Improving K-Means by Outlier Removal

Ville Hautamäki, Svetlana Cherednichenko, Ismo Kärkkäinen,
Tomi Kinnunen, and Pasi Fränti

Speech and Image Processing Unit,
Department of Computer Science, University of Joensuu,
P.O. Box 111, FI-80101, Joensuu, Finland
{villeh, schered, iak, tkinnu, franti}@cs.joensuu.fi

Abstract. We present an Outlier Removal Clustering (ORC) algorithm that provides outlier detection and data clustering simultaneously. The method employs both clustering and outlier discovery to improve estimation of the centroids of the generative distribution. The proposed algorithm consists of two stages. The first stage consist of purely K-means process, while the second stage iteratively removes the vectors which are far from their cluster centroids. We provide experimental results on three different synthetic datasets and three map images which were corrupted by lossy compression. The results indicate that the proposed method has a lower error on datasets with overlapping clusters than the competing methods.

1 Introduction

K-means [1] is an iterative clustering algorithm widely used in pattern recognition and data mining for finding statistical structures in data. K-means takes a training set $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ and a desired number of clusters M as its input and produces a *codebook* $C = \{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_M\}$. The elements $\boldsymbol{c}_i$ are called *code vectors*, and they define partition of the input space into disjoint sets $\{P_1, \ldots, P_M\}$ so that $P_i = \{\boldsymbol{x} \in \mathbb{R}^d; \|\boldsymbol{x} - \boldsymbol{c}_i\| \leq \|\boldsymbol{x} - \boldsymbol{c}_j\| \;\; \forall j \neq i\}$. These sets are called *clusters*. As the clusters are disjoint, each input vector $\boldsymbol{x}_i \in X$ belongs to exactly one cluster, the one whose code vector is nearest to $\boldsymbol{x}_i$. Cluster index is denoted here as $p_i \in \{1, 2, \ldots, M\}$. The *size* of a cluster is defined as the number of input vectors assigned to the cluster and will be denoted here as n_i.

K-means starts with an initial codebook C and partition P it and improves them iteratively. If K-means has been initialized well, the resulting code vectors tend to be located at locally dense regions of the input space. In this way, K-means can be considered as a nonparametric *density estimator* which attempts to fit a model C into the input data. Text-independent speaker recognition is an example in which K-means is used in the role of a density estimator [2]. There each cluster can be thought as roughly representing a single phonetic class, and each codebook representing the distribution of the speaker's characteristic vocal space.

Given the density estimation property of K-means, it is desirable that the code vectors would be located at the correct locations, or in other words, at

H. Kalviainen et al. (Eds.): SCIA 2005, LNCS 3540, pp. 978–987, 2005.

the centres of the actual clusters that generated X. However, even if the initial code vectors would be located *exactly* at the true locations, there is no guarantee that these would be the final estimated centroids. This happens more easily for overlapping clusters, see Fig. 1 for an illustration. In this figure, two Gaussian clusters having same variance have been generated by drawing $N = 200$ independent random samples from each class, and the distance between the mean vectors is varied in the three panels. The triangles denote the true centroids of the clusters and the open circles denote the estimated centroids using K-means. The true mean vectors have been used as the initial codebook to K-means. The dotted line shows the Voronoi partitioning based on the empirical centroids.

