Fast Fuzzy Clustering of Web Page Collections

Christian Borgelt and Andreas Nürnberger

Dept. of Knowledge Processing and Language Engineering
Otto-von-Guericke-University of Magdeburg
Universitätsplatz 2, D-39106 Magdeburg, Germany
{borgelt,nuernb}@iws.cs.uni-magdeburg.de

Abstract. We study an extension of learning vector quantization that draws on ideas from fuzzy clustering, enabling us to find fuzzy clusters of ellipsoidal shape with a competitive learning scheme. This approach may be seen as a kind of online fuzzy clustering, which can have advantages w.r.t. the execution time of the clustering algorithm. We demonstrate the usefulness of our approach by applying it to web page collections, which are, in general, difficult to cluster due to the high number of dimensions and the special distribution characteristics of the data.

1 Introduction

It is not difficult to see that classical *c-means clustering* [6, 4] and standard *learning vector quantization* applied to clustering [14, 15] are very similar: a point that one method converges to is a stable point of the other, in particular, if learning vector quantization is applied in batch mode. Since classical c-means clustering has been generalized to fuzzy clustering [1, 2, 11], the idea suggests itself to transfer some ideas that have been developed in fuzzy clustering to competitive learning, with the aim of achieving a higher flexibility.

In this paper we consider how shape and size parameters can be introduced into a fuzzified competitive learning scheme, so that we arrive at competitive learning clustering algorithms that may be seen as online versions of the more sophisticated fuzzy clustering approaches, like the Gustafson-Kessel algorithm [10] or the fuzzy maximum likelihood estimation (FMLE) algorithm [8]. The basic idea of this transfer is that the update of a reference vector in competitive learning can be seen as an exponential decay of information gained from data points processed in earlier steps—a scheme that may just as well be applied to a covariance matrix describing the size and shape of a cluster.

Such online clustering has at least two advantages for the application domain we are concerned with here, that is, for clustering collections of documents. The first is that, due to the fact that the cluster parameters are updated more often, while the greater part of the overhead comes from the computations of the distances between the data points and the cluster centers, it can be faster than standard fuzzy clustering. Secondly, this approach to clustering makes it easier to handle documents that become available in a true online fashion, because updates need only few documents, not the whole collection.

This paper is organized as follows: in Section 2 we briefly review some basics of fuzzy clustering. In Section 3 we transfer fuzzy clustering ideas to learning vector quantization and develop online update rules for the size and shape parameters of a reference vector, captured in a covariance matrix. In Section 4 we review pre-processing methods for documents and in particular the vector space model. In Section 5 we present experimental results of clustering web page collections and finally, in Section 6, we draw conclusions from our discussion.

2 Fuzzy Clustering

While most classical clustering algorithms assign each datum to exactly one cluster, thus forming a crisp partition of the given data, fuzzy clustering allows for *degrees of membership*, to which a datum belongs to different clusters [1, 2, 11]. Most fuzzy clustering algorithms are objective function based: they determine an optimal (fuzzy) partition of a given data set $\mathbf{X} = \{\boldsymbol{x}_j \mid j = 1, \ldots, n\}$ into c clusters by minimizing an objective function

$$J(\mathbf{X}, \mathbf{U}, \mathbf{C}) = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^{w} d_{ij}^{2}$$

subject to the constraints

$$\sum_{j=1}^{n} u_{ij} > 0, \text{ for all } i \in \{1, \ldots, c\}, \qquad \text{and} \tag{1}$$

$$\sum_{i=1}^{c} u_{ij} = 1, \text{ for all } j \in \{1, \ldots, n\}, \tag{2}$$

where $u_{ij} \in [0, 1]$ is the membership degree of datum $\boldsymbol{x}_j$ to cluster i and d_{ij} is the distance between datum $\boldsymbol{x}_j$ and cluster i. The $c \times n$ matrix $\mathbf{U} = (u_{ij})$ is called the *fuzzy partition matrix* and $\mathbf{C}$ describes the set of clusters by stating location parameters (i.e. the cluster center) and maybe size and shape parameters for each cluster. The parameter w, $w > 1$, is called the *fuzzifier* or *weighting exponent*. It determines the "fuzziness" of the classification: with higher values for w the boundaries between the clusters become softer, with lower values they get harder. Usually $w = 2$ is chosen. Hard clustering results in the limit for $w \to 1$. However, a hard assignment may also be determined from a fuzzy result by assigning each data point to the cluster to which it has the highest degree of membership.

