Greedy Gaussian Mixture Learning for Texture Segmentation

J.J. Verbeek, N. Vlassis and B. Kröse

Computer Science Institute, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands

Abstract. The problem of segmenting an image into several modalities representing different textures can be modeled using Gaussian mixtures. Fitting a Gaussian mixture on the data is not trivial problem and no guaranteed optimal algorithm exists. In this paper we show the benefits of a recently developed greedy procedure to Gaussian mixture learning to the problem of texture segmentation. We present the greedy learning method and provide experimental results illustrating the potential of the new method.

1 Introduction

One of the approaches to texture segmentation is the use of subspace methods in a high dimensional space spanned by the pixel intensities of patches of the image. The idea is that patches originating from the same texture are characterized by a high degree of redundancy: they are merely slightly transformed (scaled, rotated, translated, etc.) copies of the same pattern. The fact that only a limited number of degrees of freedom is present in all the patches of a texture suggests that all the patches (plus noise) live in some low dimensional manifold in the d-dimensional feature space. This motivated the use of subspace methods to model the textures. In the case of patches originating from several textures, mixtures of linear subspace models (we will use q to denote their dimensionality) are used to model the different sources [10, 4, 9].

A direct connection between modeling with mixtures of subspaces and Gaussian mixture modeling is provided by the mixtures of probabilistic principal component analyzers (MPPCA) [13]. This probabilistic formulation of subspace decomposition allows for a natural and straightforward definition of the segmentation problem in probabilistic terms. It may be noted that the MPPCA model is just a restriction of the class of all Gaussian mixtures to a subset characterized by component covariance matrices of the form $\mathbf{C} = \sigma^2\mathbf{I} + \mathbf{W}\mathbf{W}^\top$, where $\mathbf{W}$ is a $d \times q$ matrix. One may question why one would use the restricted class instead of the full class. One of the reasons is the computational cost which is significantly reduced if the dimensionality q of the subspaces is much smaller than d. In this work we consider non-restricted Gaussian mixtures ($d = q$), because of the difficult segmentation task.

Unfortunately, there exists no efficient algorithm to find, given a finite data set, the mixture that maximizes log-likelihood among all mixtures (provided

that we are looking for a mixture with relatively few mixture components as compared to the number of data points). The most popular method for learning Gaussian mixtures is the Expectation-Maximization (EM) algorithm [3]. The EM algorithm produces, given an initial mixture, a sequence of mixtures with guaranteed non-decreasing and converging log-likelihood on the data at hand. The major drawback of the EM algorithm is its dependency on the initial mixture. In case of 'unlucky' initialization the EM algorithm might converge to a poor, only locally optimal, solution. This deficiency is usually alleviated by starting the EM algorithm for several randomly (or heuristically) chosen mixtures and then selecting the mixture maximizing log-likelihood.

Recently we developed a new greedy algorithm to learn Gaussian mixtures [14]. The new algorithm has several important benefits over the standard EM approach to Gaussian mixture learning: (i) the new method is deterministic (ii) the new method generates a sequence of mixtures with an increasing number of mixture components which can be used conveniently to select an appropriate number of mixture components, and (iii) experimentally the new method is found to perform significantly better. In this paper we discuss how this new method may increase performance of texture segmentation based on Gaussian mixtures.

The paper is organized as follows: In Section 2 we discuss how Gaussian mixtures can be used to perform texture segmentation. Then in Section 3 we describe the greedy approach to Gaussian mixture learning presented in [14]. In Section 4 we describe an experiment conducted to evaluate the benefit for the texture segmentation task when using the new method instead of standard EM. Section 5 ends the paper with conclusions and a discussion.

