Soft Vector Quantization and the EM Algorithm

Ethem Alpaydın*
Department of Computer Engineering
Boğaziçi University
TR-80815 Istanbul Turkey
Phone: ++ 90 (212) 263-1540 x 1862
Fax: ++90 (212) 287-2461
`Email: alpaydin@boun.edu.tr`

September 15, 1997

Running Title: Soft Vector Quantization and EM

Section: Mathematical and Computational Analysis

To appear in **Neural Networks**.

*This work is supported by Tübitak Grant EEEAG-143 and Boğaziçi University Research Funds 95HA0108. Thanks to Mike Jordan and Zoubin Ghahramani for stimulating discussions on Gaussian mixtures and the EM algorithm. In particular, it was Mike Jordan who pointed out the advantage of coupled learning over the uncoupled version. Thanks also to the anonymous referees for constructive comments.

Abstract

The relation between hard c-means (HCM), fuzzy c-means (FCM), fuzzy learning vector quantization (FLVQ), soft competition scheme (SCS) of Yair, Zeger, & Gersho (1992) and probabilistic Gaussian mixtures (GM) have been pointed out recently by Bezdek & Pal (1995). We extend this relation to their training, showing that learning rules by these models to estimate the cluster centers can be seen as approximations to the expectation-maximization (EM) method as applied to Gaussian mixtures. HCM and unsupervised LVQ use 1-of-c type competition. In FCM and FLVQ, membership is the $-2/(m-1)$th power of the distance. In SCS and GM, Gaussian function is used. If Gaussian membership function is used, the weighted within groups sum of squared errors used as the fuzzy objective function corresponds to the maximum likelihood estimate in Gaussian mixtures with equal priors and covariances. The fuzzy clustering method named fuzzy c-means alternating optimization procedure (FCM-AO) proposed to optimize the former is then equivalent to batch EM and SCS's update rule is a variant of the online version of EM. The advantages of the probabilistic framework are: (i) we no longer have spurious spread parameters that needs finetuning as m in fuzzy vector quantization or β in SCS; instead we have a variance term that has a sound interpretation and that can be estimated from the sample; (ii) EM guarantees that the likelihood does not decrease, thus it converges to the nearest local optimum; (iii) EM also allows us to estimate the underlying distance norm and the cluster priors which we could not with the other approaches. We compare Gaussian mixtures trained with EM with LVQ (HCM), SCS and FLVQ on the IRIS dataset and see that it is more accurate due to its being able to take into account the covariance information. We finally note that vector quantization is generally an intermediate step before finding a final output for which supervision may be possible. Thus instead of an uncoupled approach where an unsupervised method is used first to find the cluster parameters followed by supervised training of the mapping based on the memberships, we advocate a coupled approach where the cluster parameters and mapping are trained supervised in a coupled way. The uncoupled approach ignores the error at the outputs which may not be ideal.

Keywords. Vector quantization, clustering, competitive learning, unsupervised learning, mixture models, EM, maximum likelihood estimation, fuzzy logic.

1 Introduction

Vector quantization methods assume that the input is organized into a number of groups or is generated by one of a number of sources (Duda & Hart, 1973). These are also known as clusters, reference vectors or components. These clusters partition the input space among them where for any input we compute a value that denotes the association of that input to any group. This is done by defining a cluster center for each cluster and measuring the distance between an input and the cluster center using an appropriate metric. The association then is inversely proportional to this distance.

Bezdek and Pal (1995) made a recent study analyzing different methods for vector quantization. In the hard approach, each input is associated only to the group with the nearest center. Example is the k-means clustering (Duda & Hart, 1973) which is known as hard c-means (HCM) in the fuzzy set literature and unsupervised learning vector quantization (LVQ) (Kohonen, 1995).

In the soft approach, this association (or membership or probability of being generated by) is a value between zero and one with the frequent requirement that the memberships sum up to one. The soft variants are the fuzzy methods fuzzy c-means (FCM), fuzzy learning vector quantization (FLVQ), soft competition scheme (SCS) proposed by Yair, Zeger, & Gersho (1992) and the probabilistic Gaussian mixtures (GM).

Bezdek & Pal (1995) showed that these methods are related and in particular, they showed that the memberships computed by SCS are estimates of the posterior probabilities computed by the probabilistic Gaussian mixtures. In this article, we show that in terms of learning the cluster centers, the soft approaches of FCM, FLVQ, SCS and GM are related. SCS learning rule for the cluster centers is a variant of the online version of the expectation-maximization (EM) method applied to Gaussian mixtures. The fuzzy learning method Fuzzy c-Means Alternating Optimization (FCM-AO) and FLVQ (Bezdek & Pal, 1995) are approximations to the batch EM if the Gaussian membership function is used, assuming equal cluster priors and covariances.

In Section 2, we discuss the hard vector quantization method and in Section 3, we discuss fuzzy vector quantization methods FCM and FLVQ. In Section 4, the soft competition scheme and in Section 5, probabilistic Gaussian mixtures and the EM algorithm are discussed. We compare LVQ, SCS and FLVQ with Gaussian mixtures on the well-known IRIS dataset in Section 6. In Section 7, we discuss the supervised approach to train the cluster parameters and its advantage over the unsupervised approach. Conclusions are given in Section 8. The Appendix details the derivation of the EM algorithm for Gaussian mixtures.

2 Hard Vector Quantization

We are given a sample $\mathcal{X} = \{\boldsymbol{x}_k\}_{k=1}^n$ where $\boldsymbol{x}_k \in \Re^d$ and the aim of vector quantization is to find cluster centers, $\boldsymbol{v}_j, j = 1 \ldots c$ that best represents the density from which the sample is independently and identically drawn. In hard c-means and unsupervised LVQ, input is associated only with the cluster having the nearest center:

$$b_{kj} = \begin{cases} 1 & \text{if } \|\boldsymbol{x}_k - \boldsymbol{v}_j\| = \min_l \|\boldsymbol{x}_k - \boldsymbol{v}_l\| \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

The cluster center, $\boldsymbol{v}_j$, is the mean of all inputs associated with that cluster. In the batch algorithm hard c-means (HCM), we have

$$\boldsymbol{v}_j = \frac{\sum_k b_{kj}\boldsymbol{x}_k}{\sum_k b_{kj}} \tag{2}$$

Once the cluster centers are updated, b_{kj} values may change. So we have an iterative procedure where Eqs. (1) and (2) are alternated until centers do not change anymore. In the online, or sequential version, learning vector quantization (LVQ) (Kohonen, 1995), a small update is done at each step:

$$\Delta\boldsymbol{v}_j = \eta b_{kj}(\boldsymbol{x}_k - \boldsymbol{v}_j) \tag{3}$$

where η is a learning factor which is gradually decreased towards zero for convergence. Both of these approaches minimize the squared error:

$$\min_{\boldsymbol{v}} \sum_k \sum_j b_{kj} ||\boldsymbol{x}_k - \boldsymbol{v}_j||^2 \tag{4}$$

Note that this approach uses Euclidean distance as the metric and thus assumes equal variances on all dimensions and zero covariances.