Constraint (1) guarantees that no cluster is empty and constraint (2) ensures that each datum has the same total influence by requiring that the membership degrees of a datum must add up to 1. Because of the second constraint this approach is usually called *probabilistic fuzzy clustering*, since with it the membership degrees for a datum formally resemble the probabilities of its being a member of the corresponding clusters. The partitioning property of a probabilistic clustering algorithm, which "distributes" the weight of a datum to the different clusters, is due to this constraint.

Unfortunately, the objective function J cannot be minimized directly. Therefore an iterative algorithm is used, which alternately optimizes the membership degrees and the cluster parameters [1, 2, 11]. That is, first the membership degrees are optimized for fixed cluster parameters, then the cluster parameters are optimized for fixed membership degrees. The main advantage of this scheme is that in each of the two steps the optimum can be computed directly. By iterating the two steps the joint optimum is approached (although, of course, it cannot be guaranteed that the global optimum will be reached—the algorithm may get stuck in a local minimum of the objective function J).

The update formulae are derived by simply setting the derivative of the objective function J w.r.t. the parameters to optimize equal to zero (necessary condition for a minimum). Independent of the chosen distance measure we thus obtain the following update formula for the membership degrees [11]:

$$u_{ij} = \frac{d_{ij}^{-\frac{2}{w-1}}}{\sum_{k=1}^{c} d_{kj}^{-\frac{2}{w-1}}}, \tag{3}$$

that is, the membership degrees represent the relative inverse squared distances of a data point to the different cluster centers, which is a very intuitive result.

The update formulae for the cluster parameters, however, depend on what parameters are used to describe a cluster (location, shape, size) and on the chosen distance measure. Therefore a general update formula cannot be given. Here we briefly review the three most common cases: The best-known fuzzy clustering algorithm is the fuzzy c-means algorithm, which is a straightforward generalization of the classical crisp c-means algorithm. It uses only cluster centers for the cluster prototypes and relies on the *Euclidean distance*, i.e.,

$$d_{ij}^2 = d^2(\boldsymbol{x}_j, \boldsymbol{\mu}_i) = (\boldsymbol{x}_j - \boldsymbol{\mu}_i)^\top (\boldsymbol{x}_j - \boldsymbol{\mu}_i),$$

where $\boldsymbol{\mu}_i$ is the center of the i-th cluster. Consequently it is restricted to finding spherical clusters of equal size. The resulting update rule is

$$\boldsymbol{\mu}_i = \frac{\sum_{j=1}^{n} u_{ij}^w \boldsymbol{x}_j}{\sum_{j=1}^{n} u_{ij}^w}, \tag{4}$$

that is, the new cluster center is the weighted mean of the data points assigned to it, which is again a very intuitive result.

The Gustafson-Kessel algorithm [10] uses the *Mahalanobis distance*, i.e.,

$$d_{ij}^2 = d^2(\boldsymbol{x}_j, \boldsymbol{\mu}_i) = (\boldsymbol{x}_j - \boldsymbol{\mu}_i)^\top \boldsymbol{\Sigma}_i^{-1} (\boldsymbol{x}_j - \boldsymbol{\mu}_i),$$

where $\boldsymbol{\mu}_i$ is the cluster center and $\boldsymbol{\Sigma}_i$ is a cluster-specific covariance matrix with determinant 1 that describes the shape of the cluster, thus allowing for ellipsoidal clusters of equal size. This distance function leads to same update rule (4) for the clusters centers. The covariance matrices are updated according to

$$\boldsymbol{\Sigma}_i = \frac{\boldsymbol{\Sigma}_i^*}{\sqrt[m]{|\boldsymbol{\Sigma}_i^*|}} \qquad \text{where} \qquad \boldsymbol{\Sigma}_i^* = \frac{\sum_{j=1}^{n} u_{ij}^w (\boldsymbol{x}_j - \boldsymbol{\mu}_i)(\boldsymbol{x}_j - \boldsymbol{\mu}_i)^\top}{\sum_{j=1}^{n} u_{ij}^w} \tag{5}$$

and m is the number of dimensions of the data space. $\mathbf{\Sigma}_i^*$ is called the *fuzzy covariance matrix*, which is simply normalized to determinant 1 to meet the abovementioned constraint. Compared to standard statistical estimation procedures, this is also a very intuitive result. It should be noted that the restriction to cluster of equal size may be relaxed by simply allowing general covariance matrices. However, depending on the characteristics of the data, this additional degree of freedom can deteriorate the robustness of the algorithm.

