Growing Gaussian mixtures network for classification applications

José L. Alba*, Laura Docío, Domingo Docampo, Oscar W. Márquez

Departamento de Tecnologías de las Comunicaciones, E.T.S.I. Telecomunicación, Campus Universitario de Vigo, 36200 Vigo, Spain

Received 26 March 1998; received in revised form 1 December 1998

Abstract

In this paper a method to automatically generate a Gaussian mixture classifier is presented. The growing process is based on the iterative addition of Gaussian nodes. Each iteration takes place in two sequential steps: first, using the EM algorithm, we maximize the likelihood of the data under the current configuration of the classifier; then, a new Gaussian node is added to the class which most improves the discriminant capabilities of the network. Growth control is imposed by means of a complexity penalizing term and a discriminant MMI condition. The classical EM algorithm for Gaussian mixtures is also extended to jointly include labeled and unlabeled data. We report some artificial experiments that show the utility of this extension and the reliability of the proposed growing technique. We also report results of the Growing Gaussian Mixtures Network on terrain classification over a Landsat-TM image using different restrictions on the covariance matrix of the Gaussian mixtures. Comparisons in classification performance with a set of MLP neural networks are provided. © 1999 Elsevier Science B.V. All rights reserved.

Zusammenfassung


Résumé

Cet article présente une méthode pour générer automatiquement un classificateur de mélange de gaussiennes. La procédure d'accroissement est basée sur l'addition itérative de noeuds gaussiens. Chaque itération est divisée en deux

*Corresponding author. Tel.: + 34-86-812126; fax: + 34-86-812116; e-mail: jalba@tsc.uvigo.es

0165-1684/99/$-see front matter © 1999 Elsevier Science B.V. All rights reserved.
PII: S0165-1684(98)00246-1
étapes séquentielles: dans la première étape, nous utilisons l’algorithme EM pour maximiser la fonction de vraisemblance des échantillons sous la configuration actuelle du classificateur; après cela, un nouveau noeud gaussien est ajouté à la classe qui augmente le plus les possibilités discriminantes du modèle. Le contrôle sur l’accroissement est imposé à l’aide d’un terme de pénalité sur la complexité du modèle et à l’aide d’une condition MMI discriminante. L’algorithme EM classique pour le mélange de gaussiennes est aussi développé pour inclure à la fois des échantillons étiquetés et non étiquetés. Nous présenterons des expériences artificiels qui montrent l’utilité de ce développement et la fiabilité de cette technique d’accroissement. Nous présenterons aussi des résultats de ce modèle de mélange de gaussiennes pour la classification de terrains en utilisant des images Landsat-TM. Dans cet exemple, nous imposons certaines restrictions de la matrice de covariance du mélange de gaussiennes. Finalement, on montre des comparaisons avec la performance de classification d’un ensemble de réseaux de neurones MLP. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: Gaussian mixtures; Growing process; EM algorithm; Classification; Neural network