3 Fuzzy Vector Quantization

In fuzzy clustering, the fuzzy membership of an input in a cluster decreases continuously as distance increases. There is a normalization to guarantee that the memberships sum up to one:

$$u_{kj} = \frac{(||\boldsymbol{x}_k - \boldsymbol{v}_j||_A)^{-\frac{2}{m-1}}}{\sum_{l=1}^{c} (||\boldsymbol{x}_k - \boldsymbol{v}_l||_A)^{-\frac{2}{m-1}}} \tag{5}$$

Thus not only the closest but all clusters are taken into account to represent the input position. A is a positive definite matrix that generalizes the Euclidean norm to the Mahalanobis norm, i.e., $||\boldsymbol{x}||_A \equiv \boldsymbol{x}^T A^{-1} \boldsymbol{x}$, thus allowing hyperellipsoidal clusters of arbitrary orientation, instead of hyperspheric ones. The batch rule, named fuzzy c-means alternating optimization (FCM-AO) algorithm, for learning cluster centers is:

$$\boldsymbol{v}_j = \frac{\sum_k (u_{kj})^m \boldsymbol{x}_k}{\sum_k (u_{kj})^m} \tag{6}$$

Bezdek & Pal (1995) write that alternating Eqs. (5) and (6), the weighted within groups sum of squared errors objective function is minimized:

$$\min_{\boldsymbol{v}} \sum_k \sum_j (u_{kj})^m ||\boldsymbol{x}_k - \boldsymbol{v}_j||_A^2 \tag{7}$$

No method is given to estimate A, the matrix defining the underlying norm. Bezdek & Pal (1995) note that the most problematical choice for the algorithm is the weighting exponent m, which can take any value between $(1, \infty)$; generally a value in the range $(1.1, 5]$ is used. When m is large, clusters have large spread and in the limiting case as m goes to infinity, all memberships are equal and all centers converge to the overall mean. In the other extreme when $m \xrightarrow{+} 1$, we have a hard competition between clusters and we get the hard c-means rule, where for any pattern, only one cluster is one and all the others are zero. In the descending fuzzy learning vector quantization ($\downarrow$FLVQ), one starts with a large value of m and decreases it gradually towards one. In Figure 1, u_{kj} values are drawn for a one-dimensional problem for different values of m.

An online version can also be derived where for each pattern $\boldsymbol{x}_k$, the centers are updated as follows:

$$\Delta \boldsymbol{v}_j = \eta (u_{kj})^m (\boldsymbol{x}_k - \boldsymbol{v}_j) \tag{8}$$

4 Soft Competition Scheme

In the Soft Competition Scheme (SCS) of Yair, Zeger, & Gersho (1992), the probability of the jth prototype winning the competition is given by a Gaussian which also decreases with increasing distance:

$$p_{kj} = \frac{\exp(-\beta ||\boldsymbol{x}_k - \boldsymbol{v}_j||^2)}{\sum_l \exp(-\beta ||\boldsymbol{x}_k - \boldsymbol{v}_l||^2)} \tag{9}$$

β is a parameter like m in fuzzy clustering which allows us to play with the spread of the clusters. In SCS, like in $\downarrow$LVQ, we start with large spread when many cluster centers are updated for a point and gradually decrease it to allow for specialization. This is implemented by starting with small β and gradually increasing it. In Figure 2, p_{kj} values are drawn for a one-dimensional problem for different values of β. The center update in SCS is done online as follows:

$$\Delta \boldsymbol{v}_j = \eta_{kj} p_{kj} (\boldsymbol{x}_k - \boldsymbol{v}_j) \tag{10}$$

η_{kj} is an adaptive learning rate proportional to $1/t$; it is taken to be roughly the reciprocal of the total number of times that $\boldsymbol{v}_j$ has been updated. It is reset to one whenever iteration counter is a perfect square. This divides the learning process into frames of increasing length where η, starting from one decreases towards zero, to allow for convergence for the current value of β. Bezdek & Pal (1995) mention the difficulty of choosing good parameter values for SCS.

5 Gaussian Mixtures and the EM Algorithm

In a mixture model, the probability density function is given by (Duda & Hart, 1973):

$$p(\boldsymbol{x}|\Phi) = \sum_{j=1}^{c} p(\boldsymbol{x}|\omega_j, \Phi) P(\omega_j) \tag{11}$$

where ω_j are the components, $P(\omega_j)$ are their prior probabilities, or mixing parameters, and $p(\boldsymbol{x}|\omega_j, \Phi)$ are the component densities, known up to a vector of parameters Φ. Sometimes Φ also includes $P(\omega_j)$. The log likelihood of the iid sample $\mathcal{X} = \{\boldsymbol{x}_k\}_k$ is

$$\begin{aligned} \mathcal{L}(\Phi|\mathcal{X}) &= \sum_k \log p(\boldsymbol{x}_k|\Phi) \\ &= \sum_k \log \sum_j p(\boldsymbol{x}_k|\omega_j, \Phi) P(\omega_j) \end{aligned} \tag{12}$$

This does not have a straightforward solution. The Expectation-Maximization (EM) algorithm (Dempster et al., 1977; Redner & Walker, 1984; Xu & Jordan, 1996) can be used which involves two steps. Here we only give the results; the derivation is deferred to the Appendix. In the E-step, we compute the posteriors:

$$P(\omega_j|\boldsymbol{x}_k, \Phi) = \frac{p(\boldsymbol{x}_k|\omega_j, \Phi) P(\omega_j)}{\sum_l p(\boldsymbol{x}_k|\omega_l, \Phi) P(\omega_l)} \equiv h_{kj} \tag{13}$$

If the component densities are taken to be d-variate Gaussian, $p(\boldsymbol{x}|\omega_j, \Phi) \sim \mathcal{N}_d(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$:

$$p(\boldsymbol{x}|\omega_j, \Phi) = \frac{1}{(2\pi)^{d/2} |\boldsymbol{\Sigma}_j|^{1/2}} \exp\left[-\frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_j) \right] \tag{14}$$

then we have ($g_j \equiv P(\omega_j)$)

$$h_{kj} = \frac{g_j |\boldsymbol{\Sigma}_j|^{-1/2} \exp\left[-(1/2)(\boldsymbol{x}_k - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1} (\boldsymbol{x}_k - \boldsymbol{\mu}_j) \right]}{\sum_l g_l |\boldsymbol{\Sigma}_l|^{-1/2} \exp\left[-(1/2)(\boldsymbol{x}_k - \boldsymbol{\mu}_l)^T \boldsymbol{\Sigma}_l^{-1} (\boldsymbol{x}_k - \boldsymbol{\mu}_l) \right]} \tag{15}$$

In the M-step, we update the component parameters Φ:

$$
\begin{aligned}
\boldsymbol{\mu}_j &= \frac{\sum_k h_{kj}\boldsymbol{x}_k}{\sum_k h_{kj}} && (16)\\
\boldsymbol{\Sigma}_j &= \frac{\sum_k h_{kj}(\boldsymbol{x}_k - \boldsymbol{\mu}_j)(\boldsymbol{x}_k - \boldsymbol{\mu}_j)^T}{\sum_k h_{kj}} && (17)\\
g_j &= \frac{1}{n}\sum_k h_{kj}
\end{aligned}
$$

When used for vector quantization, means $\boldsymbol{\mu}_j$ correspond to the reference vectors (or cluster centers or centroids) $\boldsymbol{v}_j$. h_{kj} is the posterior probability that $\boldsymbol{x}_k$ is generated by component j and as such denotes the soft association, like FCM's u_{kj} or SCS's p_{kj}.