Finally, the fuzzy maximum likelihood estimation (FMLE) algorithm [8] is based on the assumption that the data was sampled from a mixture of c multivariate normal distributions as in the statistical approach of mixture models [7, 3]. It uses a (squared) distance that is inversely proportional to the probability that a datum was generated by the normal distribution associated with a cluster, i.e.,

$$d_{ij}^2 = \left(\frac{\theta_i}{\sqrt{(2\pi)^m |\mathbf{\Sigma}_i|}} \exp\left(-\frac{1}{2} (\boldsymbol{x}_j - \boldsymbol{\mu}_i)^\top \mathbf{\Sigma}_i^{-1} (\boldsymbol{x}_j - \boldsymbol{\mu}_i) \right) \right)^{-1},$$

where θ_i is the prior probability of the cluster, $\boldsymbol{\mu}_i$ is the cluster center, $\mathbf{\Sigma}_i$ a cluster-specific covariance matrix, which in this case is not required to be normalized to determinant 1, and m the number of dimensions of the data space. For the FMLE algorithm the update rules are not derived from the objective function due to technical obstacles, but by comparing it to the well-known expectation maximization (EM) algorithm [5] for a mixture of normal distributions [7, 3], which, by analogy, leads to the same update rules for the cluster center and the cluster-specific covariance matrix [11]. The prior probability is, in direct analogy to statistical estimation, computed as

$$\theta_i = \frac{1}{n} \sum_{j=1}^{n} u_{ij}^w. \tag{6}$$

Since the high number of free parameters of the FMLE algorithm renders it unstable on certain data sets, it is usually recommended [11] to initialize it with a few steps of the very robust fuzzy c-means algorithm. The same holds, though to a somewhat lesser degree, for the Gustafson-Kessel algorithm.

It is worth noting that of both the Gustafson-Kessel as well as the FMLE algorithm there exist so-called *axes-parallel* versions, which restrict the covariance matrices $\mathbf{\Sigma}_i$ to diagonal matrices and thus allow only axes-parallel ellipsoids [12]. These variants have certain advantages w.r.t. robustness and execution time.

3 Learning Vector Quantization

Learning vector quantization [14, 15], in its classical form, is a competitive learning algorithm that has been developed in the area of artificial neural networks and that can be applied to classified as well as unclassified data. Here we confine ourselves to unclassified data, where the algorithm consists in iteratively updating a set of c so-called *reference vectors* $\boldsymbol{\mu}_i$, $i = 1, \ldots, c$, each of which

is represented by a neuron. For each data point $\boldsymbol{x}_j$, $j = 1, \ldots, n$, the closest reference vector (the so-called "winner neuron") is determined and then this reference vector (and only this vector) is updated according to

$$\boldsymbol{\mu}_i^{(\text{new})} = \boldsymbol{\mu}_i^{(\text{old})} + \eta_1 \left(\boldsymbol{x}_j - \boldsymbol{\mu}_i^{(\text{old})} \right), \tag{7}$$

where η_1 is a learning rate. This learning rate usually decreases with time in order to avoid oscillations and to enforce the convergence of the algorithm.

Membership degrees can be introduced into this basic algorithm in two different ways. In the first place, one may employ an activation function for the neurons, for which a radial function like the

$$\text{Cauchy function} \quad f(r) = \frac{1}{1+r^2} \qquad \text{or the Gaussian function} \quad f(r) = e^{-\frac{1}{2}r^2}$$

may be chosen, where r is the (radial) distance from the reference vector. In this case all reference vectors are updated for each data point, with the update being weighted with the value of the activation function. However, this scheme, which is closely related to *possibilistic fuzzy clustering* [16], usually leads to unsatisfactory results, since there is no dependence between the clusters, so that they tend to end up at the center of gravity of all data points. This corresponds to the fact that in possibilistic fuzzy clustering the objective function is truly minimized only if all cluster centers are identical [23]. Useful results are obtained only if the method gets stuck in a local minimum, which is an undesirable situation.

