Proceedings of the IASTED International Conference
SIGNAL PROCESSING AND COMMUNICATIONS
September 19-22, 2000, Marbella, Spain

MINIMIZATION OF THE VALUE OF DAVIES-BOULDIN INDEX

ISMO KÄRKKÄINEN and PASI FRÄNTI
Department of Computer Science, University of Joensuu
P.O.Box 111, FIN-80101 Joensuu, FINLAND

ABSTRACT

We study the clustering problem when using Davies-Bouldin index as the optimization criterion. The problem is to partition a given data set of N vectors into M clusters so that the value of the Davies-Bouldin index is minimized. The index differs from the mean square error in that it also takes into account the distance between code vectors. This leads to the problem that setting the code vectors as the centroids of the clusters does not give the optimal placement for minimizing the Davies-Bouldin index. We derive formula for optimizing the location of the cluster centroids.

Keywords: pattern recognition, algorithms, cluster analysis, number of clusters.

1. INTRODUCTION

Unsupervised classification is an important part of many applications in image processing and analysis. The main goal is to obtain a high quality *clustering* for a set of input vectors minimizing a given optimization criterion [1]. Clustering is a combinatorial optimization problem where the aim is to partition a set of data vectors into a number of classes. Data vectors with similar features should be grouped together and vectors with different features to different groups.

There are several established methods for generating a clustering [1, 2, 3]. The methods are usually designed to minimize the mean square error between the data vectors and their cluster centroid. Mean square error, however, is not suitable for determining the number of clusters, since it decreases as the number of clusters increases. In fact, it would result in a situation where the number of clusters equals to the number of data vectors, which yields to error value of zero. We are therefore forced to use other criteria and this may lead to changes in the algorithm itself.

Generalized Lloyd algorithm (GLA) [4] is a widely used algorithm that seeks local optimum by iteratively changing between two representations. The GLA is also used as an integral part of several more sophisticated algorithms, such as stochastic relaxation [5], deterministic annealing [6], genetic algorithm [7], and randomized local search [8]. The optimization steps of the GLA are basically designed to the minimization of the mean square error. However, there is no guarantee that these steps would be optimal for other criteria, such as *Davies-Bouldin index* (DBI) [9].

In this study, we show that the centroid of the cluster is not the optimal selection of the cluster representative (code vector) for minimizing DBI. We then study the problem for the point of view of a single cluster. We show that the code vector of a cluster should be moved from the centroid to the opposite direction from the nearest neighbor cluster. We also derive formulas that give the exact amount of the movement. As a result, we obtain a method for optimizing the evaluation criterion directly. We then apply the proposed method with the clustering algorithm presented in [8].

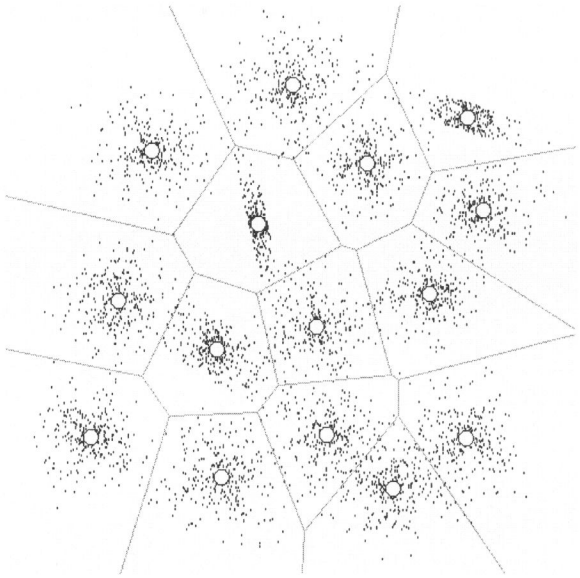

Figure 1: Example of the partition of a data set (the small black dots). The cluster centroids are represented by the white circles and the cluster boundaries by Voronoi diagram.

2. CLUSTERING PROBLEM

We use the following notations:

N	Number of data vectors.
n_i	Number of vectors in cluster i.
M	Number of clusters.
K	Number of attributes.
X	Set of N data vectors $X = \{x_1, ..., x_N\}$.
P	Set of N cluster indices $P = \{p_1, ..., p_N\}$.
C	Set of M code vectors, $C = \{c_1, ..., c_M\}$.
$d(a, b)$	Distance between vectors a and b.