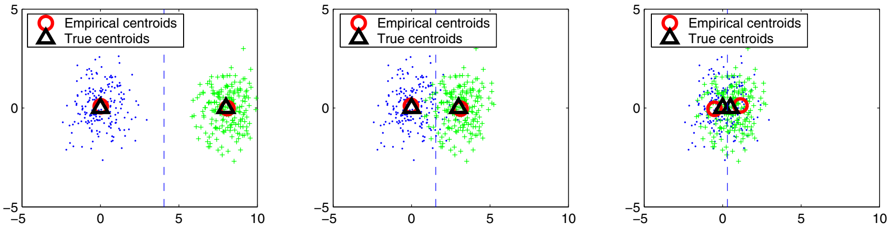

Fig. 1. Problem of K-means as a density estimator

In this trivial case, we can observe that for an increased cluster overlap, the estimated cluster centroids deviate more from the true ones. The reason for this is a fundamental problem of K-means: the clusters are disjoint, and the training vectors assigned to a wrong cluster deviate the estimated centroids away from the true ones. Figure 2 shows the error between the estimated codebook and the true codebook as a function of the distance between the cluster centers $\|\boldsymbol{c}_1 - \boldsymbol{c}_2\|$ and the ratio of their standard deviations σ_1/σ_2. Based on these observations, the usefulness of K-means as a density estimator is questionable.

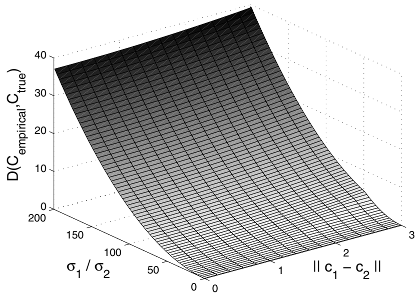

Fig. 2. Error between the true and estimated models

Parametric and fuzzy clustering models such as *Gaussian Mixture Models* (GMM) algorithm [3] and *Fuzzy C-means* (FCM) [4] can be used to attack the problem of nonoverlapping clusters. However, K-means remains probably the most widely used clustering method, because it is simple to implement and provides reasonably good results in most cases. In this paper, we improve the K-means based density estimation by embedding a simple *outlier removal* procedure in the algorithm. The proposed method iteratively removes vectors far from the currently estimated codevectors. By modifying the training set X in this way, we compensate for the nonoverlapping cluster assumption. However, the algorithm should remove also points that are in the overlapping regions of clusters.

2 Outlier Removal Followed by Clustering

Outlier is defined as a noisy observation, which does not fit to the assumed model that generated the data. In clustering, outliers are considered as observations that should be removed in order to make clustering more reliable [5].

In outlier detection methods based on clustering, outlier is defined to be an observation that does not fit to the overall clustering pattern [6]. The ability to detect outliers can be improved using a combined perspective of outlier detection and clustering. Some clustering algorithms, for example DBSCAN [7] and ROCK [8], handle outliers as special observations, but their main concern is clustering the dataset, not detecting outliers.

Outlier Detection using Indegree Number (ODIN) [9] is a local density-based outlier detection algorithm. Local density based scheme can be used in cluster thinning. Outlier removal algorithm can remove vectors from the overlapping regions between clusters, if the assumption holds that the regions are of relatively low density. Higher density is found near the cluster centroid. An obvious approach to use outlier rejection in the cluster thinning is as follows: (i) eliminate outliers (ii) cluster the data using any method.

Algorithm 1. ODIN+K-means(k, T)

```
$\{\mathrm{ind}(\boldsymbol{x}_i) | i = 1, \ldots, N\} \leftarrow$ Calculate kNN graph
for $i \leftarrow 1, \ldots, N$ do
    $o_i \leftarrow 1 / (\mathrm{ind}(\boldsymbol{x}_i) + 1)$
    if $o_i > T$ then
        $X \leftarrow X \setminus \{\boldsymbol{x}_i\}$
    end if
end for
$(C, P) \leftarrow$ K-means($X$)
```

In this paper, we compare the proposed method against aforementioned scheme, in which the outlier removal method is ODIN and the clustering algorithm K-means. In ODIN, the outliers are defined using a k-nearest neighbour (kNN) graph, in which every vertex represents a data vector, and the edges are pointers to neighbouring k vectors. The weight of an edge e_{ij} is $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|$.