An alternative is to rely on a normalization scheme as in *probabilistic fuzzy clustering*, that is, to compute the weight for the update of a reference vector as the relative inverse (squared) distance from this vector (cf. the computation of the membership degrees in fuzzy clustering, see formula (3)), or as the *relative* activation of a neuron. This is the approach we employ here, that is, we use

$$\boldsymbol{\mu}_i^{(\text{new})} = \boldsymbol{\mu}_i^{(\text{old})} + \eta_1 u_{ij} \left(\boldsymbol{x}_j - \boldsymbol{\mu}_i^{(\text{old})} \right) \tag{8}$$

with u_{ij} defined as in equation (3). Furthermore we associate with each neuron not only a reference vector $\boldsymbol{\mu}_i$, but also a covariance matrix $\boldsymbol{\Sigma}_i$, which describes the shape and (if we do not require it to be normalized to determinant 1) the size of the represented cluster.

In order to find an update rule for this covariance matrix, we observe that the above equation (7) may also be written as

$$\boldsymbol{\mu}_i^{(\text{new})} = (1 - \eta_1)\, \boldsymbol{\mu}_i^{(\text{old})} + \eta_1\, \boldsymbol{x}_j,$$

which shows that the update can be seen as an exponential decay of information gained from data points processed earlier. Transferring this idea to the covariance matrices $\boldsymbol{\Sigma}_i$ and drawing on equation (5) leads directly to

$$\boldsymbol{\Sigma}_i^{(\text{new})} = (1 - \eta_2)\, \boldsymbol{\Sigma}_i^{(\text{old})} + \eta_2\, (\boldsymbol{x}_j - \boldsymbol{\mu}_i)(\boldsymbol{x}_j - \boldsymbol{\mu}_i)^\top, \tag{9}$$

where η_2 is a learning rate, which, in general, differs from the learning rate η_1 for the reference vectors. In the fuzzy case this update may be weighted, as the update of the reference vectors, by the relative inverse (squared) distance of the data point from the reference vector or by the relative neuron activation.

It should be noted that versions of this algorithm that require the covariance matrix to be normalized to determinant 1 or restrict the covariance matrix to a diagonal matrix may be considered, too. Such constraints can improve the robustness or the execution time of the algorithm. Furthermore it should be noted that the updates may be executed in batch mode, aggregating the changes resulting from the data points and actually updating the reference vectors and covariance matrices only at the end of an epoch.

Finally, it should be noted that the additional elements of the FMLE algorithm may also be transferred to a competitive learning scheme, by using the reciprocal of the special distance function as the activation function of the neurons. The additional parameter θ_i, that is, the weight or prior probability of a cluster, is then updated according to

$$\theta_i^{(\text{new})} = (1 - \eta_3)\, \theta_i^{(\text{old})} + \eta_3\, u_{ij},$$

where η_3 is another learning rate (different from η_1 and η_2) and u_{ij} is defined as in equation (3). However, in the application described below, we confine ourselves to the simpler approach discussed above, which in addition to membership degrees introduces only covariance matrices.

4 Clustering Document Collections

To be able to cluster text document collections with the methods discussed above, we have to map the text files to numerical feature vectors. Therefore, we first applied standard preprocessing methods, i.e. stopword filtering and stemming (using the Porter Stemmer [18]), encoded each document using the vector space model [19] and finally selected a subset of terms as features for the clustering process as briefly described in the following.

4.1 The Vector Space Model

The vector space model represents text documents as vectors in an m-dimensional space, i.e., each document j is described by a numerical feature vector $\boldsymbol{x}_j = (x_{j1}, \ldots, x_{jm})$. Each element of the vector represents a word of the document collection, i.e., the size of the vector is defined by the number of words of the complete document collection.

For a given document j the so-called weight x_{jk} defines the importance of the word k in this document with respect to the given document collection C. Large weights are assigned to terms that are frequent in relevant documents but rare in the whole document collection [20]. Thus a weight x_{jk} for a term k in document j is computed as the term frequency tf_{jk} times the inverse document frequency idf_k, which describes the term specificity within the document collection.