With small samples, covariance matrices cannot be estimated with high accuracy; assuming diagonal or shared covariance matrices, we pool data. This is a form of regularization. In the case of a shared covariance matrix, the number of parameters reduce from $c \times d \times (d+1)/2$ to $d \times (d+1)/2$ which can be estimated during the M-step as

$$
\boldsymbol{\Sigma} = \sum_j g_j \boldsymbol{\Sigma}_j \tag{18}
$$

This uses the Mahalanobis distance, like in the FCM Eq. (5) with $A \equiv \boldsymbol{\Sigma}$. One can further regularize by assuming diagonal covariance matrices and reduce the number of parameters to d. In the extreme case of this where all diagonals are equal we have only one parameter and with equal priors, in the E-step we have

$$
h_{kj} = \frac{\exp\left[-(1/2\sigma^2)\|\boldsymbol{x}_k - \boldsymbol{\mu}_j\|^2\right]}{\sum_l \exp\left[-(1/2\sigma^2)\|\boldsymbol{x}_k - \boldsymbol{\mu}_l\|^2\right]} \tag{19}
$$

which is SCS's p_{kj} for $\beta \equiv 1/2\sigma^2$, as also noted by Bezdek & Pal (1995). In the case of Gaussian Mixtures (GM), this variance term can be estimated to maximize the likelihood during the M-step as

$$
\sigma^2 = \sum_j g_j \left[\frac{\frac{1}{d}\sum_k h_{kj}\|\boldsymbol{x}_k - \boldsymbol{\mu}_j\|^2}{\sum_k h_{kj}}\right] \tag{20}
$$

If h_{kj} is hardened to the extreme values of zero/one by taking a maximum, h_{kj} reduces to b_{kj} of hard c-means algorithm, as noted by Duda & Hart (1973, p. 201).

An online version to update the cluster centers can also be defined:

$$
\Delta\boldsymbol{\mu}_j = \eta h_{kj}\frac{(\boldsymbol{x}_k - \boldsymbol{\mu}_j)}{\sigma^2} \tag{21}
$$

which implements gradient-ascent for the maximum likelihood estimate:

$$
\max_{\boldsymbol{\mu}} \sum_k \log \sum_j g_j \exp\left[-\frac{1}{2\sigma^2}\|\boldsymbol{x}_k - \boldsymbol{\mu}_j\|^2\right] \tag{22}
$$

6 Numerical Experiments

Bezdek & Pal (1995) compare hard LVQ and soft SCS and ↓FLVQ on the IRIS dataset. Inputs are four dimensional real vectors and there are three classes. The projection of the dataset on the first two principal directions is given in Fig. 3. The aim is to use an unsupervised algorithm to find three cluster centers and then check whether these three centers converge to the class centers by classifying patterns to the class of the nearest center.

Table 1: Initial centers used in experiments.

Indices	Coordinates			
$\boldsymbol{v}_1$	5.006	3.428	1.462	0.246
$\boldsymbol{v}_2$	5.936	2.770	4.260	1.326
$\boldsymbol{v}_3$	6.588	2.974	5.552	2.026

Table 2: Centroids and Confusion Matrices when Euclidean distance is used. Results for Sample Mean, hard LVQ, SCS, and ↓FLVQ are taken from Bezdek & Pal (1995). Results using EM on Gaussian mixtures is added.

	Final centroids				Confusion matrix		
SM	5.006	3.418	1.464	0.244	50	0	0
	5.936	2.770	4.260	1.326	0	46	4
	6.588	2.974	5.552	2.026	0	7	43
LVQ	4.999	3.420	1.463	0.248	50	0	0
	5.873	2.746	4.366	1.414	0	47	3
	6.813	3.079	5.682	2.083	0	13	37
SCS	5.006	3.425	1.465	0.247	50	0	0
	5.884	2.743	4.370	1.414	0	47	3
	6.776	3.047	5.634	2.031	0	13	37
↓FLVQ	5.000	3.420	1.474	0.252	50	0	0
	5.884	2.748	4.371	1.411	0	47	3
	6.821	3.064	5.697	2.063	0	14	36
GM	5.006	3.418	1.464	0.244	50	0	0
$\sigma^2 = 0.134$	5.886	2.744	4.381	1.424	0	47	3
	6.828	3.065	5.697	2.056	0	14	36

Here we report results found by Bezdek & Pal using hard LVQ, SCS, and ↓FLVQ and compare them with the results we found using EM on Gaussian Mixtures. We have two sets of runs. In the first set, we do not take into account covariances and use Euclidean distance. In the second set, we take into account covariances and use the Mahalanobis distance. In each set, to form a base for comparison, we use the given class labels (instead of estimating them through the memberships) and compute the ideal sample means and the confusion matrix. Such results are denoted as SM.

Because all of the former use Euclidean distance, in the case of Gaussian mixtures too, in the first set of runs, we start by assuming diagonal covariance matrices, $\boldsymbol{\Sigma}_j = \sigma^2 I$, and estimate σ^2 from the sample (Eq. (20)). In all methods, centers are initialized as in Table 1, as used by Bezdek & Pal. Results are given in Table 2 where we see that result by GM is not significantly different from other approaches.

As seen in Fig. 3, the dimensions are highly correlated and variances are not equal so Euclidean distance is not ideal and the covariance information should be taken into account, extending Euclidean distance to Mahalanobis distance. One can have one covariance matrix shared by all clusters or have one separate for each cluster. For these two cases, we use supervised label information, estimate centers and covariance matrices and see that the substitution error decreases considerably (Table 3).

We then use EM to estimate the centers and covariances in an unsupervised manner; we do not use the given labels but estimate them as the posteriors h_{kj}. We assume equal priors. The centers are initialized as given in Table 1. In the case of the shared covariance matrix, it is initialized as the identity matrix

Table 3: Centroids and Confusion Matrices when covariance information is used. As in a supervised case, we use labels and compute the ideal values from the sample (denoted SM).

	Final centroids				Final covariances				Confusion matrix		
SM, $\boldsymbol{\Sigma}_j = \boldsymbol{\Sigma}$	5.006	3.418	1.464	0.244	0.265	0.093	0.167	0.039	50	0	0
	5.936	2.770	4.260	1.326	0.093	0.116	0.055	0.033	0	48	2
	6.588	2.974	5.552	2.026	0.167	0.055	0.185	0.043	0	1	49
					0.039	0.033	0.043	0.042			
SM, $\boldsymbol{\Sigma}_1$	5.006	3.418	1.464	0.244	0.124	0.100	0.016	0.011	50	0	0
					0.100	0.145	0.012	0.011			
					0.016	0.012	0.030	0.006			
					0.011	0.011	0.006	0.012			
$\boldsymbol{\Sigma}_2$	5.936	2.770	4.260	1.326	0.266	0.085	0.183	0.056	0	48	2
					0.085	0.099	0.083	0.041			
					0.183	0.083	0.221	0.073			
					0.056	0.041	0.073	0.039			
$\boldsymbol{\Sigma}_3$	6.588	2.974	5.552	2.026	0.404	0.094	0.303	0.049	0	1	49
					0.094	0.104	0.071	0.048			
					0.303	0.071	0.305	0.049			
					0.049	0.048	0.049	0.075			

and Eq. (18) is used in the M-step. In the case where different clusters have arbitrary different covariance matrices, they are again initialized as the identity matrix and are updated during the M-step as given in Eq. (17). Final means, covariance matrices, substitution error and the confusion matrices are given in Table 4. Log likelihood and errors are given as a function of EM iterations in Fig. 4. Note that with increasing number of parameters likelihood increases meaning a better fit to the training data which does not necessarily imply better separation of classes. This is because likelihood is an unsupervised measure whereas error is supervised and we optimize the former.