In ODIN, the outlyingess of $\boldsymbol{x}_i$ is defined as:

$$o_i = \frac{1}{\mathrm{ind}(\boldsymbol{x}_i) + 1} \ , \tag{1}$$

where $\mathrm{ind}(\boldsymbol{x}_i)$ is the *indegree* of the vertex $\boldsymbol{x}_i$, i.e. the number of edges pointing to $\boldsymbol{x}_i$. In the first step of ODIN, a kNN graph is created for the dataset X. Then, each vertex is visited to check if its outlyingness is above threshold T. Fig. 3 shows an example of kNN graph and the outlyingness values calculated for three vectors.

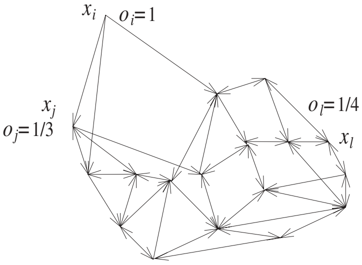

Fig. 3. Example of outlyingness factors in ODIN

3 Proposed Method

The objective of the proposed algorithm that we call *outlier removal clustering* (ORC), is to produce a codebook as close as possible to the mean vector parameters that generated the original data. It consists of two consecutive stages, which are repeated several times. In the first stage, we perform K-means algorithm until convergence, and in the second stage, we assign an outlyingness factor for each vector. Factor depends on its distance from the cluster centroid. Then algorithm iterations start, with first finding the vector with maximum distance to the partition centroid $d_{\max}$:

$$d_{\max} = \max_i\{\|\boldsymbol{x}_i - \boldsymbol{c}_{p_i}\|\}, \quad i = 1, \ldots, N \ . \tag{2}$$

Outlyingness factors for each vector are then calculated. We define the outlyingness of a vector $\boldsymbol{x}_i$ as follows:

$$o_i = \frac{\|\boldsymbol{x}_i - \boldsymbol{c}_{p_i}\|}{d_{\max}} \ . \tag{3}$$

We see that all outlyingness factors of the dataset are normalized to the scale $[0, 1]$. The greater the value, the more likely the vector is an outlier. An example of dataset clustered in three clusters and calculated outlyingness factors is shown in Fig. 4.

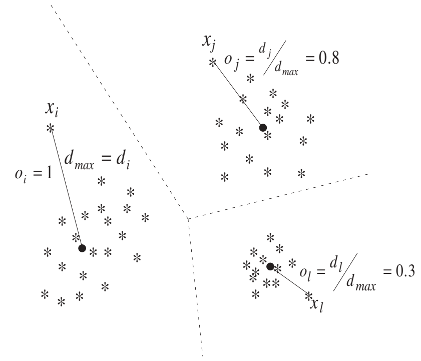

Fig. 4. Example of outlyingness factors in ORC

Algorithm 2. ORC(I, T)

```
$C \leftarrow$ Run K-means with multiple initial solutions, pick best $C$
for $j \leftarrow 1, \ldots, I$ do
    $d_{\max} \leftarrow \max_i\{\|\boldsymbol{x}_i - \boldsymbol{c}_{p_i}\|\}$
    for $i \leftarrow 1, \ldots, N$ do
        $o_i = \|\boldsymbol{x}_i - \boldsymbol{c}_{p_i}\| \big/ d_{\max}$
        if $o_i > T$ then
            $X \leftarrow X \setminus \{\boldsymbol{x}_i\}$
        end if
    end for
    $(C, P) \leftarrow$ K-means$(X, C)$
end for
```

The vectors for which $o_i > T$, are defined as outliers and removed from the dataset. At the end of each iteration, K-means is run with previous the C as the initial codebook, so new solution will be a fine-tuned solution for the reduced dataset. By setting the threshold to $T < 1$, at least one vector is removed. Thus, increasing the number of iterations and decreasing the threshold will in effect remove more vectors from the dataset, possibly all vectors.

Fig. 5 shows an example of running the proposed method on a dataset with strongly overlapping cluster so that even the cluster boundaries are not easily observable. The black dots are the original centroids. We see that with 40 iterations clusters are little bit separated and with 70 iterations clusters are totally separated.