In [21] a weighting scheme was proposed that has meanwhile proven its usability in practice. Besides term frequency and inverse document frequency (defined as $\mathrm{idf}_k = \log(n/n_k)$), a length normalization factor is used to ensure that all documents have equal chances of being retrieved independent of their lengths:

$$x_{jk} = \frac{\mathrm{tf}_{jk} \log \frac{n}{n_k}}{\sqrt{\sum_{l=1}^{m} \left(\mathrm{tf}_{jl} \log \frac{n}{n_l}\right)^2}}, \tag{10}$$

where n is the size of the document collection C, n_k the number of documents in C that contain term k, and m the number of terms that are considered.

Based on a weighting scheme a document j is described by an m-dimensional vector $\boldsymbol{x}_j = (x_{j1}, \ldots, x_{jm})$ of term weights and the similarity S of two documents (or the similarity of a document and a query vector) can be computed based on the inner product of the vectors (by which — if we assume normalized vectors — the cosine between the two document vectors is computed), i.e.

$$S(\boldsymbol{x}_j, \boldsymbol{x}_k) = \sum_{l=1}^{m} x_{jl} \cdot x_{kl}. \tag{11}$$

For a more detailed discussion of the vector space model and weighting schemes see, for instance, [9, 20, 19].

Note that for normalized vectors the scalar product is not much different in behavior from the Euclidean distance, since for two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ it is

$$\cos\varphi = \frac{\boldsymbol{xy}}{|\boldsymbol{x}| \cdot |\boldsymbol{y}|} = 1 - \frac{1}{2}\, d^2\left(\frac{\boldsymbol{x}}{|\boldsymbol{x}|}, \frac{\boldsymbol{y}}{|\boldsymbol{y}|}\right).$$

Although the scalar product is faster to compute, it enforces spherical clusters. Therefore we rely on the Mahalanobis distance in our approach.

4.2 Index Term Selection

To reduce the number of words in the vector description we applied a simple method for keyword selection by extracting keywords based on their entropy. In the approach discussed in [13], for each word k in the vocabulary the entropy as defined by [17] was computed:

$$W_k = 1 + \frac{1}{\log_2 n} \sum_{j=1}^{n} p_{jk} \log_2 p_{jk} \quad \text{with} \quad p_{jk} = \frac{\mathrm{tf}_{jk}}{\sum_{l=1}^{n} \mathrm{tf}_{lk}}, \tag{12}$$

where tf_{jk} is the frequency of word k in document j, and n is the number of documents in the collection. Here the entropy gives a measure how well a word is suited to separate documents by keyword search. For instance, words that occur in many documents will have low entropy. The entropy can be seen as a measure of the importance of a word in the given domain context. As index words a number of words that have a high entropy relative to their overall frequency

Dataset Character	Dataset Category	Associated Theme
A	Commercial Banks	Banking & Finance
B	Building Societies	Banking & Finance
C	Insurance Agencies	Banking & Finance
D	Java	Programming Languages
E	C / C++	Programming Languages
F	Visual Basic	Programming Languages
G	Astronomy	Science
H	Biology	Science
I	Soccer	Sport
J	Motor Racing	Sport
K	Sport	Sport

Table 1. Categories and Themes of the used benchmark data set.

have been chosen, i.e. from words occurring equally often those with the higher entropy can be preferred. Empirically this procedure has been found to yield a set of relevant words that are suited to serve as index terms [13].

In order to obtain a fixed number of index terms that appropriately cover the documents, a greedy strategy was applied: From the first document in the set of documents select the term with the highest relative entropy as an index term. Then mark this document and all other documents containing this term. From the first of the remaining unmarked documents select again the term with the highest relative entropy as an index term. Then mark again this document and all other documents containing this term. Repeat this process until all documents are marked, then unmark all of them and start again. The process can be terminated when the desired number of index terms have been selected.

5 Experiments

For our experimental studies we chose the collection of web page documents used in [22].[1] The data set consists of 11,000 web pages classified into 11 equally-sized categories each containing 1,000 web documents. To each category one of four distinct themes, namely Banking and Finance, Programming Languages, Science, and Sport as shown in Table 1 was assigned. The authors of [22] reported baseline classification rates using c-means clustering and different preprocessing strategies (alternatively stopword filtering and/or stemming, different number of index terms).