2 Gaussian Mixtures for Texture Segmentation

The texture segmentation task can be seen as a traditional clustering or unsupervised classification problem. Given a set of patches extracted from an image, the task is two-fold: (i) find a (parametric) description of the different modes or clusters present in the data and (ii) classify the data using the cluster descriptions. In this paper we focus on the first part, where we use the class of Gaussian mixtures as representations for the clusters. The choice for non-restricted Gaussian mixtures was motivated by experimental results presented in [9]. There, texture segmentation experiments are presented where the performance of the MPPCA models, unrestricted Gaussian mixture models, and mixtures of independent component analysers are compared. It is shown there that the unrestricted Gaussian mixture model performs better on some difficult segmentation tasks than the MPPCA model.

A Gaussian mixture is defined as a convex combination of Gaussian densities. A Gaussian density in a d-dimensional space, parameterized by its mean $\mathbf{m} \in \mathbb{R}^d$ and $d \times d$ covariance matrix $\mathbf{C}$ is defined by the density:

$$\phi(\mathbf{x};\theta) = (2\pi)^{-d/2} \det(\mathbf{C})^{-1/2} \exp\left(-(\mathbf{x}-\mathbf{m})^\top \mathbf{C}^{-1}(\mathbf{x}-\mathbf{m})/2\right), \qquad (1)$$

where θ denotes the parameters $\mathbf{m}$ and $\mathbf{C}$. A mixture of k Gaussians is then defined as:

$$f_k(\mathbf{x}) = \sum_{i=1}^{k} \pi_i \phi(\mathbf{x}; \theta_i), \quad \text{with} \quad \sum_{j=1}^{k} \pi_j = 1 \quad \text{and for} \quad j \in \{1, \ldots, k\} : \pi_j \geq 0. \tag{2}$$

The π_i are called the mixing weights and $\phi(\mathbf{x}; \theta_i)$ the components of the mixture. For the classification we will use a probabilistic maximum posterior classifier, as to minimize the risk of misclassification [11]. The classification is implemented by assigning a data point $\mathbf{x}$ to class i^*, where $i^* = \arg\max_i \{p(i \mid \mathbf{x})\}$ and $p(i \mid \mathbf{x}) = \pi_i \phi(\mathbf{x}; \theta_i) / f_k(\mathbf{x})$.

Evaluation of Segmentations In our experiments we use supervised data generated from k different textures, i.e. each patch is accompanied with a label of the texture it originated from. We cluster this data in k clusters on the basis of a k component Gaussian mixture. The goal is to find a clustering that segments the image well, which can be formulated as finding a clustering that allows concise prediction of the (true) texture label on the basis of the model-assigned cluster label.

To evaluate a given clustering we consider the $k \times k$ confusion matrix A, where A_{ij} denotes how many patches of texture j are assigned to cluster i. Let B be a random variable which ranges over the k textures and C a random variable ranging over the k clusters. As a measure of how informative a given clustering is we compute the conditional entropy $H(B \mid C)$. Note that this quantity is directly related to the mutual information between the two variables $I(B; C) = H(B) - H(B \mid C)$. Since $H(B)$ is fixed for a given data set, we have that the lower $H(B \mid C)$ is, the higher the mutual information is and thus the better the clustering is. In the above the entropy of a discrete random variable X is a measure of uncertainty in X and is defined as:

$$H(X) = \sum_{x} p(X = x) \log p(X = x) \tag{3}$$

and the conditional entropy is defined as:

$$H(X \mid Y) = E_Y H(X \mid Y = y) = \sum_{x,y} p(Y = y, X = x) \log p(X = x \mid Y = y). \tag{4}$$

If we set the logarithm to base 2, the (conditional) entropy measures how many bits are needed on average per outcome to encode a string of outcomes of the variable if we use an optimal code scheme. See [2] for an excellent discussion of these concepts.