4 Experiments

We run experiments on three synthetic datasets denoted as A1, S3 and S4 [10], which are shown in Fig. 6 and summarized in Table 1. The original cluster

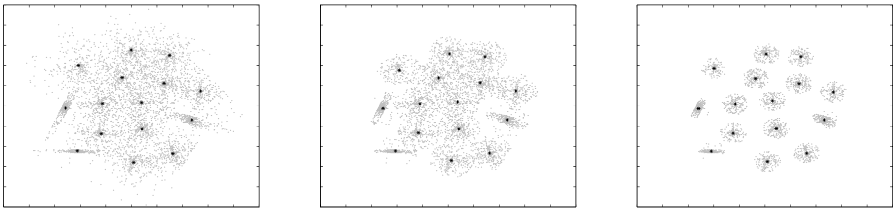

Fig. 5. Example of ORC. Original dataset (left), after 40 iterations (center), and after 70 iterations (right). The removal threshold is set to $T = 0.95$ in all cases

centroids are also shown in the same figure. Vectors in datasets are drawn from multinormal distributions. In dataset A1, the clusters are fairly well separated. In dataset S3, the clusters are slightly overlapping, and in dataset S4, the clusters are highly overlapping.

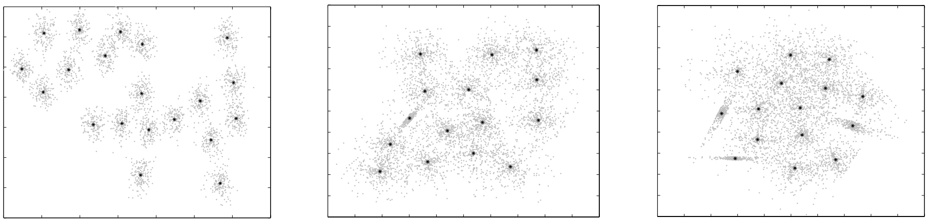

Fig. 6. Datasets, A1 (left), S3 (center) and S4 (right)

We run experiments also on three map image datasets (M1, M2 and M3), which are shown in Fig. 7. Map images are distorted by compressing them with a JPEG lossy compression method. The objective is to use color quantization for finding as close approximation of the original colors as possible. JPEG compression of map images creates so-called *ringing* around the edges due to the quantization of the cosine function coefficients. In [11], color quantization methods were used to find the original colors. We apply the proposed algorithm to this problem, and we assume that the number of colors is known in advance.

We calculate *mean absolute error* (MAE) to measure the difference between the empirical codebook and the generative codebook. For ODIN with K-means, we vary both the neighbourhood size k and the number of vectors removed. For ORC, we vary the number of iterations I and the threshold T.

Table 1. Summary of the datasets

Dataset	N	M	Dataset	N	M
A1	3000	20	M1	73714	5
S3	5000	15	M2	126246	6
S4	5000	15	M3	69115	5

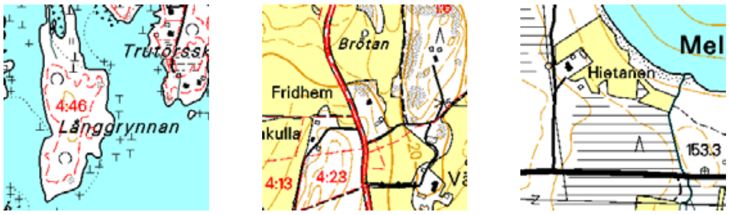

Fig. 7. Sample 256x256 pixel fragment from the test images M1 (left), M2 (center) and M3 (right)