### Nomenclature

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Omega_m$</td>
<td>indicator for class $m$</td>
</tr>
<tr>
<td>$\Xi$</td>
<td>Vector of parameters defining the multi-class Gaussian mixture model. It is composed by the following parameters: $\pi_m, 1 \leq m \leq M$ prior probability of each class $\Xi_m, 1 \leq m \leq M$ vector of parameters defining each mono-modal Gaussian mixture model. It is composed by the following parameters: $l_m$ number of Gaussian mixtures in class $\Omega_m$</td>
</tr>
<tr>
<td>$w_{lm}, 1 \leq l \leq l_m, 1 \leq m \leq M$</td>
<td>mixture coefficient for $l$th Gaussian node in class $\Omega_m$</td>
</tr>
<tr>
<td>$c_{lm}, 1 \leq l \leq l_m, 1 \leq m \leq M$</td>
<td>sample mean associated to the $l$th Gaussian node in class $\Omega_m$</td>
</tr>
<tr>
<td>$\Sigma_{lm}, 1 \leq l \leq l_m, 1 \leq m \leq M$</td>
<td>sample covariance matrix associated to the $l$th Gaussian node in class $\Omega_m$</td>
</tr>
<tr>
<td>$\gamma(x;\Xi)$</td>
<td>parametric approximation of the probability density function of $x$. It is defined as a multi-class Gaussian mixture</td>
</tr>
<tr>
<td>$\Phi(x;\Xi_m)$</td>
<td>parametric approximation of the probability density function of $x \in \Omega_m$. It is defined as a Gaussian mixture model for class $\Omega_m$</td>
</tr>
<tr>
<td>$\Psi_{lm}(x)$</td>
<td>$D$-dimensional Gaussian density function with mean $c_{lm}$ and covariance matrix $\Sigma_{lm}$</td>
</tr>
<tr>
<td>$f_{lm}(x_i)$</td>
<td>estimate of the posterior probability of Gaussian node $l$th in class $\Omega_m$ given the pattern $x_i$ of known class-membership</td>
</tr>
<tr>
<td>$q_{lm}(x_j)$</td>
<td>estimate of the posterior probability of Gaussian node $l$th in class $\Omega_m$ given the pattern $x_j$ of unknown class-membership</td>
</tr>
<tr>
<td>$l(\Xi;X)$</td>
<td>Maximum likelihood estimate (MLE) of data $X$ under the model parameterized by $\Xi$</td>
</tr>
<tr>
<td>$L(\Xi;X)$</td>
<td>$\log(l(\Xi;X))$</td>
</tr>
<tr>
<td>$AIC(\Xi)$</td>
<td>Akaike’s Information Theoretic Criterion for the model parameterized by $\Xi$</td>
</tr>
<tr>
<td>$I_{\Xi}(X, M)$</td>
<td>average mutual information between the input random variable $x$ and discrete random variable $m$, under the parameterized multi-class Gaussian mixture model</td>
</tr>
<tr>
<td>$\mathcal{C}(D_{lm})$</td>
<td>Cardinal number of the data set $D_{lm}$</td>
</tr>
</tbody>
</table>

### 1. Introduction

Gaussian mixtures (GM) have been widely recognized in the pattern recognition literature as a powerful clustering technique that generalizes the popular concept of Winner-Takes-All for partitioning the input space. GM techniques introduce a ‘soft’ assignment to the clusters by weighting the samples with their posterior probabilities under each Gaussian. The usual way to train the parameters of these models relies upon the maximization of the log-likelihood of the data; this is eased with the use of the Expectation-Maximization (EM) algorithm [8], a variable metric, gradient ascent algorithm that converges to a local maximum of the likelihood function.

Gaussian mixtures have also been found useful in classification applications, when each input observation is associated to a class-label. For that reason, GM modeling has entered the realm of
neural networks as a useful paradigm to train networks of Gaussian nodes, better known as Radial Basis Functions (RBF) networks. Nowlan [17] has made some comparisons on RBF networks, showing improvement in recognition performance when the centers and variances of the local receptive field units are estimated via a Gaussian mixture Maximum Likelihood (ML) problem. In this hybrid approach, the discriminant training has effects only on the network connections from the outputs of the Gaussian nodes to the network outputs. Streit [25] suited the multi-class GM classifier to a feed-forward network, training every class separately; the discriminant training in this case was only due to the probabilistic constraint on the class priors. Recently, Miller [15] has shown that the decision function of an RBF classifier is equivalent to the Bayes decision function associated with a particular mixture model whose Gaussian components are probabilistically assigned to classes. This equivalence also allowed the interpretation of the Miller’s generalized mixture model [14] as a particular case of a relative new type of neural network architecture called mixtures of experts [11,12] and, specifically, the model with a localized form of the gating network introduced by Xu et al. [27].

An open question, directly related to the general order selection problem, is the determination of the proper number of Gaussian nodes in the model. When the distribution of data can be modeled as a finite number of Gaussian mixtures, Fukunaga [10] showed that the likelihood of the data with respect to the model is a good criterion to find the number of mixtures. Unfortunately, this criterion fails when the distribution cannot be easily fitted by a simple GM model, as it happens when the complexity of the needed GM model is too large compared to the number of available data. Some papers in the neural networks literature have dealt with the related problem of increasing the number of local-receptive-field nodes during training. It is worth noting the architecture proposed by Platt (Resource-Allocating-Network [18]) that tries to create new nodes where a misclassification is detected or the growing network proposed by Roberts [20] for novelty detection that relies on a ‘novelty threshold’ to add new nodes. Even when these methods do not deal directly with the GM model, the equivalence shown by Miller allows us to think about them as interesting approaches to grow a GM model too.