Mean square error (MSE) is the average pairwise distance between data vectors and the corresponding cluster representative (code vector). Usually d is Euclidean distance, but other metrics are also used. Given the set of code vectors (C), and the set of cluster indices (P) for each training vector, MSE can be calculated as

$$MSE = \frac{1}{N}\sum_{i=1}^{N} d(x_i, c_{p_i})^2 \quad (1)$$

There exist several clustering algorithms, of which we now recall two. Both of the algorithms take the data set (X), the number of clusters (M), and an initial solution (e.g. random clustering) as input, and produce the partition (P), and/or the set of code vectors (C) as the output..

Generalized Lloyd algorithm, also known as *LBG* or *k-means*, starts with an initial codebook, which is iteratively improved by using two optimality conditions in turn. In the first step, each data vector x_i is mapped to the nearest code vector c_j. In the second step, a new set of code vectors is obtained by replacing each code vector as the centroid (average) of the data vectors in the cluster. The new solution is never worse than the previous one [3]. The solution converges to the nearest local optimum. The algorithm is easy to implement but very sensitive to the initial solution due to the fact that only local changes are made to the solution. Pseudocode of the GLA is presented in Figure 2.

```
GLA(X,C) return C,P
REPEAT
    C_previous ← C;
    FOR all i ∈ [1, N] DO
        p_i ← j such that x_i is closest to c_j;
    FOR all j ∈ [1, M] DO
        c_j ← Average of x_i, whose p_i = j;
UNTIL C = C_previous;
Return C,P;
```

Figure 2: Pseudocode for GLA.

Randomized local search (RLS) [8] overcomes the sensitivity to the initial solution. It starts with a random solution, which is then improved by a predefined number of iterations. At each step, a new candidate solution is constructed as follows. The clustering structure of the current solution is first modified by replacing a randomly chosen code vector by a randomly chosen input vector. The partition of the new solution is then adjusted in respect to the modified set of clusters. Two iterations of the GLA are applied to fine-tune the trial solution. The candidate is evaluated and accepted if it improves the previous solution. The algorithm is iterated for a fixed number of iterations. Pseudocode of the RLS algorithm is presented in Figure 3.

3. SOLVING THE NUMBER OF CLUSTERS

In order to determine the correct number of clusters, we must have a criterion that takes into account the number of clusters and an algorithm that varies the number of clusters. Notable property of MSE is that it's value decreases as the number of clusters increases. In fact, the optimal solution would be $N = M$, and the MSE = 0. This makes the MSE unsuitable for determining the number of clusters.

```
LS(X,M) return C, P
C ← Set of randomly chosen data vectors.
FOR all i ∈ [1, N] DO
    p_i ← j such that x_i is nearest to c_j;
FOR a ← 1 TO NumberOfIterations DO
    C_new ← C;
    c_j ← Randomly chosen data vector
    C_new, P_new ← GLA(X,C_new);
    IF MSE(X, C_new, P_new) < MSE(X, C, P)
    THEN C ← C_new; P ← P_new;
    END IF
END FOR
return C, P;
```

Figure 3: Pseudocode for local search.

We use Davies-Bouldin index [9] as the criterion because, in our experiments, we have found it in our experiments quite reliable among several alternative measures; with regard to pointing out the correct number of clusters. Davies-Bouldin index takes into account both the error caused by representing the data vectors with their cluster centroids (*intra cluster diversity*) and the distance between clusters (*inter cluster distance*). The intra cluster diversity of a cluster j is calculated as

$$MSE_j = \frac{1}{N}\sum_{p_i=j} d(c_j, x_i)^2 \quad (2)$$

The inter cluster distance of the cluster j and k is measured as the distance between their centroids c_j and c_k. With these two measures we can then calculate the *closeness* of the two clusters as the sum of their MSE-values divided by the distance of their centroids:

$$R_{j,k} = \frac{MSE_j + MSE_k}{|c_j - c_k|} \quad (3)$$

The closeness of the cluster is proportional to their MSE-values and indirectly proportional the distance of their centroids. Small values of $R_{j,k}$ indicate that the clusters are separated and large values that the clusters are close to each other.