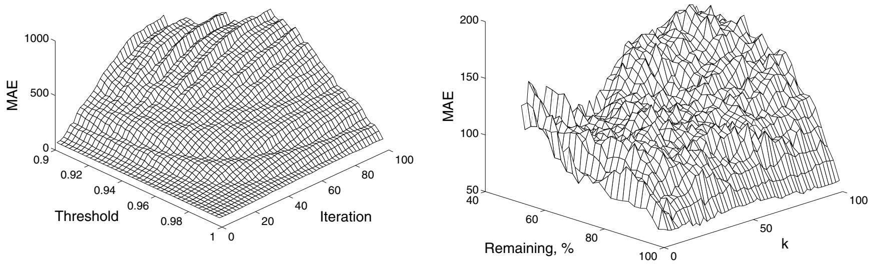

Fig. 8. Results for the dataset A1 for ORC (left) and for ODIN with K-means (right)

4.1 Results

Fig. 8 shows the results for the dataset A1. We observe that increasing the parameters in the algorithms increases the error. Fig. 9 shows the results for the dataset S3. Situation without ORC iterations and threshold is shown in the back corner (in contrast to the previous figure, due to the shape of the error surface). ODIN has two “valleys”, where distortion values are lower, but the error is consistently decreasing as iterations proceed or the threshold is decreased.

Fig. 10 shows the results for the dataset S4. Again, ODIN with K-means has two “valleys” where the error is lower. Regarding the number of remaining vectors, we see that the more vectors we remove with the ORC algorithm, the

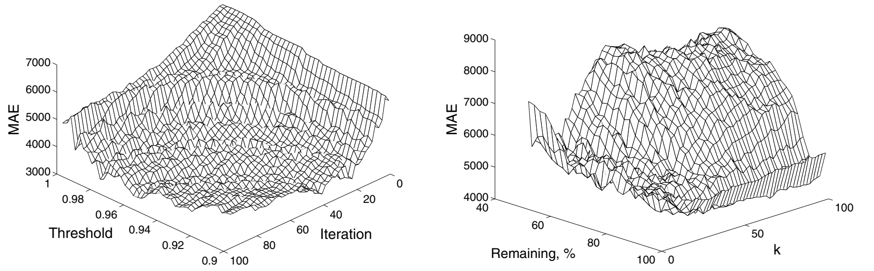

Fig. 9. Results for the dataset S3 for ORC (left) and for ODIN with K-means (right)

better the accuracy will be. This is because the ORC algorithm works as designed for S4 dataset by removing vectors that reside between clusters. On the other hand, when increasing parameters in ODIN algorithm first, we get lower error and then the error starts to increase.

Results for the M1 - M3 datasets running the ORC algorithm are presented in Figs. 11 and 12. We note that for all the test cases, ORC reaches lower error when the number of iterations is increased enough or the threshold is decreased. Error surface of the dataset M1 has an interesting behaviour, where error first increases and then it starts to decrease. Error surfaces for ODIN are omitted as with all parameter combinations the error increases in relation to the standard K-means.

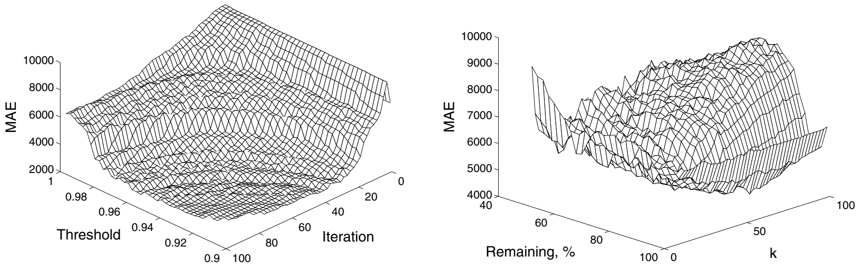

Fig. 10. Results for the dataset S4 for ORC (left) and for ODIN K-means (right)