To test the robustness of GM to initial values, we run the three variants of Gaussian mixtures starting from ten randomly chosen centers on the IRIS dataset. The three variants are (i) diagonal shared covariance matrix $\boldsymbol{\Sigma}_j = \boldsymbol{\Sigma} = \sigma^2 I$, (ii) shared full covariance matrix $\boldsymbol{\Sigma}_j = \boldsymbol{\Sigma}$, and (iii) separate covariance matrices $\boldsymbol{\Sigma}_j$ for different clusters. Results are given in Table 5. In all three cases, starting from different initial centers, final likelihoods were nearly equal (stdev of 5E-4). The final confusion matrices were always the same implying that the initial state is not critical.

With only 50 samples per cluster, we cannot have an accurate estimation of a separate covariance matrix that has ten free parameters per cluster. For the case of a shared covariance matrix, we pool all 150 samples to estimate the shared ten parameters and this leads to better generalization. Although as we see in Fig. 3, Class 1 has different spread than Classes 2 and 3, we see in Table 3 that this assumption of common covariance is not harmful for this particular sample. In the case of a diagonal covariance matrix, we pool all 150 samples to estimate one parameter but though this has low variance, clearly it is very biased and is not accurate.

7 Supervised Training of Cluster Parameters

Vector quantization can be seen as a mapping from the original input space to a new space whose dimensions are the associations, h_j. Using a Gaussian membership function, we have:

Table 4: Centroids and Confusion Matrices when covariance information is used with EM on Gaussian mixtures. We do not use labels but estimate them during the E-step and in the M-step, we update class information using these soft labels. Hard LVQ, SCS, and $\downarrow$FLVQ do not use covariance information and are not included in this comparison.

	Final centroids				Final covariances				Confusion matrix		
GM, $\boldsymbol{\Sigma}_j = \boldsymbol{\Sigma}$	5.006	3.418	1.464	0.244	0.263	0.090	0.169	0.039	50	0	0
	5.942	2.761	4.260	1.320	0.090	0.112	0.051	0.031	0	48	2
	6.575	2.981	5.540	2.026	0.169	0.051	0.186	0.042	0	1	49
					0.039	0.031	0.042	0.040			
GM, $\boldsymbol{\Sigma}_1$	5.006	3.418	1.464	0.244	0.122	0.098	0.016	0.010	50	0	0
					0.098	0.142	0.011	0.011			
					0.016	0.011	0.030	0.006			
					0.010	0.011	0.006	0.011			
$\boldsymbol{\Sigma}_2$	5.917	2.779	4.208	1.299	0.275	0.096	0.186	0.055	0	45	5
					0.096	0.092	0.091	0.043			
					0.186	0.091	0.203	0.062			
					0.055	0.043	0.062	0.033			
$\boldsymbol{\Sigma}_3$	6.548	2.950	5.486	1.989	0.387	0.092	0.302	0.060	0	0	50
					0.092	0.111	0.084	0.056			
					0.302	0.084	0.324	0.072			
					0.060	0.056	0.072	0.084			

Table 5: Comparison of results using Gaussian Mixtures over the IRIS dataset. These are averages of ten independent runs with randomly initialized centers. In all three cases, standard deviations of errors and confusion matrices were zero.

	Parameters	Log Likelihood	Errors	Confusion matrix		
$\boldsymbol{\Sigma}_j = \boldsymbol{\Sigma} = \sigma^2 I$	1	-404.3	17.0	50.0	0.0	0.0
				0.0	47.0	3.0
				0.0	14.0	36.0
$\boldsymbol{\Sigma}_j = \boldsymbol{\Sigma}$	10	-256.3	3.0	50.0	0.0	0.0
				0.0	48.0	2.0
				0.0	1.0	49.0
$\boldsymbol{\Sigma}_j$	30	-181.5	5.0	50.0	0.0	0.0
				0.0	45.0	5.0
				0.0	0.0	50.0

$$h_j = \frac{\exp\left[-(1/2\sigma^2)\|\boldsymbol{x} - \boldsymbol{\mu}_j\|^2\right]}{\sum_l \exp\left[-(1/2\sigma^2)\|\boldsymbol{x} - \boldsymbol{\mu}_l\|^2\right]} \tag{23}$$

In most applications finding the associations h_j is an intermediate step before computing final output(s), e.g., for function approximation or classification. The output is generally taken as a linear weighted sum of these associations:

$$O_i = \sum_{j=1}^{c} W_{ij} h_j \tag{24}$$

To implement the required mapping, we need to find the $\boldsymbol{\mu}_j$, σ and $\boldsymbol{W}_{ij}$. We can use one of the techniques proposed before to find the $\boldsymbol{\mu}_j$ in an unsupervised manner, use a heuristic to find σ, e.g., half of the average inter-center distance, and then use a supervised method to find the $\boldsymbol{W}_{ij}$. This approach is known as radial-basis functions in the neural network literature (Moody & Darken, 1989). In the fuzzy-logic literature (Jou, 1993), this is a multi-input-multi-output system based on the Gaussian membership function, product-inference rule (the product of exponentials is equal to the exponential of a sum), a singleton fuzzifier (center $\boldsymbol{\mu}_j$ and spread parameter σ^2) and a center average defuzzifier.

A better idea seems to train also the cluster center and spread in a supervised way, or finetune them in a supervised way after they have been trained in an unsupervised way. In function approximation, using the mean square error as the error measure with linear output units:

$$\min_{\boldsymbol{\mu}, \sigma, \boldsymbol{W}} \sum_i (R_i - O_i)^2 \tag{25}$$

where R_i is the required output for the ith output unit. By gradient-descent, we get the following update rules:

$$\begin{aligned} \Delta W_{ij} &= \eta (R_i - O_i) h_j \\ \Delta \boldsymbol{\mu}_j &= \eta \left[\sum_i (R_i - O_i) h_j (W_{ij} - O_i) \right] \frac{(\boldsymbol{x} - \boldsymbol{\mu}_j)}{\sigma^2} \end{aligned} \tag{26}$$

$\Delta\sigma$ can also be computed in this way if required. The difference between Eq. (21) and (26) is that in the former case we only look at the input distribution whereas in the latter case we also take into account the supervised error at the output and the contribution of that cluster to the output. This latter case where the training of cluster centers, $\boldsymbol{\mu}_j$, and the cluster weights, W_{ij}, is coupled is clearly better than the uncoupled case which ignores the useful information of error on the outputs. Thus the coupled approach is generally to be preferred. This has also been shown previously for the case of local linear models (Alpaydın & Jordan, 1996).

Here we give equations to train a supervised mapping using gradient-based learning. It is also possible to extend the probabilistic model of Gaussian mixtures for supervised learning and use EM to estimate the parameters. The mixture components then correspond to experts (Jordan & Xu, 1995) or latent, or hidden, causes (Bishop et al., 1997) which, acting in combination, give rise to the apparent complexity of the observed dataset.