By other hand, the mixtures of experts literature also covers the problem of model selection through iteratively growing and/or pruning experts which takes even more interest when dealing with hierarchically mixtures of experts (HME). The main differences among these methods rely on the selection of the expert to split. Waterhouse [26] evaluates the maximal increase in expert likelihood after splitting all the experts connected to the input. Fritsch [9] uses expert-dependent likelihood functions to select the minimum and split that expert. Saito [22] builds a similar expert-dependent error function to find the maximum and Ramamurti [19] makes a growing ME network for regression through an expert-dependent error function also, but evaluated over a validation set. In most of the cases the network complexity control (if any) is performed using a validation set and/or pruning.

In this paper we propose a growing Gaussian mixture network (GGMN) that adapts itself to the class-conditional distribution of the input and shows a good trade-off between accurate representation and correct classification. The representation commitment ensures that the GM model for each class could eventually fit the class-conditional pdf, in which case the optimum Bayes classifier could be directly applied. As this is not usually true in real-world applications, the classification goal requires the use of a growing discriminant condition. Due to the hard assignment of nodes to classes (that Miller called partitioned mixture model [14]) the decision of which node to split is changed now to the decision of which class needs a new node and where this node should be localized. In this way the growth method is governed by likelihood measurements over misclassified data sets, in a form that slightly resembles the expert-dependent measurements used by some researchers in the mixture of experts literature [20,22]. Besides, to avoid overfitting in the growth process, we also make use of an AIC criterion to penalize the complexity of the model. The complexity control imposed by the discriminant growing condition and the AIC criterion replaces the validation set in the growing process. We will see that this control leads to a stable network when every insertion of node is followed by pruning of the same node.
The rest of the paper is organized as follows. In Section 2 we formulate the GM model and recall the EM re-estimation equations for multi-class ML training. Section 3 deals with the growing algorithm and introduces the double condition that leads to discriminative placement of Gaussian nodes and to avoid over-fitting. Section 4 extends the EM formulation to include the common case of having unlabeled data within the data set that can be used to avoid overfitting if the labeled data set is small. Section 5 explains a real classification problem based on a Landsat Thematic Mapper image and shows extensive results. Comparisons among ML Gaussian, GGMN and MLP classification of this real data are also provided. Section 6 closes the paper with the conclusions.

2. ML training

The semi-parametric model we deal with is composed of a mixture of $M$ class-conditional densities. Each class is modeled as a multi-variate Gaussian mixture, a flexible choice to approximate any kind of multi-modal or mono-modal density function. From now on we will work with the approximation of the unknown pdfs:

$$\gamma(x; \Xi) = \sum_{m=1}^{M} \pi_m \phi(x; \Xi_m) = \sum_{m=1}^{M} \pi_m \sum_{l=1}^{l_m} w_{lm} \psi_{lm}(x),$$

(1)

where for the class $\Omega_m$, $l_m$ and $w_{lm}$ indicate the number of Gaussian nodes and the value of the $l$th

---

![Fig. 1. Network representation of a multi-class Gaussian mixture classifier.](image-url)
mixture coefficient, respectively, and $\Psi_{lm}(x)$ is a $D$-dimensional Gaussian density function with mean $\tilde{c}_{lm}$ and covariance matrix $\Sigma_{lm}$. We will denote by $\Xi_m$ the vector containing the free parameters of the class $\Omega_m$, and by $\Xi$ the whole set of parameters of the $M$ classes. When the input distribution can be parameterized as a mixture of Gaussian mixtures, and the training process is able to find the true parameters, this network works as a MAP Bayesian classifier by checking the maximum of the class-outputs: $\max_m\{\pi_m\Phi(x; \Xi_m)\}$. Fig. 1 represents the network that implements the multi-class GM model.