In order to calculate DBI-value, we take for each cluster j the maximum value of (3) and denote it as R_j:

$$R_j = \max_{i \neq j} R_{i,j} \quad (4)$$

The overall Davies-Bouldin index is defined as:

$$DBI = \frac{1}{M} \sum_{i=1}^{M} R_i \quad (5)$$

The simplest algorithm that determines the correct number of clusters just varies M over a range of values $[M_{min}..M_{max}]$ in a loop and generates clustering for every number of M using any algorithm. We use the randomized local search (RLS) for generating the clustering for each value of M. We refer this approach as *brute force local search*. The pseudocode for the algorithm is presented in Figure 4.

```
BFLS(X, Mmin, Mmax) return C, P
Cbest, Pbest ← LS(X, Mmin);
FOR i ← Mmin + 1 TO Mmax DO
    C,P ← RLS(X, i);
    IF DBI(X,C,P) <DBI(X, Cbest, Pbest) THEN
        Cbest ← C; Pbest ← P;
END FOR
return Cbest, Pbest;
```

Figure 4: pseudocode for brute force local search.

4. SEEKING FOR OPTIMAL DBI

The data required to evaluate the clustering is the same for both MSE and DBI. Both measures need the cluster representatives (code vectors) and the partition, and this information is already available in the GLA and the RLS algorithms. The optimal partition can also be generated in the same way for both measures. Partitioning is performed by mapping each vector to the cluster that has the closest code vector.

$$p_i \leftarrow \underset{1 \leq j \leq M}{\operatorname{argmin}} d(x_i, c_j)^2 \; \forall i \in [1, N] \quad (6)$$

It is therefore straightforward to apply DBI instead of MSE in the BFLS algorithm.

The main problem lies in the fact that while we are using DBI in the BFLS algorithm for evaluating the solutions we are still using MSE in the RLS and GLA to refine the solutions. Minimizing MSE certainly minimizes the denominator of (3), but it does not do the same for the nominator. I.e., the code vectors may move closer to each other when we apply the GLA, but for DBI, it might be better to move the code vectors away from each other by a small amount. Although moving the code vector may increase MSE, the move is beneficial because the offset in the increase of the distance between code vectors offsets this increase in MSE, resulting in an overall decrease of DBI.

We propose to use DBI not only to select among the several clustering with various number of clusters, but also for optimal placement the code vectors. In order to do this, we develop a method to place the code vectors so that the increase in MSE_i does not offset the increase in the distance between the code vectors. We must therefore determine the direction and the magnitude of the shift. We will present formulas for local optimizations only; i.e., in the case of each cluster separately. The globally optimal settlement of the vectors is much more difficult to obtain because the shift of a single code vector may cause changes in the optimal placement for another vectors. Moreover, it is also possible that after moving a code vector of a cluster j away from its neighboring cluster k, the cluster k will no longer be the nearest cluster for j.

We will show later that the direction of the movement doesn't affect the increase of MSE, provided that the original placement of the code vector is the centroid of the cluster. We will also show that the optimal direction of the movement for minimizing DBI is the opposite from the direction to the centroid of the nearest cluster. We presume that the distance function in use is the Euclidean distance.

Lemma 1: The direction of the movement is the opposite from the centroid of the nearest neighbor cluster.

Lemma 2: The magnitude of the optimal movement, i.e. the length of the vector to be added to the code vector is:

$$|\delta| = \sqrt{|c_j - c_k|^2 + (MSE_j + MSE_k)} - |c_j - c_k| \quad (7)$$

We prove the lemmas in the Chapter 5. The effect of the movement is illustrated in Figure 5.

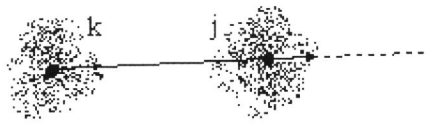

Figure 5: The direction of the movement.

The movement is implemented inside the GLA as an additional fine-tuning step, as shown in Figure 6. In this case, the code vectors are moved after each GLA iteration. Another approach would be to make the fine-tuning outside the GLA in the main loop of the RLS algorithm. However, this is only a matter of implementation as we expect the effect to be the same in both approaches.