In the following we present results we obtained with our algorithm using the preprocessing strategies described above. After stemming and stop word filtering we obtained 163,860 words. This set was further reduced by removing terms that are shorter than 4 characters and that occur less then 15 or more than $11,000/12 \approx 917$ times in the whole collection. Thus we made sure that no

[1] This collection is available for download from
`http://www.pedal.rdg.ac.uk/banksearchdataset/index.htm`.

words that perfectly separate one class from another are used in the describing vectors. From the remaining 10626 words we selected 400 words by applying the greedy index-term selection approach described in Section 4.2. For our clustering experiments we selected finally subsets of the 20, 50, 100, 150, ..., 350, 400 most frequent words in the subset to be clustered. Based on these words we determined vector space descriptions for each document (see Section 4.1, Equation (10)) that we used in our clustering experiments. All vectors were normalized to unit length.

To assess the clustering performance we computed the performance on the same data sets used in [22], i.e., we clustered the union of the dissimilar data sets A and I, and the semantically more similar data sets B and C. The mean classification performance reported in [22] using hard c-means are 67.51% for data sets A and I and 75.44% for data sets B and C, in both cases stemming and stopword filtering methods were applied. In a third experiment we used all classes and tried to find clusters describing the four main themes, i.e. banking, programming languages, science, and sport.

For our experiments we used c-means, fuzzy clustering and learning vector quantization methods with and without cluster centers normalized to unit length, with and without variances (i.e., spherical clusters and axes-parallel ellipsoids—diagonal covariance matrices—of equal size), and with the inverse squared distance or the Gaussian function for the activation. The learning vector quantization algorithm updated the cluster parameters once for every 100 documents. The results for some parameterizations of the algorithms are shown in Figures 1 to 3. All results are the mean values of ten runs, which differed in the initial cluster positions and the order in which documents were processed. The dotted lines show the default accuracy (obtained if all documents are assigned to the majority class). The diamonds show the average classification accuracy in percent (left axis) with bars indicating the standard deviation. The grey dots and lines show the average execution times in seconds (right axis).[2]

The top row of each figure shows the results for spherical clusters with cluster centers normalized to unit length (i.e., the centers resulting from the update formulae given above were divided by their lengths). In this case we used a Gaussian membership/activation function and a fixed cluster radius of $\frac{1}{3}$. (Note that with an inverse squared distance membership/activation, due to the normalization to sum 1, the radius has no effect, leading to considerably worse results for fuzzy clustering and learning vector quantization. Therefore we omit these results here.) As can be seen, all algorithms work reasonably well, with fuzzy clustering and learning vector quantization having a slight edge in accuracy over hard c-means clustering, which is particularly clear for the four cluster case (see Figure 3). It may be a little surprising that learning vector quantization not only outperforms fuzzy clustering w.r.t. the execution time, but can even compete with hard c-means clustering in this respect. Note that the execution times

[2] All experiments were carried out with a program written in C and compiled with gcc 3.3.3 on a Pentium 4C 2.6GHz system with 1GB of main memory running S.u.S.E. Linux 9.1. The program and its sources can be downloaded free of charge at `http://fuzzy.cs.uni-magdeburg.de/~borgelt/cluster.html`.

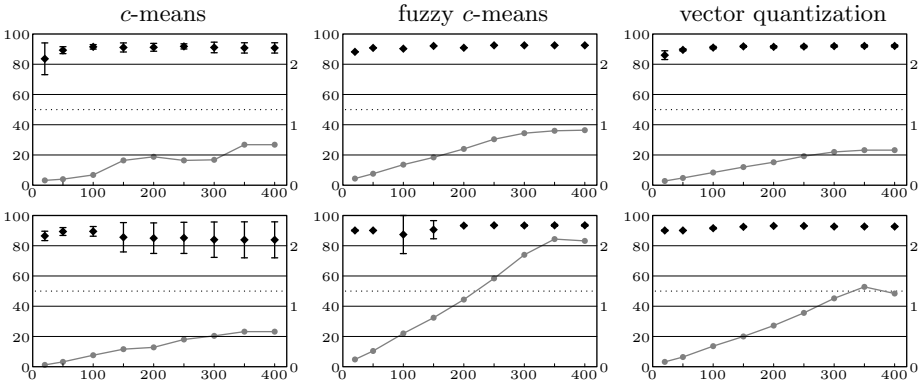

Fig. 1. Accuracy on commercial banks versus soccer (top row: normalized centers, fixed uniform variances; bottom row: free centers, adaptable variances).