The labeled patches in each experiment provide joint realizations of the variables B and C resulting in the matrix A. The matrix A provides empirical estimates of the probabilities needed to compute the conditional entropy:

$$p(B = b \mid C = c) \approx A_{c,b} / \sum_{b'} A_{c,b'}, \tag{5}$$

and

$$p(C = c) \approx \sum_{b'} A_{c,b'} / \sum_{b',c'} A_{c',b'}. \tag{6}$$

3 Learning Gaussian Mixtures

In most applications where Gaussian mixtures are used to model a data set $\mathbf{X}_n = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, the parameters of the model are found by means of the EM algorithm [3]. The EM algorithm is conveniently implemented by iterative application of the equations listed below for all components $j \in \{1, \ldots, k\}$:

$$P(j \mid \mathbf{x}_i) := \frac{\pi_j \phi(\mathbf{x}_i; \theta_j)}{f_k(\mathbf{x}_i)}, \tag{7}$$

$$\pi_j := \frac{1}{n} \sum_{i=1}^{n} P(j \mid \mathbf{x}_i), \tag{8}$$

$$\mathbf{m}_j := \frac{\sum_{i=1}^{n} P(j \mid \mathbf{x}_i) \mathbf{x}_i}{n\pi_j}, \tag{9}$$

$$\mathbf{C}_j := \frac{\sum_{i=1}^{n} P(j \mid \mathbf{x}_i)(\mathbf{x}_i - \mathbf{m}_j)(\mathbf{x}_i - \mathbf{m}_j)^\top}{n\pi_j}. \tag{10}$$

In [14] we presented a new deterministic greedy method to learn Gaussian mixtures. The method builds the k component mixture component-wise, by interleaving updating a mixture until convergence and then inserting a new component. The method shows resemblance to the Vertex Direction Method [8] and is motivated by a recent approximation result by Li [6]. This approximation result states that for an *arbitrary* probability density function f there exists a sequence $\{f_k\}$ of finite mixtures such that $f_k(\mathbf{x}) = \sum_{i=1}^{k} \pi_i \phi(\mathbf{x}; \theta_i)$ achieves Kullback-Leibler (KL) divergence[1] $D(f \parallel f_k) \leq D(f \parallel g_P) + c/k$ for every $g_P = \int \phi(\mathbf{x}; \theta) P(d\theta)$. Hence, the difference in KL divergence achievable by k-component mixtures and the KL divergence achievable by any (possibly non-finite) mixture from the same family of components tends to zero with speed c/k (where c is a constant not dependent on k but only on the component family). Furthermore, this bound holds for mixtures obtained by the following greedy procedure:

1. Let f_1 be the maximum likelihood Gaussian for the data. Set $k := 1$.
2. Perform a search to find the optimal new component $\phi(\mathbf{x}; \theta^*)$ and corresponding mixing weight α^*, where

$$\{\theta^*, \alpha^*\} = \arg\max_{\{\theta, \alpha\}} \sum_{i=1}^{n} \log\left[(1 - \alpha) f_k(\mathbf{x}_i) + \alpha \phi(\mathbf{x}_i; \theta)\right] \tag{11}$$

   with f_k fixed.

[1] The KL divergence is defined as: $D(f \parallel g) = \int_\Omega f(x) \log\left(f(x)/g(x)\right) \mathrm{d}x$ where Ω is the domain of the densities f and g, see [2] for details.

3. Set $f_{k+1}(\mathbf{x}) := (1-\alpha^*)f_k(\mathbf{x}) + \alpha^*\phi(\mathbf{x};\theta^*)$ and $k := k+1$;
4. Update f_k using EM until convergence.
5. If a stopping criterion is met then quit, else go to step 2.

This greedy scheme replaces the problem of finding a good initialization for EM with a sequence of component allocation problems. In general, the optimal component allocation step 2 can not be implemented to find the global maximum θ and α. Therefore search heuristics have to be employed in order to find near optimal solutions in a reasonable amount of time. In the rest of this section we describe the implementation of the component allocation step as proposed in [14] which is a refinement of the method in [15].

Efficient Component Allocation. Since no algorithm is known to implement step 2 above, we limit ourselves to consider only a finite number of 'candidate' components and pick the best among those. In [15] a candidate component is located at every data point, however this results in a run time of $O(n^2)$, which is prohibitive for reasonable n.