The routine *FineTuneCodeVectors* refines the location of the code vectors. However, it is not clear how the fine-tuning should be implemented as there are a number of practical problems. For M clusters, we can draw a graph that shows the relationships of the clusters, in the sense how they contribute to the R_j-values of other clusters. The graph might look like that of in Figure 7.

GLA(X,C) return C,P
REPEAT
$C_{previous} \leftarrow C$;
FOR all $i \in [1, N]$ DO
$p_i \leftarrow j$ such that x_i is closest to c_j;
FOR all $j \in [1, M]$ DO
$c_j \leftarrow$ Average of x_i, whose $p_i = j$;
FineTuneCodeVectors(C,P,X);
UNTIL $C = C_{previous}$;
Return C,P;

Figure 6: pseudocode for modified GLA.

For each cluster, we face the problem of which one to move. In some cases the selection is easy. For example, when considering Figure 7, it is better to move A, since moving B might move it closer to D and thus worsen the situation there. Using the same logic, it is better to move E, H and G than to try to move F, since most likely F would move towards some other code vector. If we think of the number of edges coming to and leaving a node, we can construct a heuristic that the nodes with less edges should be moved before others. This seems to be a good rule, but it still does not guarantee an optimal placement of the vectors. For example, it is possible that the code vector j moves too far from its neighbor k, and as a consequence, the $R_{j,k}$ decreases so much that it will no longer be the maximum. For example, moving C might cause it to become closer to A, which means that we should move C the amount where $R_{C,D} = R_{C,A}$ and not any further.

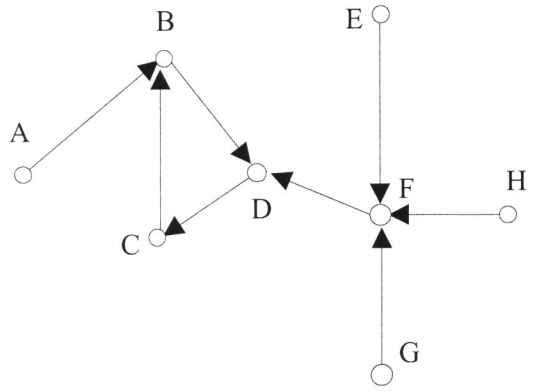

Figure 7: Nearest neighbor graph of the clusters demonstrating the contributors for R_j for each cluster.

We implemented two different methods. The first method (*simple method*) just takes into account the code vector of the nearest cluster (with which the maximum of $R_{j,k}$). The second method (*averaging method*) takes into account all clusters that had their maximum $R_{j,k}$ obtained with the cluster in question. The displacement is then calculated for all cluster pairs and then the average of these vectors was taken as the final displacement.

We also experimented with various ways to weigh the vectors in order to get at least the locally optimal solution but we did not found any feasible way to compute the globally optimum placement (found with exhaustive search in our tests). Among the experimented methods we used (i) per-cluster MSEs as defined by equation (2) and (ii) the R_j-values (4) for the neighbor clusters in question. These two experimental methods produced an offset vector that had more incorrect magnitude and more incorrect direction than the averaging method. Thus, the use of a more complex algorithm did not seem to be justified.

5. PROOF OF LEMMAS 1 AND 2

First we show that the increase in MSE_j does not depend on the direction of the movement. Then we determine the direction into which the code vector should be moved, and finally, we solve the magnitude of the movement.

Lemma 3: Increase in MSE depends only on the magnitude of the movement vector, provided that the original code vector is the average of the training vectors in cluster.

Proof of lemma 3: Let MSE_j be the per-cluster MSE with the code vector c_j and let δ be the movement vector that is to be added to c_j. The MSE for the cluster j is:

$$\begin{aligned} MSE_j &= \frac{1}{n_j}\sum_{p_i=j} |c_j - x_i|^2 \\ &= \frac{1}{n_j}\sum_{p_i=j}\sum_{k=1}^{K} \sqrt{(c_{jk} - x_{ik})^2}^2 \\ &= \frac{1}{n_j}\sum_{p_i=j}\sum_{k=1}^{K} (c_k^2 + x_{ik}^2 - 2c_{jk}x_{ik}) \end{aligned} \tag{8}$$