8 Conclusions

The structural information due to a data point is carried by all of the c distances to the cluster centers. This distance depends both on the center positions *and* the spread of data around this center. Using Euclidean distance assumes that the data is spread symmetrically radially around the center. The spread parameters m in FCM (and FLVQ) and β in SCS cause a symmetric effect around a cluster center. Furthermore these parameters cannot be estimated from the data but should be set by the designer. When this assumption does not hold, the covariance of the data should be estimated and taken into account. Using Gaussian mixtures and the EM algorithm allows this in that one can estimate the centers *and* the covariances that maximize the likelihood of the given sample.

Like in any maximum likelihood scheme, having too many free parameters with insufficient data may lead to overfitting. One can then use a regularization approach and limit the number of parameters. One way to do this in the case of Gaussian mixtures is to use one covariance matrix shared by all clusters. One can further regularize by assuming all inputs to be uncorrelated which implies a covariance matrix with zero off-diagonals.

A similar approach is to assume a Bayesian prior for the parameters to effectively reduce the variance due to the data sample and then use EM to obtain a Maximum A Posteriori (MAP) estimate. This is especially useful if good priors are known and/or the data sample is small. Another approach to decrease variance is to average the outputs of ensembles of Gaussian mixture density estimators trained on bootstrap replicates of the dataset (Ormoneit & Tresp, 1996).

The Gaussian mixture model can be extended and EM can be used to generate a topographic map (Bishop et al., 1997) or a supervised mapping (Jordan & Xu, 1995).

We also remark that vector quantization is generally an intermediate step before finding a final output for which supervision is possible. Thus instead of an uncoupled approach where we use an unsupervised method to find the cluster parameters and use these clusters to learn the mapping in a supervised manner, we advocate a coupled approach where the cluster parameters are also trained (or finetuned) in a supervised way, coupled to the learning of the mappings. The uncoupled approach ignores the error at the outputs which may lead to worse results.

Appendix: EM and Gaussian Mixtures

Here we give a derivation of the EM algorithm as applied to mixtures of Gaussians; for an alternative derivation, see Duda & Hart (1973), pp. 192–200. A comparison of the EM algorithm and other algorithms for the learning of Gaussian mixture models is given by Xu & Jordan (1996).

The Expectation-Maximization (EM) algorithm (Dempster et al., 1977; Redner & Walker, 1984) is used in maximum likelihood estimation where the problem involves two sets of random variables of which one, X, is observable and the other, Z, is hidden. The goal of the algorithm is to find the parameter vector Φ that maximizes the likelihood of the observed values of X, $\mathcal{L}(\Phi|\mathcal{X})$. But in cases where this is not feasible, we associate the extra *hidden* variables Z and express the underlying model using both to maximize the likelihood of the joint distribution of X and Z, the *complete* likelihood $\mathcal{L}_c(\Phi|\mathcal{X},\mathcal{Z})$.

Since the Z values are not observed, we cannot work directly with the complete data likelihood $\mathcal{L}_c$, instead we work with its expectation, $\mathcal{Q}$, given the current parameter values Φ^t. This is the expectation (E) step of the algorithm. Then in the maximization (M) step, we look for the new parameter values, Φ^{t+1}, that maximize this. Thus:

$$\begin{aligned}
\text{E-step} &: \quad \mathcal{Q}(\Phi|\Phi^t) = E[\mathcal{L}_c(\Phi|\mathcal{X},\mathcal{Z})|\mathcal{X},\Phi^t)] \\
\text{M-step} &: \quad \Phi^{t+1} = \arg\max_{\Phi} \mathcal{Q}(\Phi|\Phi^t)
\end{aligned}$$

Dempster et al. proved that an increase in $\mathcal{Q}$ implies an increase in the incomplete likelihood:

$$\mathcal{L}(\Phi^{t+1}|\mathcal{X}) \geq \mathcal{L}(\Phi^t|\mathcal{X})$$

In the case of mixtures, the hidden variables are the sources of observations, i.e., which observation belongs to which cluster. If these had been given (for example as class labels in a supervised setting), we would have known which parameters to adjust to fit that data point. The EM algorithm works as follows: During the E-step we estimate these labels given our current knowledge of classes and in the M-step, we update our class knowledge given the "soft labels" determined in the E-step. It is these soft labels that correspond to the soft associations used in vector quantization.

We define a vector of *indicator variables* $\boldsymbol{z}_k = \{z_{k1}, \ldots, z_{kc}\}$ where $z_{kj} = 1$ if $\boldsymbol{x}_k$ belongs to cluster ω_j, and zero otherwise. $\boldsymbol{z}$ is a multinomial distribution from c categories with prior probabilities $P(\omega_j) \equiv g_j$.Then

$$P(\boldsymbol{z}_k) = \prod_{j=1}^{c} g_j^{z_{kj}} \tag{27}$$

The likelihood of an observation $\boldsymbol{x}_k$ is equal to its probability specified by the component that generated it:

$$p(\boldsymbol{x}_k|\boldsymbol{z}_k) = \prod_{j=1}^{c} p_j(\boldsymbol{x}_k)^{z_{kj}} \tag{28}$$

$p_j(\boldsymbol{x}_k)$ is the probability that $\boldsymbol{x}_k$ is generated by component ω_j, also denoted as $p(\boldsymbol{x}_k|\omega_j)$. The joint density is

$$p(\boldsymbol{x}_k, \boldsymbol{z}_k) = P(\boldsymbol{z}_k)p(\boldsymbol{x}_k|\boldsymbol{z}_k)$$

and the complete data likelihood of the iid sample $\mathcal{X}$ is

$$\begin{aligned}
\mathcal{L}_c(\Phi|\mathcal{X}, \mathcal{Z}) &= \log \prod_{k=1}^{n} p(\boldsymbol{x}_k, \boldsymbol{z}_k|\Phi) \\
&= \sum_k \log p(\boldsymbol{x}_k, \boldsymbol{z}_k|\Phi) \\
&= \sum_k \log P(\boldsymbol{z}_k|\Phi) + \log p(\boldsymbol{x}_k|\boldsymbol{z}_k, \Phi) \\
&= \sum_k \sum_j z_{kj}[\log g_j + \log p_j(\boldsymbol{x}_k|\Phi)]
\end{aligned} \tag{29}$$

E-step: We define

$$\begin{aligned}
\mathcal{Q}(\Phi|\Phi^t) &\equiv E\left[\log P(X, Z)|\mathcal{X}, \Phi^t\right] \\
&= E\left[\mathcal{L}_c(\Phi|\mathcal{X}, \mathcal{Z})|\mathcal{X}, \Phi^t)\right] \\
&= \sum_k \sum_j E[z_{kj}|\mathcal{X}, \Phi^t][\log g_j + \log p_j(\boldsymbol{x}_k|\Phi^t)]
\end{aligned}$$

Note that

$$\begin{aligned}
E[z_{kj}|\mathcal{X}, \Phi^t] &= E[z_{kj}|\boldsymbol{x}_k, \Phi^t] \\
&= P(z_{kj} = 1|\boldsymbol{x}_k, \Phi^t) \\
&= \frac{p(\boldsymbol{x}_k|z_{kj} = 1, \Phi^t)P(z_{kj} = 1|\Phi^t)}{p(\boldsymbol{x}_k|\Phi^t)} \\
&= \frac{p_j(\boldsymbol{x}_k|\Phi^t)g_j}{\sum_l p_l(\boldsymbol{x}_k|\Phi^t)g_l} \\
&= \frac{p(\boldsymbol{x}_k|\omega_j, \Phi^t)P(\omega_j)}{\sum_l p(\boldsymbol{x}_k|\omega_l, \Phi^t)P(\omega_l)} \\
&= P(\omega_j|\boldsymbol{x}_k, \Phi^t) \equiv h_{kj}
\end{aligned} \tag{30}$$

Thus the expected value of the hidden variable, $E[z_{kj}]$, is the posterior probability that $\boldsymbol{x}_k$ is generated by component ω_j.