Our new method makes use of *kd*-trees [1], which were originally designed to speed-up the execution of nearest neighbor queries and related problems like computing all points in an ϵ neighborhood of a given query point. A kd-tree defines a recursive binary partitioning of a k-dimensional data set, where the root node contains all data. Sproull [12] proposed to partition the data at each node by cutting with a hyper-plane perpendicular to the direction with largest variance of the data present in that node, i.e. perpendicular to the Principal Component [5]. One may view the resulting procedure as a 'nested' Principal Component Analysis. If we fully expand the tree, the leafs of the tree are given by the individual data points. We can regard each node in the tree as a bucket containing a portion of the data. Every layer of the tree gives a partitioning of the data. It turns out that these partitions provide quite reasonable *clusterings* of the data in terms of mean squared distance from the data to their closest cluster center (the cluster centers are given by the bucket means), see [7].

Now, let us return to mixtures. A mixture f_k describes the data at some scale, we may expect the mixture components in the mixture f_{k+1} to be 'smaller' (i.e. the determinant of their covariance matrix is likely to be smaller) since the same structure in the data can now be modeled by more components and hence in more detail. Our component allocation procedure exploits this idea by generating for each existing mixture component i 6 candidates based on a subset A_i of the data, where $A_i = \{\mathbf{x} \in \mathbf{X}_n : P(i \mid \mathbf{x}) = \max_j\{P(j \mid \mathbf{x})\}\}$ with $1 \leq j \leq k$. The posteriors $P(i \mid \mathbf{x})$ are available directly since we already computed them for the EM updates of the k-component mixture, see equation (7). For each set A_i we construct the first 6 nodes, residing in the first two layers, of the kd-tree T_i based on the data $\mathbf{x} \in A_i$ only. Then, for each of the $6k$ nodes we initialize a new component with the mean and covariance of the data present in that node. The initial mixing weights for candidates generated from A_i are set to $\pi_i/2$. The reader may have noticed that using only two layers of the 'local' kd-trees is somewhat arbitrary. However, experiments indicated that using more layers does not improve results significantly while slowing down the algorithm.

The *initial* candidates can be replaced easily by candidates that yield higher likelihood when mixed into the existing mixture. To obtain these better candidates we apply an EM algorithm again, but now to update only the candidate components and their mixing weights as to maximize $\mathcal{L}_{k+1}$, while keeping the existing mixture f_k fixed [15]. Each iteration of these 'partial' updates takes $O(nk)$ computations, since we have to evaluate the likelihood of each datum under each of the $6k$ candidate components.

We stop the partial updates if the change in log-likelihood of the resulting $(k+1)$-component mixtures drops below some threshold or if some maximal number of iterations is reached. After these partial updates we set the new component $\phi(\mathbf{x};\theta_{k+1})$ as the candidate that maximizes the log-likelihood when mixed into the existing mixture.

The run time of the proposed greedy algorithm is $O(nk^2)$, which is a factor k slower than the standard EM. However, this is compensated by the fact that standard EM must be started for several initial mixtures to reduce the dependency on the initialization.

As an example we included the evolution of a solution for artificially generated data. On the left the subsequent mixtures $f_1, \ldots, f_4$ are depicted by their mean and an ellipse which has the eigenvectors of the covariance matrix as axes and radii of twice the corresponding eigenvalue. On the right we plotted the candidate new components after the partial EM steps.