Using this, we derive formula for the change of MSE_j when the code vector is moved to point $c_j + \delta$ from the point c_j:

$$\begin{aligned} \Delta MSE_\delta &= MSE_{j+\delta} - MSE_j \\ &= \frac{1}{n_j}\sum_{p_i=j}\sum_{k=1}^{K}\left((c_{jk} + \delta_k)^2 + x_{ik}^2 - 2(c_{jk} + \delta_k)x_{ik}\right) \\ &\quad - \frac{1}{n_j}\sum_{p_i=j}\sum_{k=1}^{K}\left(c_{jk}^2 + x_{ik}^2 - 2c_{jk}x_{ik}\right) \\ &= \frac{1}{n_j}\sum_{p_i=j}\sum_{k=1}^{K}\left(\delta_k^2 + 2c_{jk}\delta_k - 2\delta_k x_{ik}\right) \\ &= \sum_{k=1}^{K}\left(\frac{1}{n_j}\sum_{p_i=j}\delta_k \cdot \delta_k + \frac{1}{n_j}\sum_{p_i=j}2c_{jk}\cdot\delta_k \right. \\ &\quad \left. + \frac{1}{n_j}\sum_{p_i=j}2\delta_k \cdot x_{ik}\right) = \delta\cdot\delta + 2c_j\cdot\delta - 2\delta\cdot\bar{x}_j \end{aligned} \tag{9}$$

By the assumption that c_j equals to the average of the points ($\bar{x}$) in the cluster, we get:

$$\Delta MSE_{\delta} = \delta \cdot \delta = |\delta|^2 \tag{10}$$

Thus, the direction of δ has no effect on the MSE and the only affecting factor is the magnitude of the vector δ.

Proof of lemma 1: In order to minimize $R_{j,k}$ we must maximize the distance between the code vectors c_j and c_{stat}. We presume that only one code vector (c_j) is being moved and the (c_{stat}) other remains stationary. Logically, δ should point away from c_{stat}. Let g be the vector c_j - c_{stat}. We determine the angle α between the vectors δ and g in order to maximize the distance r, see Figure 8. The length of δ can be assumed to be fixed.

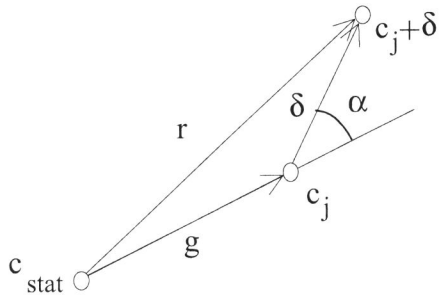

Figure 8: Elements for determining the angle of α.

Since the distance is always non-negative we can solve the maxima of the square distance between $c_j+\delta$ and c_{stat}:

$$\begin{aligned} |r| &= \sqrt{(|g|+|\delta|\cos\alpha)^2 + (|\delta|^2 \sin^2\alpha)} \\ |r|^2 &= |g|^2 + |\delta|^2 \cos^2\alpha + |g||\delta|\cos\alpha \\ &\quad + |\delta|^2 \sin^2\alpha \\ &= |g|^2 + |\delta|^2 (\cos^2\alpha + \sin^2\alpha) + |g||\delta|\cos\alpha \\ &= |g|^2 + |\delta|^2 + |g||\delta|\cos\alpha \end{aligned} \tag{11}$$

The lengths of the original distance vector g and the movement vector δ are positive constants. The only variable in equation (11) is α, and therefore it is easy to see that the maxima is obtained when cos α=1. Thus, the optimal value is α=0, which proves the lemma 1.

Proof of lemma 2: The next question is to determine how much the code vector should be moved. This depends on the per-cluster MSEs.

We must find $|\delta|$ that minimizes the $R_{j,k}$ when c_j is moved $|\delta|$ units away from c_k. The direction of δ is now fixed and we need to determine its length only. The increase in MSE_j is $|\delta|^2$ according to (10), and the increase in the cluster distance is $|\delta|$, since g and δ have the same direction. Thus, the new R_j value after the movement is:

$$R_j' = \frac{MSE_j + MSE_k + |\delta|^2}{|g| + |\delta|} \tag{12}$$

To find out the $|\delta|$ that minimizes (13) we take its derivative and find its zero-crossing points. This yields to:

$$R_j' d|\delta| = \frac{|\delta|^2 + 2|g||\delta| - (MSE_j + MSE_k)}{(|g| + |\delta|)^2} \tag{13}$$

The divisor is non-zero because the code vectors are not identical Therefore, we need to find the zero-crossing points of the denominator. Since it is a second-order polynomial with regards to $|\delta|$, we can solve it using formula:

$$\begin{aligned} &\frac{-2|g| \pm \sqrt{4|g|^2 + 4(MSE_j + MSE_k)}}{2} \\ &= -|g| \pm \sqrt{|g|^2 + MSE_j + MSE_k} \end{aligned} \tag{14}$$

Since negative vector lengths make no sense, we get (7), and the lemma 2 is therefore proven.