Model training is performed via the minimization of the Kullback–Leibler divergence between the true distribution and the one provided by the model:

$$K(p, q) = \int p(x) \ln \frac{p(x)}{q(x)} \, dx$$

that, for independent input patterns, leads to the Maximum Likelihood Estimate (MLE), based on the maximization of the function

$$l(\Xi; X) = \prod_{i=1}^{N} \gamma(x_i; \Xi).$$

This optimization problem is eased by the EM algorithm. The re-estimation equations for ML training when no restrictions on the covariance matrices are imposed can be performed separately for each class, and the class-priors have the simple form $\pi_m = N_m / N$, $N_m$ being the number of samples from the labeled data set assigned to the class $\Omega_m$. For every iteration of the EM algorithm the new parameters are

$$w_{lm}^{+1} = \frac{1}{N_m} \sum_{x_i \in X_m} f_{lm}^{(p)}(x_i),$$

$$\tilde{c}_{lm}^{+1} = \frac{1}{N_m w_{lm}^{+1}} \sum_{x_i \in X_m} f_{lm}^{(p)}(x_i)x_i,$$  \hspace{1cm} (2)

$$\Sigma_{lm}^{+1} = \frac{1}{N_m w_{lm}^{+1}} \sum_{x_i \in X_m} f_{lm}^{(p)}(x_i)[x_i - \tilde{c}_{lm}^{(p)}][x_i - \tilde{c}_{lm}^{(p)}]^T,$$

where $f_{lm}^{(p)}(x_i)$ is the estimate of the posterior probability of Gaussian node $l$th in the class $\Omega_m$ given the pattern $x_i$, when class membership is known:

$$f_{lm}(x_i) = \frac{w_{lm} \Psi_{lm}(x_i)}{\Phi(x_i; \Xi_m)}$$

If restrictions on covariance matrices are imposed, the re-estimation equations only change by pooling the tied matrices weighted by the priors of the Gaussian nodes associated to them [25]. In the last section we will analyze some experiments in which such kind of restrictions are used.

3. Discriminant selection of new mixture components

The determination of the proper number of Gaussian mixtures in a statistical model, as well as the number of clusters in a non-parametric clustering problem, is an open question. Using the MSE criterion to determine the number of clusters in a WTA algorithm produces no clue about the optimum because the MSE decreases with the addition of new nodes. Although ML-based algorithms suffer from a similar drawback, the log-likelihood function, when clusters are well separated, can be simplified in such a way that the negative entropy of the clusters $\sum_{l=1}^{K} w_{lm} \ln(w_{lm})$ appears naturally as a complexity-penalty term when the number of clusters increases [1]. This implies that, even though the log-likelihood function still increases with the number of nodes, a rough estimation of the network size can be extracted from the change of slope of the function that represents log-likelihood versus number of Gaussian nodes. However, the simplification that leads to this entropy term cannot always be applied because we usually do not know the degree of overlapping among clusters; it is thus necessary to add another penalty term.

The next two subsections recall two different criteria that we have implemented to find the proper network configuration. The first one includes a penalty term that tries to avoid data overfitting in a ML approximation. The second one introduces discriminant decisions in the growing process. Later on we will explain the way they have been implemented.
3.1. Complexity penalty

The Akaike’s Information Theoretic Criterion (AIC) [2] can be directly applied to this problem because it relies upon minimizing the mean of the Kullback-Leibler divergence over all data sets of \( N \) observations, implicitly involving a kind of cross-validation process that is the most usual way to deal with over-fitting in training:

\[
AIC(\Xi) = 2 \left[ -\sum_{i=1}^{N} \ln(\theta(x_i; \Xi)) + P \right].
\]

To use the AIC criterion we assume that the real probability distribution belongs to a hierarchically ordered family of models, whose members are characterized by a dimension \( P \) that accounts for the number of free parameters in the vector \( \Xi \): a more complex member is characterized by a larger \( P \). The objective function, \( AIC(\Xi) \), only depends on the member of the Gaussian mixture family parameterized by the vector \( \Xi \). Minimizing AIC results in selecting the model from the family that achieves the largest-likelihood using the minimum number of parameters. Therefore, Gaussian mixtures can be viewed as a family of models satisfying the requirements underlying the application of the AIC criterion. It has been shown, though, that when the true pdf is not well approximated by the parameterized model, the AIC criterion provides a more complex model than desired [23].