4 Experimental Results

In this experiment the task is to cluster $16 \times 16 = 256$ pixel gray-valued images. The images are patches taken from 512×512 images of 37 Brodatz textures. A small selection of these textures is provided in Figure 2. The number of textures involved in each experiment is denoted by $k \in \{2,3,4,5,6\}$. For each value of k we constructed 100 data sets by randomly extracting 500 patches from k randomly selected textures. The patches are represented as 256 dimensional vectors. In order to speed up the computations and to increase the validity of Gaussian noise, we projected these vectors linearly to a lower dimensional subspace by means of PCA as to retain 80% of the total variance. In this lower dimensional space (typically somewhere between 10 and 70 dimensions were retained) we learn a k component Gaussian mixture. Note that the 'correct' number of mixture components, k, is assumed known in this setup. If k is unknown, model complexity selection methods may be employed in order to find an apropriate value for k. Note that this task is facilitated by the use of a greedy algorithm to learn the Guassian mixture, since the algorithm generates a sequence of mixtures with an increasing number of components.

In Figure 3 we provide a table with the averages of $H(B \mid C)$ over 100 experiments for each value of k. We compared the new greedy method with one run of standard EM. Furthermore, we also compared the performance of the new method with the performance of the maximum likelihood mixture among several mixtures obtained by starting multiple standard EM runs until they

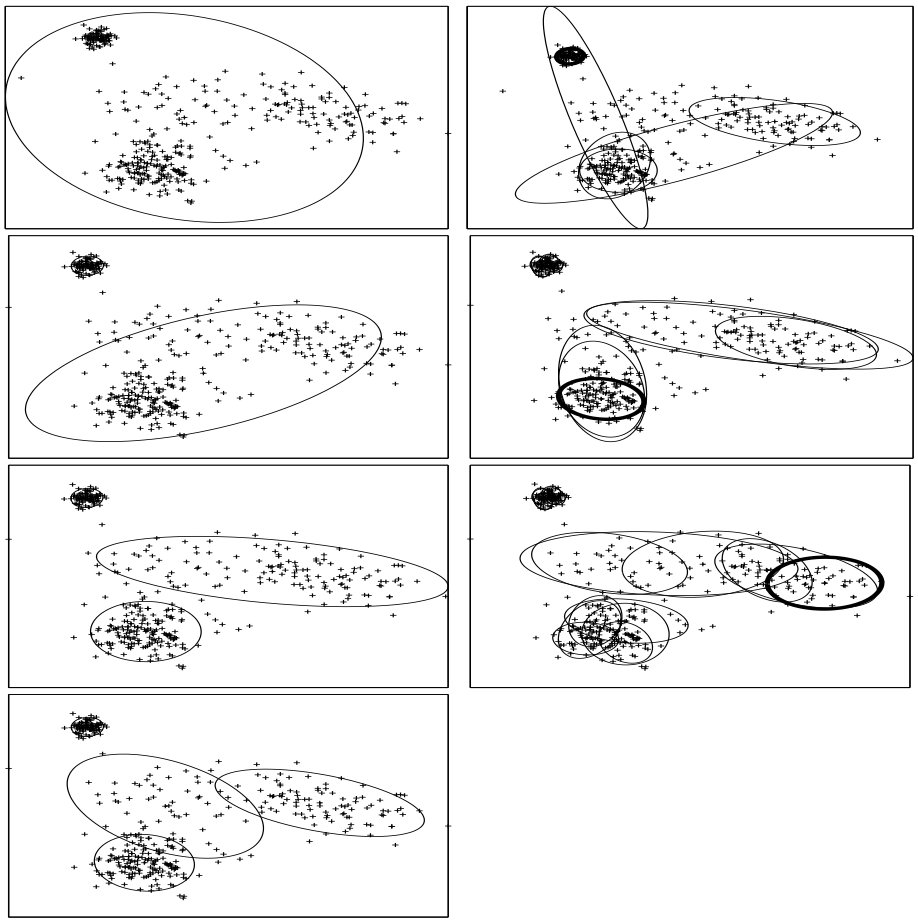

Fig. 1. A trace of the construction of a 4-component mixture distribution. Left are the mixtures $f_1, \ldots, f_4$, right are the candidate new components in each component allocation step. Thick lines indicate the selected candidate.