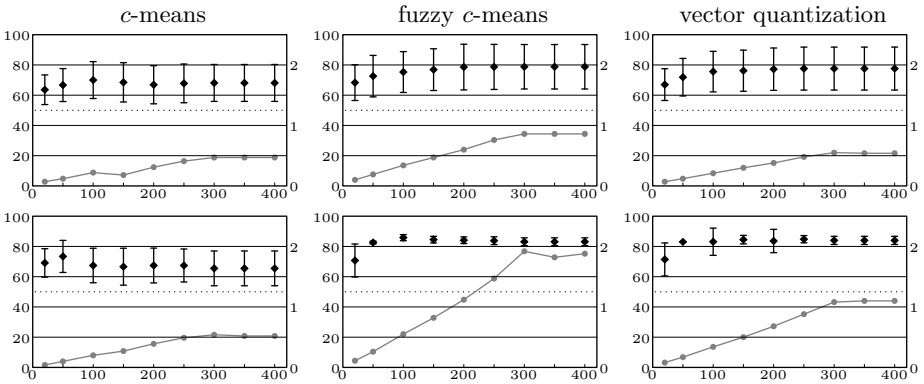

Fig. 2. Accuracy on building companies versus insurance agencies (top row: normalized centers, fixed uniform variances; bottom row: free centers, adaptable variances).

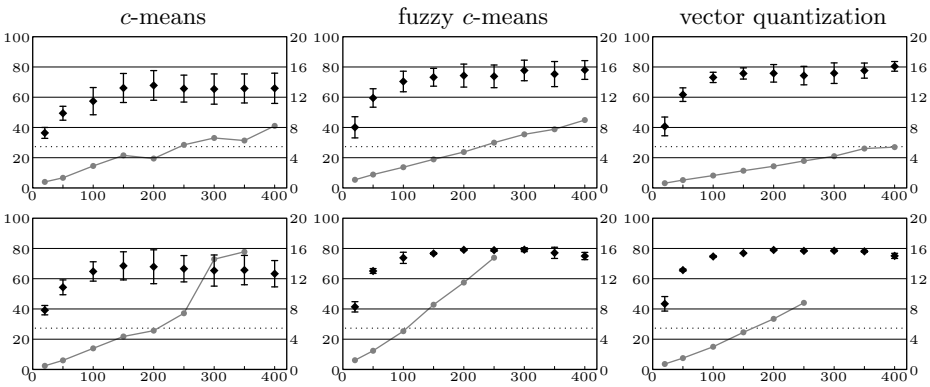

Fig. 3. Accuracy on major themes (four clusters; top row: normalized centers, fixed uniform variances; bottom row: free centers, adaptable variances).

level off for 350 and 400 words in Figures 1 and 2, because only 316 and 300 words appear in these document subsets, respectively.

The bottom row of each figure shows the results for free cluster centers (i.e., they are not normalized to unit length), axis-parallel ellipsoids (diagonal covariance matrices), and with the inverse squared distance for the membership/activation. Obviously, the performance of hard c-means is slightly worse in this case, while fuzzy c-means and learning vector quantization yield good result. In particular, the result achieved with learning vector quantization in the four class case (see Figure 3), which is very good and highly stable, is remarkable. This, however, comes at some cost w.r.t. the execution time of the algorithm.

It is interesting to note that in other experiments (which we cannot present here due to reasons of space) it turned out that using ellipsoidal clusters usually slightly deteriorates performance if the cluster centers are normalized, but improves performance for free cluster centers (i.e. no normalization to unit length).

6 Conclusions

In this paper we transferred some ideas from fuzzy clustering, in particular the use of a covariance matrix to describe the shape and the size of a cluster, to learning vector quantization. We developed a fuzzy competitive learning scheme for these new reference vector parameters and applied our algorithm to the difficult task of clustering document collections.

Our experiments showed that this approach can be used successfully for clustering collections of documents, with learning vector quantization leading to shorter execution times. In addition, learning vector quantization appears to be slightly more robust than fuzzy clustering, which is known to be considerably more robust than crisp clustering. Finally, it enables a truly "online" clustering, since only a fraction of the documents is needed for each update.