3.2. Discriminant criteria

On the other hand, as was noted in the introduction, we are concerned with the classification performance of the system; hence, the placement of new Gaussian nodes should be commanded by a discriminant criterion. Maximum likelihood estimate does not provide a discriminant solution because every class-conditional model, \( \Phi(x; \Xi_m) \), is trained without taking into account the observations and models from the other classes. At this point is interesting to highlight that the generalized (GM) mixture model proposed by Miller [14] as an RBF-equivalent classifier incorporates discriminant capabilities through the posteriors of each Gaussian node. The growing algorithm we propose has been developed with the restrictions of the ‘partition’ mixture (PM) model. The adaptation of the growing algorithm to the Miller’s GM model is left for future work.

The average mutual information [16] between the input random variable \( x \) and the random variable \( m \) over the members of the \( M \) classes turns out to be an ML-related criterion that can be used to provide, at least theoretically, a useful discriminative growing condition. \( I(X; M) \), where \( X \) denotes the set of labeled observations and \( M \) the set of classes, can be written as

\[
I(X; M) = \int \frac{\sum_{m=1}^{M} p(x, m) \ln \frac{p(x, m)}{p(x) p(m)}}{dx},
\]

where \( M \) indicates the number of classes. To deal with the finite set of observations, we need to approximate the statistical expectations by the sample average over the \( N \) training data; using the parameterized Gaussian mixture model for the distribution we have

\[
I(\Xi; M) = \frac{1}{N} \sum_{m=1}^{M} \sum_{x \in X_m} \ln \Phi(x; \Xi_m) - \frac{1}{N} \sum_{i=1}^{N} \ln \left( \sum_{m=1}^{M} \pi_m \Phi(x; \Xi_m) \right),
\]

that can also be seen as the likelihood-based relationship

\[
I(\Xi; M) = \sum_{m=1}^{M} L(\Xi_m; X \mid \Omega_m) - L(\Xi; X).
\]

3.3. Growing procedure

The growing algorithm starts with an ML Gaussian classifier, i.e. one node per class, and iteratively tries to insert a new node in one of the classes. The main idea underlying the algorithm [3] is to model class-regions of the feature space that are not well-fitted by the GM model of that class and consequently overlap with other classes. Now, taking into account that our final goal is to build a classifier, we should not try to fit in the same class clusters which do not contribute to decrease the classification error. Therefore, only misclassified patterns are allowed to command an increment in the complexity of the network.
The insertion algorithm can be summarized as follows:
1. Classify all the data using the model from the previous step. Wrong classified data indicate candidate regions to insert a new Gaussian node.
   For every $\Omega_m$ class do:
   1.1. Build a set $D_m$ of wrongly classified samples,
   \[
   D_m = \{ x \in \Omega_m | \pi_m \Phi(x; \Xi_m) < \pi_n \Phi(x; \Xi_n), \text{for some } n \neq m \}.
   \]
   1.2. For $l = 1$ to $l_m$, build subsets $D_{lm}$ following intra-class classification,
   \[
   D_{lm} = \{ x \in D_m | w_{lm} \Psi_{lm}(x) > w_{km} \Psi_{km}(x) \forall k \neq l \}.
   \]
   1.3. Find the Gaussian node $l^*$ with the worst representation capability over its own data,
   \[
   l^* = \arg \min_l \left\{ \frac{1}{|D_{lm}|} \sum_{x \in D_{lm}} f_{lm}(x) \right\}.
   \]
   1.4. Now, for every unduly winner class $\Omega_n$ in $D_{lm}$ build the subsets $D_{lm,n}$,
   \[
   D_{lm,n} = \{ x \in D_{lm} | \pi_m \Phi(x; \Xi_m) < \pi_n \Phi(x; \Xi_n) \}.
   \]
   1.5. Compute the likelihood of data in $D_{lm,n}$ with respect to the $l^*m$ Gaussian node,
   \[
   P_{lm}(D_{lm,n}) = \prod_{x \in D_{lm,n}} \pi_m w_{lm} \Psi_{lm}(x).
   \]
   1.6. For every overlapping class $\Omega_m$, compute the likelihood of data in $D_{lm,n}$ with respect to the Gaussian nodes in the class $\Omega_n$ with highest posterior probability $f_{lm}(x)$,
   \[
   P_n(D_{lm,n}) = \prod_{x \in D_{lm,n}} \pi_n w_{ln} \Psi_{ln}(x).
   \]
   1.7. Compute the log-likelihood ratio for all classes overlapping with the class $\Omega_m$,
   \[
   L_{mn} = \ln \frac{P_n(D_{lm,n})}{P_{lm}(D_{lm,n})}.
   \]
   Check whether $L_{mn} \leq$ insertion threshold.