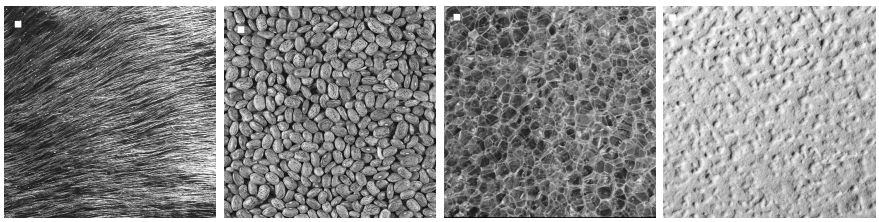

Fig. 2. Several Brodatz textures, the white squares indicate the patch size.

consumed the same amount of time as the new method. Typically 2 to 3 runs were allowed, this number of runs did not depend on the number of textures used. The conditional entropy corresponding to the clustering found by the greedy algorithm is only about 1/2 to 2/3 of the conditional entropy obtained when using standard EM.

k	2	3	4	5	6
Greedy method	0.32	0.51	0.72	0.80	0.92
Multiple starts of EM	0.53	0.81	1.06	1.32	1.40
One run standard EM	0.54	0.83	1.06	1.29	1.37
Uniform distribution	1	1.59	2	2.32	2.58

Fig. 3. Average conditional entropy for different values of k for the standard EM, standard EM with multiple random initializations, and the proposed greedy method to learn Gaussian mixtures compared with the conditional entropy for clusters distributing uniform over textures.

Figure 4 illustrates the difference between the greedy and the standard EM method in more detail for $k = 3$. The left plot shows 3 modes in the histogram of the conditional entropy:

1. Close to zero: Good segmentation, every component captures a texture. The conditional entropy is close to zero since there is almost no uncertainty about the texture class if we know the maximum posterior mixture component.
2. Close to 2/3 bit: One texture is separated and the two others are confused. In 1/3 of the cases we have very low entropy (the separated texture) and in 2/3 of the cases we have entropy close to 1 bit (the two confused textures). Since two textures are confused we need on average approximately one bit to indicate which one is used. Taking the average over all cases we arrive at approximately 2/3 bit.
3. Close to 3/2 bits: Very poor segmentation, all textures are confused. Note that $H(B \mid C) \approx 1.59$ bits for clusters that distribute uniformly over the textures.

In the right plot, showing performance of the greedy method, we see that the third mode has almost disappeared illustrating the superior performance of the mixture learned with the greedy method.

5 Discussion and Conclusions

Discussion In this paper we have shown the benefit of using our new greedy approach to Gaussian mixture learning for texture segmentation. For the segmentation we used non-restricted Gaussian mixtures. In many applications the patch size will be (much) larger than the 16×16 pixel windows used here. In

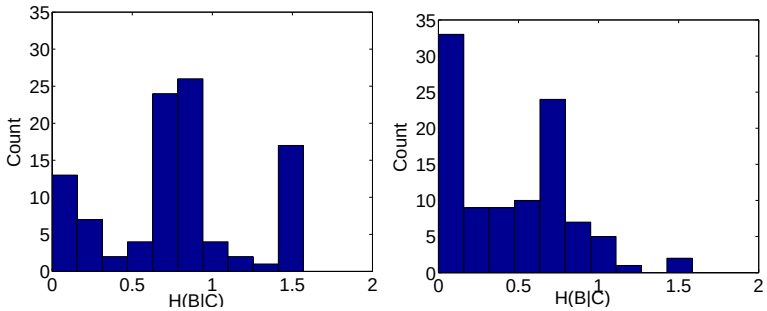

Fig. 4. Histograms of $H(B \mid C)$ for $k = 3$ comparing the standard EM (left) and the new greedy method (right) to learn the Gaussian mixture.

such cases it may be preferable to use the MPPCA model instead, as to limit the amount of computation needed. As described in [14], the new method extends naturally to the MPPCA model. The generation of candidate components for A_i in the component allocation step is modified as follows: The noise parameter σ^2 is set to the mean of the smallest $d - q$ eigenvalues of the covariance matrix of the data present in the node. The matrix $\mathbf{W}$ of the new components equals the q eigenvectors with largest eigenvalues of the covariance matrix of the data present in the node. The column of $\mathbf{W}$ containing the i-th eigenvector is multiplied by a factor $\lambda_i - \sigma^2$, where λ_i is the i-th eigenvalue. In the same report [14] experiments are described where the greedy method is applied to learning MPPCA models for image reconstruction. In that application the greedy method also boosted the performance of the MPPCA model significantly.