6. Test results

We have tested the proposed methods with two different training sets. The first set is the one shown in Figure 1, and the second set is in Figure 9. The data sets are artificially generated so that they contain 15 clusters. The clusters are clearly visible in the first set. In the second set, the clusters overlap and there are no clear boundaries, but it is still possible to identify the clusters.

Figure 9: The second data set.

We ran the BFLS algorithm 100 times with and without the optimizations related to the DBI. The same settings with regard to the number of iterations and the range for the number of clusters were used in all tests. For each number of clusters, we applied the local search algorithm using 5000 iterations and for each iteration, two iterations of the GLA were performed.

The DBI-values of the original method are summarized in Figures 10 and 11 for the range from 13 to 17 clusters. The correct number of clusters (15) was found in all the 100 runs when applied to the first data set. Thus, the method can distinguish the correct number of clusters quite easily with training sets where the clusters are more clearly separated.

In the case of the second set, the distinction is not as clear as the DBI-values are virtually equal for M=14 and M=15, see Figure 11. The method finds the correct number of clusters only in 18 times out of the 100 test runs. The resulting clustering is shown in Figure 9, where the missing cluster should be in the middle.

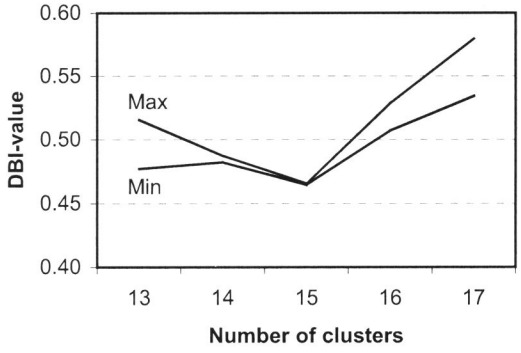

Figure 10: DBI-values for the test set 1 using the original method. The results are from 100 test runs.

The reason for poor performance might have been the low number of GLA-iterations. We therefore tried to improve the method by increasing the number of the GLA-iterations to see whether it would solve the problem. We applied 10 GLA-iterations for each local search iterations. The results are shown in Figure 12, where the black bars represent the modified method. The result clearly shows that the modification did not help but, on the contrary, it made the problem even worse.

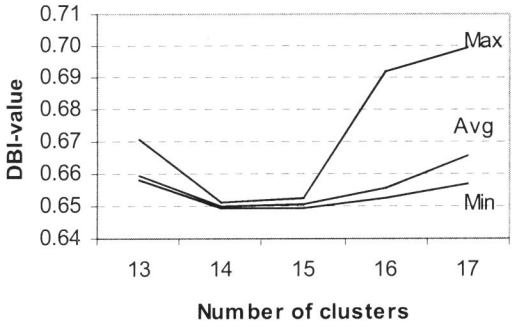

Figure 11: DBI-values for the test set 2 using the original method. The results are from 100 test runs.

The results of the new methods (*simple method* and *averaging method*) are summarized in Figure 13. They show that without the improvements, only 18 runs out of 100 found the correct number. When applying the simple optimization, the results do not change much from the original. The correct number of clusters is found 16 times. When applying the more averaging method, the correct number of clusters is found 17 times.

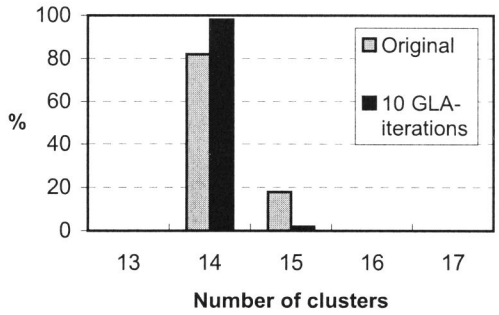

Figure 12: The number of times the different numbers of clusters are found with the original method by increasing the number of the GLA iterations.