2. Look for the maximum of the log-likelihood ratios that passed the insertion threshold; this indicates the candidate class to accept a new Gaussian node:
   \[
   m^*, n^* = \arg \max_{m,n} \{ L_{mn} \}; \quad m, n = 1, \ldots, M.
   \]

3. The new Gaussian node will be inserted at the mass center of $D_{lm^*,mn^*}$.

The insertion threshold used to check the pairwise log-likelihood ratio $L_{mn}$ is needed to detect when two classes have nearly the same probability of generating the subset $D_{lm,n}$ of misclassified data. The insertion in these cases would not be useful because of the unavoidable overlapped area.

Once the Gaussian node is inserted, the EM algorithm re-estimates the parameters of the candidate class. This process can easily shift the mean of the new Gaussian node, but this actually presents some advantages. Outliers are easily detected since they tend to create a Gaussian that sharpens at every iteration of the EM algorithm, driving the function to a singular point. Detection of outliers will prevent from adding a new node, and results in a ‘natural’ pruning process, but it is still necessary to satisfy the two criteria introduced previously (AIC and MMI) to drive the network through a non-complex discriminative solution. These two criteria are then transformed to a pair of insertion conditions.

3.4. Insertion conditions

The discriminative insertion condition is based on the increment of $I_\Xi(X;M)$ as defined in Eq. (3). In the growing process of the algorithm, the MMI criterion is used only to ensure that the new Gaussian node will increase the value of $I_\Xi(X;M)$ after being inserted into the candidate class. Performing some simple algebra the insertion condition for a new Gaussian node in the class $\Omega_m$ can be written as

\[
\sum_{x \in \Omega_m} \ln \frac{\phi(x; \hat{\Xi}_m)}{\phi(x; \bar{\Xi}_m)} > \sum_{l=1}^{N} \ln \frac{\gamma(x_l; \hat{\Xi})}{\gamma(x_l; \bar{\Xi})},
\]

where $\phi(x_l; \hat{\Xi}_m)$ and $\gamma(x_l; \hat{\Xi})$ stand for the likelihood of $x_l$ with respect to the new model of the class $\Omega_m$ and to the new multi-class model, respectively. Satisfying this condition means that the new Gaussian node is a better representative for the data within the class $\Omega_m$ than for the whole data set.

Besides, to prevent the model from overfitting, we have also applied the AIC criterion. The log-likelihood of the new model with $Q$ parameters is calculated and compared with the previous one.
with $P$ parameters, hence the decision to keep the new Gaussian node has to be also based on

\[
AIC(\tilde{Z}) < AIC(\hat{Z})
\]

\[
\Rightarrow \sum_{i=1}^{N} \ln(\gamma(x_i; \tilde{Z})) - \sum_{i=1}^{N} \ln(\gamma(x_i; \hat{Z})) > Q - P.
\]  

(4)

Note that the AIC minimization is performed over a subset of all the possible hierarchically ordered combination of Gaussian models, the subset composed by the increasingly complex networks coming out from the growing procedure. The difference $Q - P$ (difference of free parameters between $\tilde{Z}$ and $\hat{Z}$) is constant when we only insert one node at a time, and depends on the dimensionality of the input space and the kind of restrictions we impose on the covariance matrix.