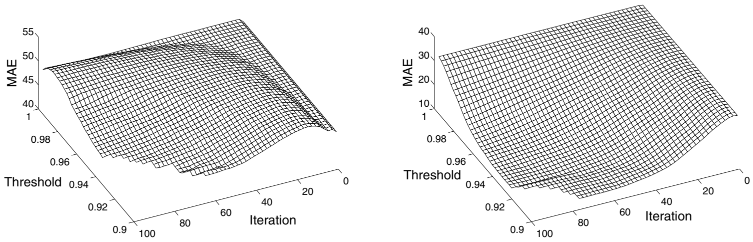

Fig. 11. Results for the datasets M1 (left) and M2 (right) for ORC

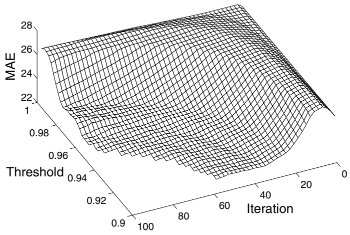

Fig. 12. Results for the dataset M3 for ORC

Table 2. Best MAEs obtained

Algorithm	A1	S3	S4	M1	M2	M3
Plain K-means	60	5719	7100	47	32	26
ODIN + K-means	58	4439	4754	61	48	45
ORC	56	3329	2813	45	13	23

In Table 2, we show the smallest MAE between original codebook and those obtained by using K-means, ODIN with K-means and ORC. The results indicate potential of the proposed method. The ORC algorithm outperforms the traditional K-means and K-means preceded by outlier removal for all three data sets. For the non-overlapping data set (A1), the results are close to each other. However, when cluster overlap is increased, the proposed algorithm shows substantially improved performance over the baseline methods. For the most difficult data set (S4), the proposed method gives 1.5 - 2 times smaller error. Although parameter setting might be a difficult depending on the dataset. For the map image datasets, ORC performs systematically better than K-means in all cases. With datasets M1 and M3, ORC and K-means are close to each other in performance, but for M2 ORC more than halves the error in relation to K-means.

5 Conclusions

In this paper, we have proposed to integrate outlier removal into K-means clustering (ORC) for nonparametric model estimation. The proposed method was also compared with the standard K-means without outlier removal, and a simple approach in which outlier removal precedes the actual clustering. The proposed method was evaluated on three synthetic data sets with known parameters of the generative distribution and three map image datasets with known cluster centroids.

The test results show that the method outperforms the two baseline methods, particularly in the case of heavily overlapping clusters. A drawback is that correct parameter setting seems to depend on the dataset. Thus, the parameter setting should be automatized in future.

References

1. Linde, Y., Buzo, A., Gray, R.M.: An algorithm for vector quantizer desing. IEEE Transactions on Communications **28** (1980) 84–95
2. Kinnunen, T., Karpov, E., Fränti, P.: Real-time speaker identification and verification. IEEE Transactions on Speech and Audio Processing (2005) Accepted for publication.
3. Dempster, A., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society B **39** (1977) 1–38

4. Dunn, J.: A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. Journal of Cybernetics **3** (1974) 32–57
5. Guha, S., Rastogi, R., Shim, K.: CURE an efficient clustering algorithm for large databases. In: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, Seattle, Washington (1998) 73–84
6. Zhang, T., Ramakrishnan, R., Livny, M.: BIRCH: A new data clustering algorithm and its applications. Data Mining and Knowledge Discovery **1** (1997) 141–182
7. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: 2nd International Conference on Knowledge Discovery and Data Mining. (1996) 226–231
8. Guha, S., Rastogi, R., Shim, K.: ROCK: A robust clustering algorithm for categorical attributes. In: 15th International Conference on Data Engineering. (1999) 512–521
9. Hautamäki, V., Kärkkäinen, I., Fränti, P.: Outlier detection using k-nearest neighbour graph. In: 17th International Conference on Pattern Recognition (ICPR 2004), Cambridge, United Kingdom (2004) 430–433
10. Virmajoki, O.: Pairwise Nearest Neighbor Method Revisited. PhD thesis, University of Joensuu, Joensuu, Finland (2004)
11. Kopylov, P., Fränti, P.: Color quantization of map images. In: IASTED Conference on Visualization, Imaging, and Image Processing (VIIP'04), Marbella, Spain (2004) 837–842