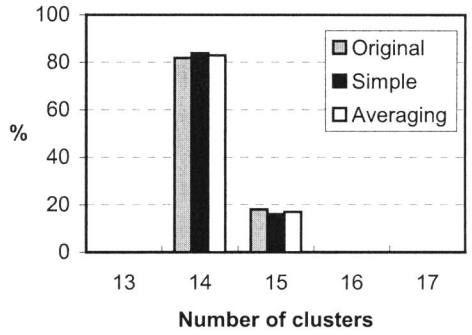

Figure 13: The number of times the different numbers of clusters ar found with different optimizations.

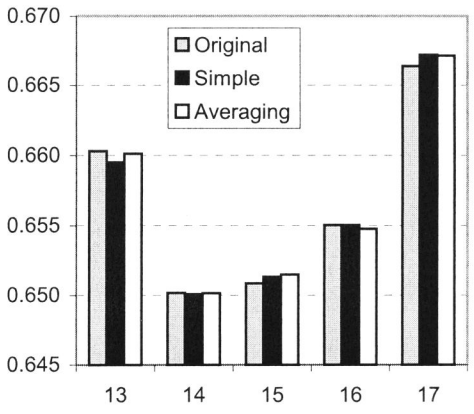

Figure 14: Average DBI-values with different optimizations.

The results of optimizations are shown in figures 13 and 14. Figure 14 shows how the average value of the DBI changes with the optimizations. The results for the correct number of clusters are worse than without optimizations. However, the reason for this seems to be that the maxima for each number of clusters are much greater with optimizations than without. The minima and median values are in fact smaller, but not much., the difference was at best 0.04 %.

7. CONCLUSIONS

We have derived formula for optimizing the location of the cluster centroids using Davies-Bouldin index. The formula was then applied in the clustering when solving the correct number of clusters. The results were theoretically well argued but their impact in the practical application was not significant. From the experiments, we can draw the following tentative conclusions.

Firstly, it seems that the DBI works well when the clusters are clearly separable. The suboptimality of the centroid location seems not to be a serious problem when we use the brute force search strategy in the clustering. The problem arises only in extreme situations when the clusters are greatly overlapping.

Secondly, it is possible that the formula giving the locally optimal location of a single code vector is not sufficient. Instead, globally optimum placement of the code vectors could resolve the problem but it is computationally not a feasible solution. Third possible reason is the fact that these optimizations sometimes produce DBI-values that are clearly worse than those obtained without. Moreover, it is uncertain whether the DBI itself is the correct measure as it has some heuristic elements. For example, it considers for each cluster only the nearest cluster in the measurement. Further studies would therefore be needed to reach a conclusive answer to the problem.

References:

1. B.S. Everitt: *Cluster Analysis.* 3rd edition, (London: Edward Arnold / Halsted Press, 1993).
2. L. Kaufman and P.J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*, (New York: John Wiley Sons, 1990).
3. A. Gersho and R.M. Gray, *Vector Quantization and Signal Compression.* (Dordrecht: Kluwer Academic Publishers, 1992).
4. Y. Linde, A. Buzo, R.M. Gray: An algorithm for vector quantizer design. *IEEE Transactions on Communications* **28** (1), 1980, 84-95.
5. K. Zeger and A. Gersho: Stochastic relaxation algorithm for improved vector quantizer design, *Electronics Letters*, **25** (14), 1989, 896-898.
6. K. Rose: Deterministic annealing for clustering, compression, classification, regression, and related optimization problems, *Proc. of the IEEE* **86** (11), Nov. 1998, 2210-2239.
7. P. Fränti, J. Kivijärvi, T. Kaukoranta, O. Nevalainen: Genetic algorithms for large scale clustering problems, *The Computer Journal* **40** (9), 1997, pp. 547-554.
8. P. Fränti and J. Kivijärvi: Randomized local search algorithm for the clustering problem, *Pattern Analysis and Applications*, 2000 (to appear).
9. D.L. Davies and D.W. Bouldin: A cluster separation measure, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **1** (2), 1979, 224-227.