The component allocation step of the algorithm can be simplified (see [14]) to cost only $O(n)$ computations instead of $O(kn)$. If in step 4 of the general scheme an $O(n)$ procedure is used (instead of EM which costs $O(kn)$ operations) the total run time of the algorithm is reduced to $O(kn)$.

Conclusions We presented a new greedy method for learning Gaussian mixtures and shown its benefits when applied to texture segmentation based on Gaussian mixtures. The method is likely to be beneficial for many other applications involving Gaussian mixtures in the computer vision field as well as other fields of research. The new method is deterministic and is experimentally found to outperform standard EM significantly, even if multiple starts of EM are used. The determinism of the new method renders it especially useful as a part of larger systems: since the behavior of the total system will not depend on the randomness (of the initializations) of the Gaussian mixture learning, other parts of the system can be evaluated more easily.

Acknowledgment This research is supported by the Technology Foundation STW (project nr. AIF4997) applied science division of NWO and the technology programme of the Dutch Ministry of Economic Affairs.

References

1. J. L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.
2. T. Cover and J. Thomas. *Elements of Information Theory*. Wiley, 1991.
3. A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Statist. Soc. B*, 39:1–38, 1977.
4. G.E. Hinton. Modelling the manifolds of images of handwritten digits. *IEEE transactions on Neural Networks*, 8(1):65–74, 1997.
5. I.T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1986.
6. J. Q. Li and A. R. Barron. Mixture density estimation. In *Advances in Neural Information Processing Systems 12*. The MIT Press, 2000.
7. A. Likas, N. Vlassis, and J.J. Verbeek. The global k-means clustering algorithm. Technical report, Computer Science Institute, University of Amsterdam, The Netherlands, February 2001. IAS-UVA-01-02.
8. B. G. Lindsay. The geometry of mixture likelihoods: a general theory. *Ann. Statist.*, 11(1):86–94, 1983.
9. D. de Ridder, J. Kittler, and R.P.W. Duin. Probabilistic pca and ica subspace mixture models for image segmentation. In M. Mirmehdi and B. Thomas, editors, *British Machine Vision Conference*, pages 112–121, 2000.
10. D. de Ridder, J. Kittler, O. Lemmers, and R.P.W. Duin. The adaptive subspace map for texture segmentation. In A. Sanfeliu, J.J. Villanueve, M. Vanrell, R. Alquezar, J.O. Eklundh, and Y. Aloimonos, editors, *15th IAPR International Conference on Pattern Recognition*, pages 216–220. IAPR, IEEE Computer Society Press, 2000.
11. B.D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge, U.K., 1996.
12. R. F. Sproull. Refinements to nearest-neighbor searching in k-dimensional trees. *Algorithmica*, 6:579–589, 1991.
13. M. E. Tipping and C. M. Bishop. Mixtures of probabilistic principal component analysers. *Neural Computation*, 11(2):443–482, 1999.
14. J.J. Verbeek, N. Vlassis, and B. Kröse. Efficient greedy learning of Gaussian mixtures. Technical Report IAS-UVA-01-04, Computer Science Institute, University of Amsterdam, The Netherlands, May 2001.
15. N. Vlassis and A. Likas. A greedy EM algorithm for Gaussian mixture learning. Technical report, Computer Science Institute, University of Amsterdam, The Netherlands, September 2000. IAS-UVA-00-08, to appear in Neural Processing Letters.