M-step: We maximize $\mathcal{Q}$ to get the next set of parameter values Φ^{t+1}:

$$\Phi^{t+1} = \arg\max_{\Phi} \left[\mathcal{Q}(\Phi|\Phi^t)\right]$$

$$\begin{aligned}\mathcal{Q}(\Phi|\Phi^t) &= \sum_k \sum_j h_{kj}[\log g_j + \log p_j(\boldsymbol{x}_k|\Phi^t)] \\ &= \sum_k \sum_j h_{kj} \log g_j + \sum_k \sum_j h_{kj} \log p_j(\boldsymbol{x}_k|\Phi^t)\end{aligned} \tag{31}$$

As is generally done, we can assume $g_j = 1/c$ to be constant and not estimate them. Note that the first term is independent of the components and thus can be dropped while estimating the parameters of the components. We solve for

$$\nabla_{\Phi} \sum_k \sum_j h_{kj} \log p_j(\boldsymbol{x}_k|\Phi) = 0 \tag{32}$$

If we assume Gaussian components, $p_j(\boldsymbol{x}_k|\Phi) \sim \mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$, we get:

$$\begin{aligned}\boldsymbol{\mu}_j^{t+1} &= \frac{\sum_k h_{kj}\boldsymbol{x}_k}{\sum_k h_{kj}} \\ \boldsymbol{\Sigma}_j^{t+1} &= \frac{\sum_k h_{kj}(\boldsymbol{x}_k - \boldsymbol{\mu}_j^{t+1})(\boldsymbol{x}_k - \boldsymbol{\mu}_j^{t+1})^T}{\sum_k h_{kj}}\end{aligned} \tag{33}$$

When $p_j(\boldsymbol{x}_k|\Phi) \sim \mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma})$, the case of a shared covariance matrix, Eq. (32) reduces to

$$\min_{\boldsymbol{\mu}} \sum_k \sum_j h_{kj}(\boldsymbol{x}_k - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_k - \boldsymbol{\mu}_j) \tag{34}$$

which is equivalent to Eq. (7) of fuzzy vector quantization with h_{kj} replacing $(u_{kj})^m$. When $p_j(\boldsymbol{x}_k|\Phi) \sim \mathcal{N}(\boldsymbol{\mu}_j, \sigma^2 I)$, the case of a shared diagonal matrix, we have

$$\min_{\boldsymbol{\mu}} \sum_k \sum_j h_{kj} \|\boldsymbol{x}_k - \boldsymbol{\mu}_j\|^2 \tag{35}$$

which is Eq. (4) of hard vector quantization with h_{kj} replacing b_{kj}.

The model also allows estimating the g_j which corresponds to including g_j into the parameter vector Φ. We note that the second term of Eq. (31) is independent of g_j and thus can be dropped. Then using the constraint that $\sum_j g_j = 1$ as the Lagrangian, we solve for:

$$\nabla_{g_j} \sum_k \sum_j h_{kj} \log g_j - \lambda \left(\sum_j g_j - 1\right) = 0 \tag{36}$$

and get

$$g_j = \frac{1}{n} \sum_k h_{kj} \tag{37}$$

References

[1] Alpaydın, E., & Jordan, M. I. (1996). Local linear perceptrons for classification, *IEEE Transactions on Neural Networks*, **7**, 788–792.

[2] Bezdek, J. C., & Pal, N. R. (1995). Two soft relatives of learning vector quantization, *Neural Networks*, **8**, 729–743.

[3] Bishop, C. M., Svensén, M., & Williams, C. K. I. (1997). GTM: A principled alternative to the self-organizing map. In M. C. Mozer, M. I. Jordan, & T. Petsche (Eds.), *Advances in neural information processing systems* **9**, Cambridge, MA: MIT Press.

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

[5] Duda, R. O., & Hart, P. E. (1973). Pattern classification and scene analysis, New York: Wiley.

[6] Jordan, M. I., & Xu, L. (1995). Convergence results for the EM approach to mixtures of experts architectures, *Neural Networks*, **8**, 1409–1431.

[7] Jou, C.-C. (1993). Comparing learning performance of neural networks and fuzzy systems. In *Proceedings of the IEEE international conference on neural networks*, (Vol. 2, pp. 1028–1033).

[8] Kohonen, T. (1995). Self-organizing maps, Berlin: Springer.

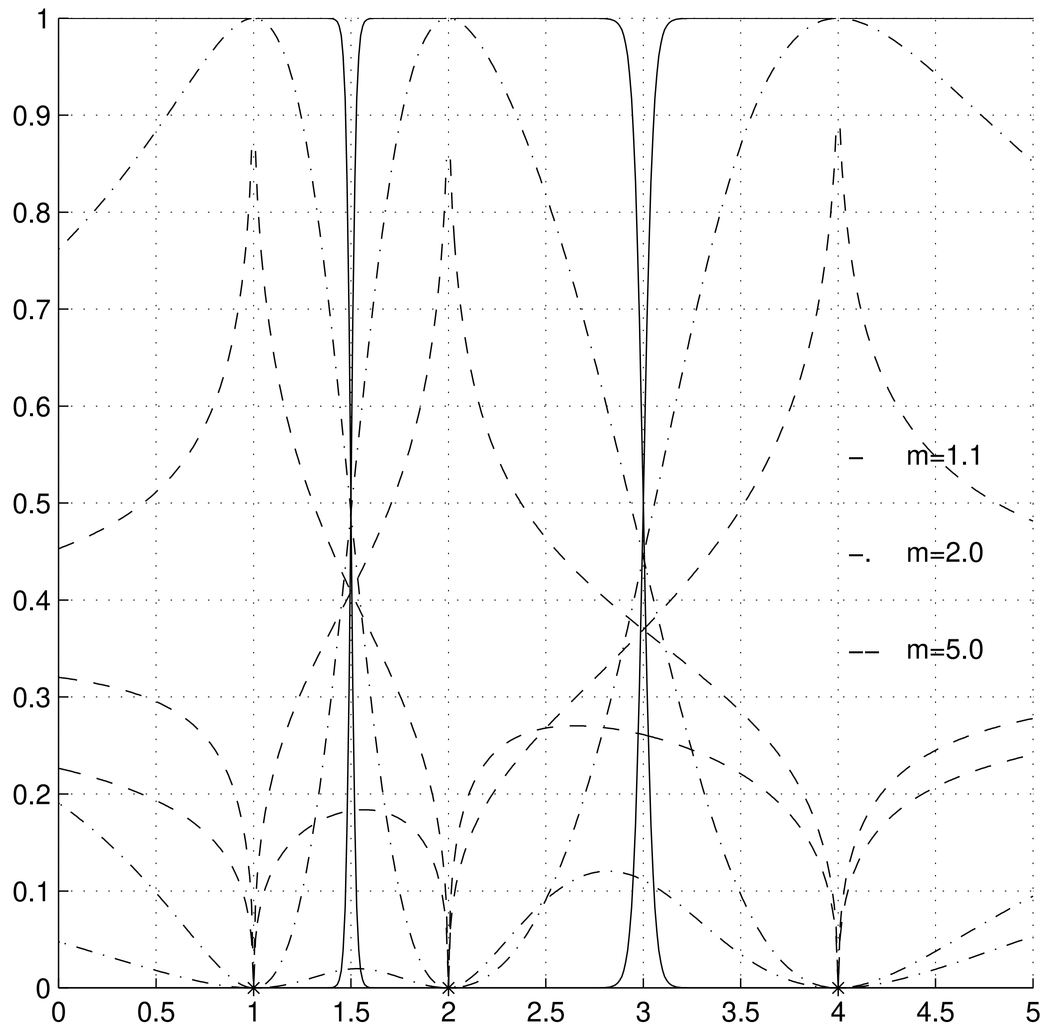
[9] Moody, J. & Darken, C. (1989). Fast learning in networks of locally tuned processing units, *Neural Computation*, **1**, 281–294.