References

1. J.C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms.* Plenum Press, New York, NY, USA 1981
2. J.C. Bezdek, J. Keller, R. Krishnapuram, and N. Pal. *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing.* Kluwer, Dordrecht, Netherlands 1999
3. J. Bilmes. A Gentle Tutorial on the EM Algorithm and Its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. University of Berkeley, Tech. Rep. ICSI-TR-97-021, 1997
4. H.H. Bock. *Automatische Klassifikation.* Vandenhoeck & Ruprecht, Göttingen, Germany 1974
5. A.P. Dempster, N. Laird, and D. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society (Series B)* 39:1–38. Blackwell, Oxford, United Kingdom 1977
6. R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis.* J. Wiley & Sons, New York, NY, USA 1973

7. B.S. Everitt and D.J. Hand. *Finite Mixture Distributions*. Chapman & Hall, London, UK 1981
8. I. Gath and A.B. Geva. Unsupervised Optimal Fuzzy Clustering. *IEEE Trans. Pattern Analysis & Machine Intelligence* 11:773–781. IEEE Press, Piscataway, NJ, USA, 1989
9. W.R. Greiff. A Theory of Term Weighting Based on Exploratory Data Analysis. *Proc. 21st Ann. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval (Sydney, Australia)*, 17–19. ACM Press, New York, NY, USA 1998
10. E.E. Gustafson and W.C. Kessel. Fuzzy Clustering with a Fuzzy Covariance Matrix. *Proc. 18th IEEE Conference on Decision and Control (IEEE CDC, San Diego, CA)*, 761–766, IEEE Press, Piscataway, NJ, USA 1979
11. F. Höppner, F. Klawonn, R. Kruse, and T. Runkler. *Fuzzy Cluster Analysis*. J. Wiley & Sons, Chichester, England 1999
12. F. Klawonn and R. Kruse. Constructing a Fuzzy Controller from Data. *Fuzzy Sets and Systems* 85:177-193. North-Holland, Amsterdam, Netherlands 1997
13. A. Klose, A. Nürnberger, R. Kruse, G.K. Hartmann, and M. Richards. Interactive Text Retrieval Based on Document Similarities. *Physics and Chemistry of the Earth, Part A: Solid Earth and Geodesy* 25:649–654. Elsevier, Amsterdam, Netherlands 2000
14. T. Kohonen. *Learning Vector Quantization for Pattern Recognition*. Technical Report TKK-F-A601. Helsinki University of Technology, Finland 1986
15. T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, Heidelberg, Germany 1995 (3rd ext. edition 2001)
16. R. Krishnapuram and J. Keller. A Possibilistic Approach to Clustering, *IEEE Transactions on Fuzzy Systems*, 1:98-110. IEEE Press, Piscataway, NJ, USA 1993
17. K.E. Lochbaum and L.A. Streeter. Combining and Comparing the Effectiveness of Latent Semantic Indexing and the Ordinary Vector Space Model for Information Retrieval. *Information Processing and Management* 25:665–676. Elsevier, Amsterdam, Netherlands 1989
18. M. Porter. An Algorithm for Suffix Stripping. *Program: Electronic Library & Information Systems* 14(3):130–137. Emerald, Bradford, United Kingdom 1980
19. G. Salton, A. Wong, and C.S. Yang. A Vector Space Model for Automatic Indexing. *Communications of the ACM* 18:613–620 ACM Press, New York, NY, USA 1975
20. G. Salton and C. Buckley. Term Weighting Approaches in Automatic Text Retrieval. *Information Processing & Management* 24:513–523. Elsevier, Amsterdam, Netherlands 1988
21. G. Salton, J. Allan, and C. Buckley. Automatic Structuring and Retrieval of Large Text Files. *Communications of the ACM* 37:97–108. ACM Press, New York, NY, USA 1994
22. M.P. Sinka, and D.W. Corne. A large benchmark dataset for web document clustering. *A. Abraham, J. Ruiz-del-Solar, and M. Köppen (eds.), Soft Computing Systems: Design, Management and Applications*, 881–890. IOS Press, Amsterdam, The Netherlands 2002
23. H. Timm, C. Borgelt, and R. Kruse. A Modification to Improve Possibilistic Cluster Analysis. *Proc. IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE 2002, Honolulu, Hawaii)*. IEEE Press, Piscataway, NJ, USA 2002