm.

In contrast to the proposed MVC method both the K-means and the GMM method need the number of clusters to be known a priori.

We showed that the MVC method copes better with *outliers* than the K-means algorithm. The GMM method is in principle able to separate the outliers, but has problems with the optimization process leading to convergence into local criterion optima. The MVC algorithm is more robust in finding the optimum of its criterion than both the K-means and GMM algorithm. We must note that other and better optimization schemes for both the K-means model (e.g. [5], [21]) and the Gaussian mixtures modeling (e.g. [17], [27]) have been developed. However, the improved optimization of these algorithms is achieved at the cost of (considerable) additional computation time or algorithm complexity.

The MVC algorithm is up to 100 times slower than the very efficient K-means algorithm, especially for small data sets and a low number of clusters. This is partially caused by the fact that we did not adjust the maximum number of epochs parameter E_{max} to the size of the data set. For larger data sets with a higher number of clusters the differences in computation time between both algorithms almost disappear. An advantage of the MVC algorithm with respect to computational efficiency is that it can be implemented on *parallel and distributed computer architectures* relatively easily. Accordingly, for large data sets the MVC algorithm may be advantageous also for efficiency reasons. In such a distributed computing environment, clusters can be maintained by separate processes. Then, only clusters that are neighbors communicate with each other. The main point of consideration will be how to balance the cluster processes on the available computers when clusters merge and when samples are isolated into new clusters.

An interesting property of the proposed method is that it enables the *assessment of cluster tendencies*. Generally, the curve resulting from varying the maximum variance constraint parameter as a function of the square-error displays some prominent plateaus that reveal the structure of the data. We indicated a way to find *significant* structure in the data by rating the strength of the plateaus. Accordingly, we were able to find proper settings of the maximum variance constraint parameter, which is the only model parameter.

A drawback of the MVC algorithm may be that it uses a distance rank list for every sample. The size of this rank list grows proportional to the square of the number of samples, so the amount of storage needed can become substantially. The main problem, however, lies in the computation of these rank lists. Since these lists are sorted, their construction costs $O(Nlog(N))$ operations. In order to prevent the rank list from becoming a bottleneck for the application of the MVC algorithm, a maximum distance constraint d_{max} can be imposed in addition to the maximum cluster variance constraint, e.g. $d_{max} = 2\sigma_{max}$. Then only those samples need to be ranked that are within the d_{max} range of the reference sample.

References

[1] R. Adams and L. Bishop. Seeded region growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):641–647, 1994.

[2] P. Andrey and P. Tarroux. Unsupervised image segmentation using a distributed genetic algorithm. *Pattern Recognition*, 27(5):659–673, 1994.

[3] P. Andrey and P. Tarroux. Unsupervised segmentation of markov random field modeled textured images using selectionist relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):252–262, March 1998.

[4] P.J. Angeline and J.B. Pollack. Competitive environments evolve better solutions for complex tasks. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 264–270, 1993.

[5] G.P. Babu, N. Murty, and S.S. Keerthi. A stochastic connectionist approach for global optimization with application to pattern clustering. *IEEE Transactions on Systems, Man, and Cybernetics—Part B*, 30(1):10–24, 2000.

[6] J.C. Bezdek and N.R. Pal. Some new indexes of cluster validity. *IEEE Transactions on Systems, Man, and Cybernetics — Part B*, 28(3):301–315, 1998.

[7] D.E. Brown and C.L.Huntley. A practical application of simulated annealing to clustering. *Pattern Recognition*, 25(4):401–412, 1992.

[8] D.L. Davies and D.W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):224–227, April 1979.

[9] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38, 1977.

[10] A. di Nola, V. Loia, and A. Staiano. Genetic-based spatial clustering. In *The Ninth IEEE International Conference on Fuzzy Systems*, pages 953–956, 2000.

[11] J. C. Dunn. Well separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, 4:95–104, 1974.

[12] K.C. Gowda and G. Krishna. Agglomerative clustering using the concept of mutual nearest neighborhood. *Pattern Recognition*, 10:105–112, 1978.

[13] L.O. Hall, I.B. Özyurt, and J.C. Bezdek. Clustering with a genetically optimized approach. *IEEE Transactions on Evolutionary Computation*, 3(2):103–112, july 1999.

[14] J.H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, 1975.

[15] S.L. Horowitz and T. Pavlidis. Picture segmentation by a tree traversal algorithm. *Journal of the ACM*, 23(2):368–388, 1976.

[16] L.J. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985.

[17] S. Ingrassia. A comparison between simulated annealing and the EM algorithms in normal mixtures decompositions. *Statistics and Computing*, 2:203–211, 1992.

[18] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall Inc., New Jersey, 1988.

[19] R.A. Jarvis and E.A. Patrick. Clustering using a similarity measure based on shared near neighbors. *IEEE Transactions on Computers*, 22:1025–1034, 1973.

[20] R.W. Klein and R.C. Dubes. Experiments in projection and clustering by simulated annealing. *Pattern Recognition*, 22(2):213–220, 1989.

[21] K. Krishna and M.N. Murty. Genetic K-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics—Part B*, 29(3):433–439, Jun 1999.

[22] T. Van Le. Evolutionary fuzzy clustering. In *Proceedings IEEE International Conference on Evolutionary Computation*, volume 2, pages 753–758, Killington, Vermont, USA, Oct. 14-16 1995. IEEE Computer Society Press.

[23] J. MacQueen. Some methods for classification and analysis of multivariate observations. In L.M. Le Cam and J. Neyman, editors, *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

[24] E.J. Pauwels and G. Frederix. Finding salient regions in images: Nonparametric clustering for image segmentation and grouping. *Computer Vision and Image Understanding*, 75(1,2):73–85, 1999.

[25] S.Z. Selim and K. Alsultan. A simulated annealing algorithm for the clustering problem. *Pattern recognition*, 24(10):1003–1008, 1991.

[26] S. Theodoridis and K. Koutroumbas. *Pattern Recognition*. Academic Press, London, 1999.

[27] D.M. Titterington, A.F.M. Smith, and U.E. Makov. *Statistical Analysis of Finite Mixture Distributions*. John Wiley and Sons, New York, 1990.

[28] C.J. Veenman, M.J.T. Reinders, and E. Backer. A cellular coevolutionary algorithm for image segmentation. *Submitted to IEEE Transactions on Image Processing*, 2001.

[29] L. Zhao, Y. Tsujimura, and M. Gen. Genetic algorithm for fuzzy clustering. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 716–719, 1996.