[10] Ormoneit, D., & Tresp, V. (1996). Improved Gaussian mixture density estimates using Bayesian penalty terms and network averaging. In D. S. Touretzky, M. C. Mozer, & M. E. Hasselmo (Eds.), *Advances in neural information processing systems* **8**, Cambridge, MA: MIT Press.

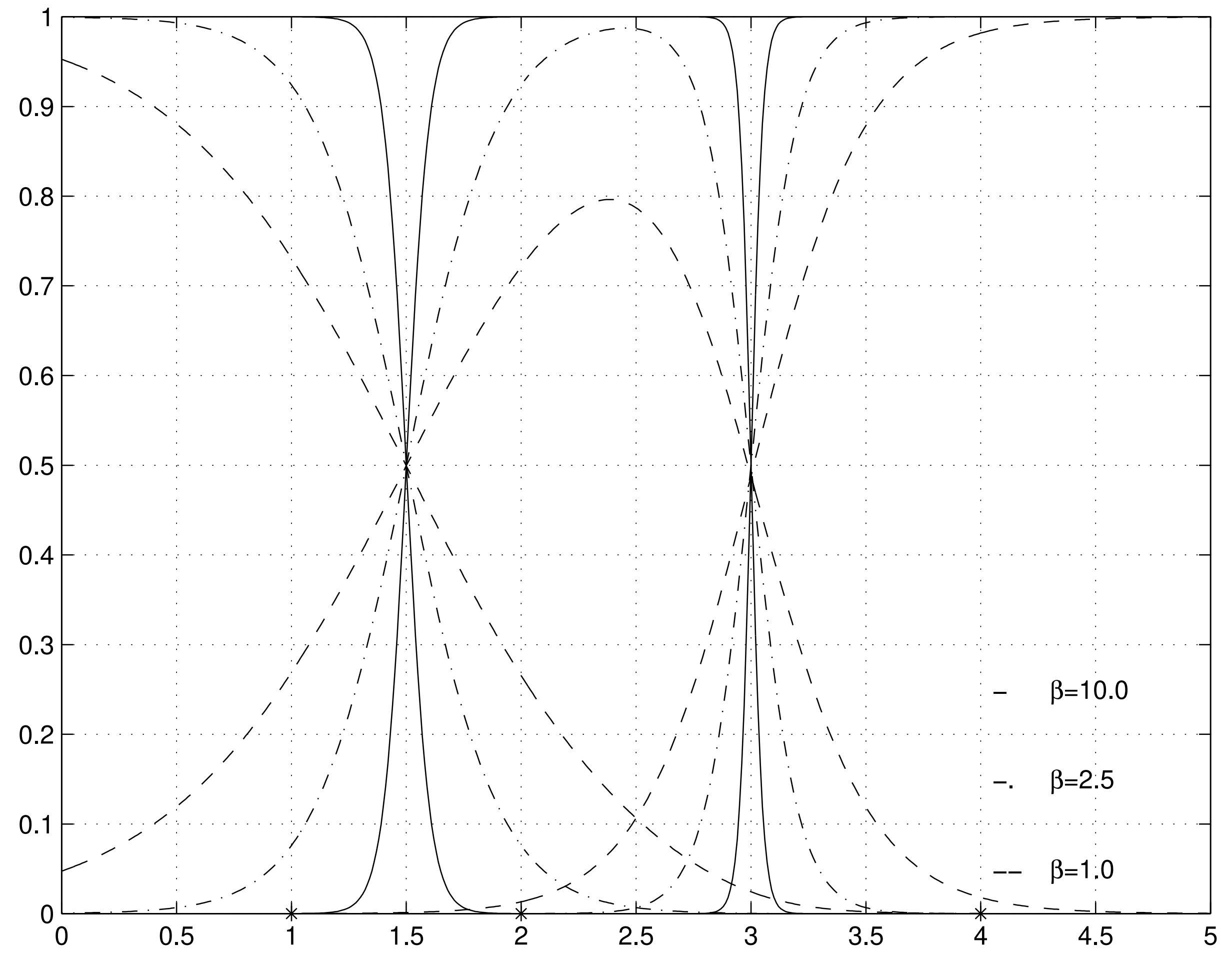
[11] Redner, R. A., & Walker, H. F. (1984). Mixture densities, maximum likelihood and the EM algorithm, *SIAM Review*, **26**, 195–239.

[12] Xu, L., & Jordan, M. I. (1996). On convergence properties of the EM algorithm for Gaussian mixtures, *Neural Computation*, **8**, 129–151.

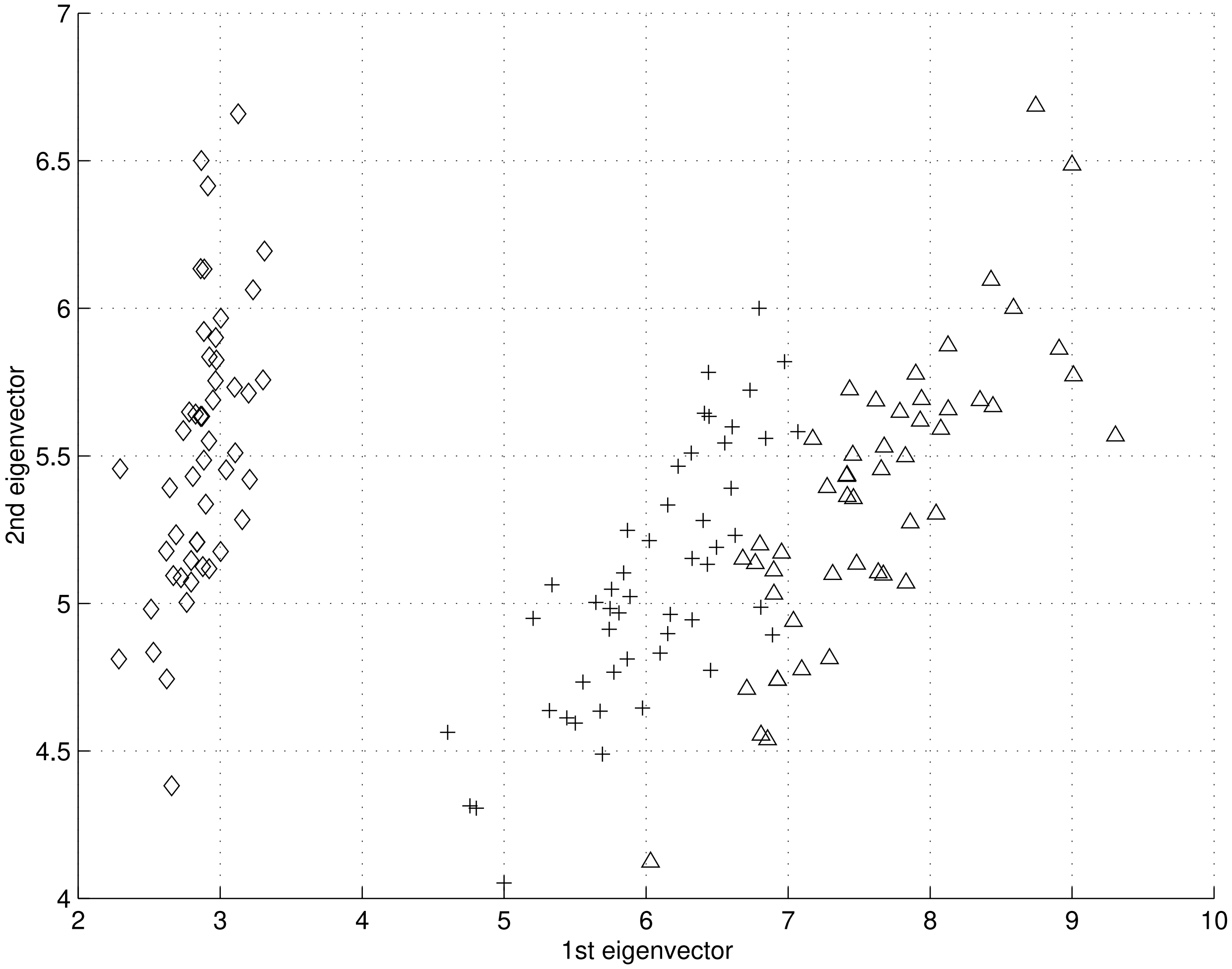
[13] Yair, E., Zeger, K., & Gersho, A. (1992). Competitive learning and soft competition for vector quantizer design, *IEEE Transactions on Signal Processing*, **40**, 294–309.



1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
m=1.1
m=2.0
m=5.0



1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
β=10.0
β=2.5
β=1.0



7
6.5
6
5.5
5
4.5
4
2
3
4
5
6
7
8
9
10
2nd eigenvector
1st eigenvector

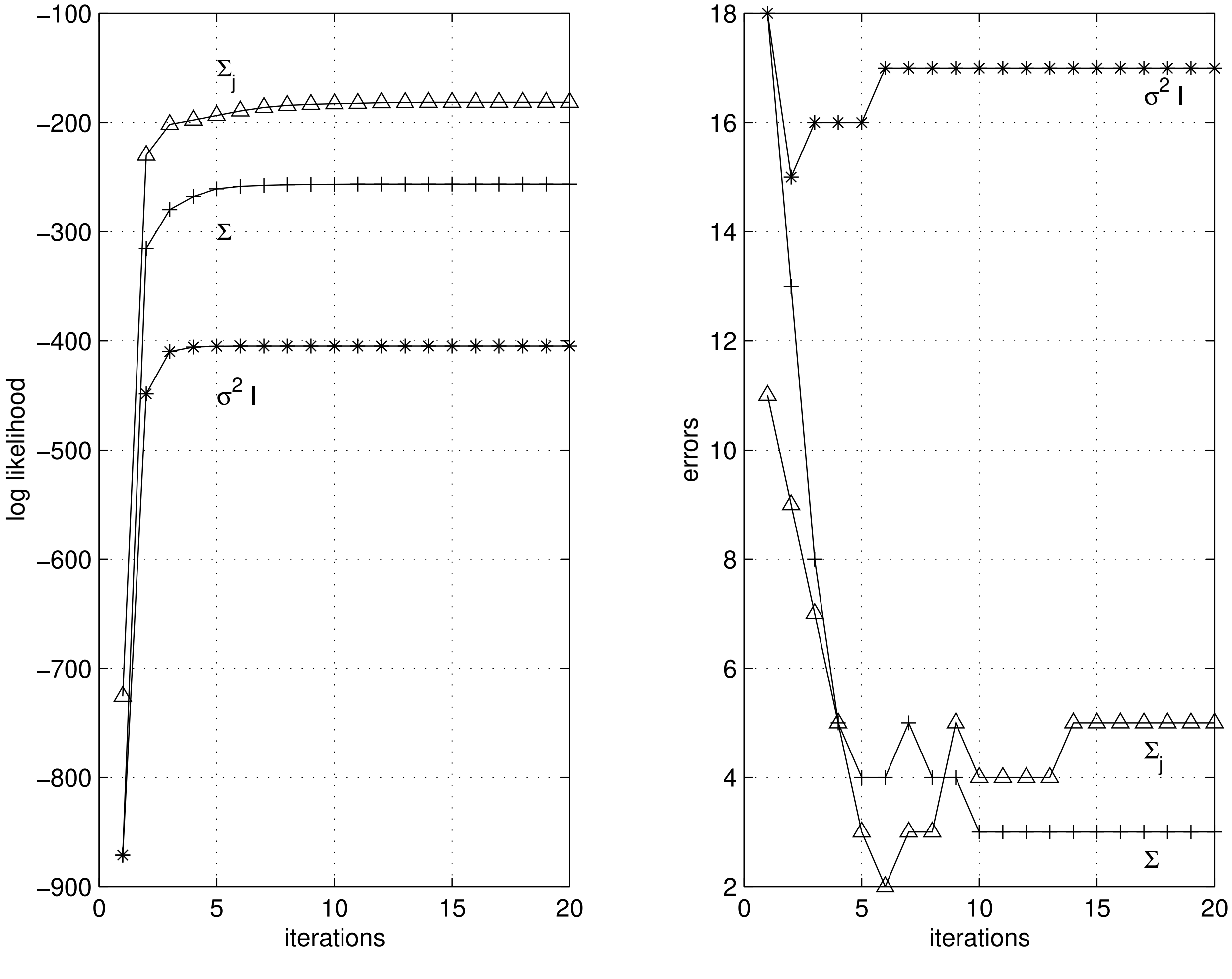

−100
−200
−300
−400
−500
−600
−700
−800
−900
log likelihood
0
5
10
15
20
iterations
Σ_j
Σ
$\sigma^2 I$
18
16
14
12
10
8
6
4
2
errors
iterations
$\sigma^2 I$
Σ_j
Σ

Figure 1: The fuzzy memberships u_{kj} are drawn for a one dimensional problem with three clusters at 1.0, 2.0, and 4.0. Smaller values of m lead to harder partitioning of the input space.

Figure 2: The probabilities p_{kj} computed by soft competition scheme (SCS) are drawn for a one dimensional problem with three clusters at 1.0, 2.0, and 4.0. Larger values of β imply smaller variance and lead to harder partitioning of the input space.

Figure 3: IRIS dataset projected on the two principal components. These two directions explain 98% of the variance.

Figure 4: Log likelihood and errors vs EM iterations for the three cases of (1) Shared diagonal covariance matrix, $\sigma^2 I$, marked as '$*$' (2) Shared covariance matrix, Σ, marked as '$+$' and (3) Separate covariance matrices, Σ_j, marked as '$\triangle$'. Higher likelihood does not necessarily imply lower error. The former is an unsupervised measure whereas the latter is supervised. Σ_j overfits and $\sigma^2 I$ underfits the data.