The rationale for the application of this insertion criterion is the following: starting with few nodes very unlikely results in the fitting of a class-multi-mode distribution. As was noted before, we do not try to fit clusters in the same class that do not contribute to decrease the classification error; hence, only misclassified patterns will force the growth of the network. The re-estimation performed just after the node is inserted has the double task of detecting an outlier or fitting a new cluster. When an outlier is detected as explained in the previous subsection, node insertion has to be aborted before finishing the re-estimation to select a new candidate node. A clearly separable cluster creates a Gaussian that fits it and therefore the insertion conditions are easily satisfied. On the other hand, patterns lying in naturally overlapped regions can create nodes that are usually moved towards an old cluster; this usually means that the insertion conditions will not be satisfied. It is interesting to note here that a misclassified pattern that helped to create a Gaussian node can remain misclassified after the re-estimation; nevertheless, it has contributed to better represent its class near the decision boundary.

Extensive experiments have been made with artificially generated distributions to test the behavior of the GGMN. Fig. 2 shows a single realization of a bi-dimensional two-class distribution composed by 3 mixtures/class and 500 samples/class. This classification problem has been presented to the GGMN using an increasing number of data from 60/class to 1000/class and 10 realizations of the distributions in each data-case. Fig. 3 shows one example using GGMN with full covariance matrices. The graphics show the error rate represented as boxplots for every data-case. The $x$-axis is labeled with the number of training samples/class used and the mean number of Gaussian nodes/class achieved after the GGMN stops growing. The horizontal line shows the Bayes error. Both graphics show the same experiments but including or not the AIC as a insertion criterion, respectively. It is worth noting that the growing process stops around 3 nodes/class using AIC even if the number of training data increases. The AIC works fine because the distribution is contained in the family of models 'scanned' by GGMN. Results at the bottom graphic are slightly worse and GGMN grows a larger model.

These results can lead to think that using AIC is always recommended. But, what happens if the distribution cannot be fitted by a simple model? Fig. 4 shows the same experiments but using diagonal covariance matrices in the GGMN. A very complex model would be necessary in this case to fit the full covariance matrices Gaussian mixture model that generates the data. Both graphics show the error rate if AIC is used or not, respectively.

![Fig. 2. A single realization of the bi-dimensional two-class distribution.](image-url)
The extra nodes learned by GGMN without including AIC improves the performance of the classifier. It is also worth noting that GGMN with AIC creates more nodes than the previous experiment to cope with the lack of flexibility of the model.

In short, including AIC or not depends on the a priori knowledge about the likely fitness of the data with a GM and the trade off between error rate and complexity of the network.

4. EM for labeled and unlabeled data

In many real applications that require classification of new data it is quite difficult or expensive to get labeled patterns to be used in the training process. That is the case of remotely sensed or medical images, for example. In those cases, a parametric approach like the one we report here is more suitable than a pure non-parametric modeling, since the latter requires much more data for an accurate classification of unseen patterns. Even with these parametric models, the larger the number of labeled patterns, the better the estimate of the parameters. If we do not know whether the number of labeled data is enough for an accurate estimate, but have access to a huge unlabeled data set, it would be very useful to introduce the unlabeled data in the likelihood cost function and use the EM algorithm to reestimate the parameters. In this section we present the extension of the classical EM algorithm for Gaussian mixtures to deal with labeled \((x_i)\) and unlabeled \((x'_j)\) data.
The likelihood function for \( N \) labeled and \( N' \) unlabeled data can be written in this form:
\[
l(\mathbf{Z}; \mathbf{X}) = \prod_{i=1}^{N} \pi_c \phi(x_i; \mathbf{Z}_i) \prod_{j=1}^{N'} \sum_{m=1}^{M} \pi_m \phi(x_j'; \mathbf{Z}_m),
\]
where \( c_i \) indicates the class-membership of \( x_i \).

Maximizing this log-likelihood with the EM algorithm requires the definition of some unobserved variable \( z \) and a complete data set \( Y = (X,Z) \) \([4,12]\). The mathematical derivation of the optimization algorithm is left to Appendix A. Miller \([14]\) has also derived a couple of EM algorithms to deal with unlabeled data in his Generalized mixture model. The expressions we derive here can be seen as a particular case from those and differ mainly on the class priors, node priors and node posteriors for the labeled data set. Let us write down here the re-estimation expressions that result from our derivation:

\[
\begin{align*}
\pi^{(p+1)}_m &= \frac{N_m + \sum_{l=1}^{N'} \sum_{j=1}^{p} q^{(p)}_{lm}(x_j)}{N + N'}, \\
w^{(p+1)}_{lm} &= \frac{\sum_{i=lm} \phi^{(p)}_{lm}(x_i) + \sum_{j=1}^{p} q^{(p)}_{lm}(x_j)}{(N + N') \pi^{(p+1)}_m}, \\
c^{(p+1)}_{lm} &= \frac{\sum_{i=lm} \phi^{(p)}_{lm}(x_i) x_i + \sum_{j=1}^{p} q^{(p)}_{lm}(x_j) x_j}{(N + N') \pi^{(p+1)}_m w^{(p+1)}_{lm}}, \\
\Sigma^{(p+1)}_{lm} &= \frac{\sum_{i=lm} \phi^{(p)}_{lm}(x_i) (x_i - c^{(p)}_{lm})(x_i - c^{(p)}_{lm})^T + \sum_{j=1}^{p} q^{(p)}_{lm}(x_j) (x_j - c^{(p)}_{lm})(x_j - c^{(p)}_{lm})^T}{(N + N') \pi^{(p+1)}_m w^{(p+1)}_{lm}}.
\end{align*}
\]  

The functions \( f_{lm}(x_i) \) and \( q_{lm}(x_j) \) represent the estimate of the posterior probability of Gaussian node \( l \)th in the class \( \Omega_m \) given \( x_i \) and \( x_j \), when class membership is known and unknown, respectively.

When unlabeled data is not used, all the previous expressions reduce to those reported at Eqs. (2). Some comments can be made with respect to these re-estimation formulas:

- The posterior \( q^{(p)}_{lm}(x'_j) \) accounts for the whole set of classes as a pooled clustering problem, whereas \( f^{(p)}_{lm}(x) \) only takes care of patterns from the class \( \Omega_m \). Occasional singularity or ill-conditioning of the covariance matrices can thus be avoided including enough unlabeled data.
- The ‘unlabeled terms’ (those weighted by \( q_{lm}(x'_j) \)) try to model the pdf of \( x_i \), \( p(x) \). These terms represent a non-discriminative component in the EM re-estimation process.

- If classes are highly overlapped at regions of large \( p(x) \) values, and the real distribution is not well fitted by the Gaussian mixtures model, use of ‘unlabeled terms’ can lead to a worse performance than using only ‘labeled’ terms.

The effect of worsening the classifier has also been observed in \([24]\) in the context of remote sensing images when the input dimension was small compared to the number of labeled training samples. However, Miller \([14]\) has not observed this behavior in experiments with his generalized mixture model and reported better results than the partitioned mixture model used for Shahshahani and Landgrebe \([24]\) in a high-dimension satellite imagery data. This better behavior is probably due to the probabilistic assignment of nodes to classes that results in a more complex model that can fit the class-conditional distributions.

To avoid worsening the GGMN classifier accuracy in highly overlapped cases, we found it useful to perform an analysis of the degree of class-overlap after the growing process (easier than testing the degree of model fitness). We have used an empirical measurement; the likelihood ratio between the labeled data given the ‘unlabeled model’ and given the ‘labeled model’:

\[
h_{\Sigma}(X) = \frac{\prod_{i=1}^{N} \sum_{m=1}^{M} \pi_m \phi(x_i; \mathbf{Z}_m)}{\prod_{i=1}^{N} \pi_c \phi(x_i; \mathbf{Z}_c)}.
\]

The function \( -\ln(h_{\Sigma}(X)) \) can be also seen as the mutual information \( I_{\Sigma}(X; M) \) plus a constant term related to the entropy of the classes and the number of labeled data \( N \sum_{m=1}^{M} \pi_m \ln \pi_m \).

The value of \( h_{\Sigma}(X) \) is always larger than 1. The smaller the value of \( h_{\Sigma}(X) \), the smaller the overlap effect for the trained model and the higher the
probability of improving the classification rate using a non-supervised fine-tuning phase. Empirically we have observed that a threshold for $h_\gamma(X)$ can be found to ensure a reduction of the classification error when unlabeled data are included in the extended optimization algorithm. If the GM model fits the data this reduction is always guaranteed.

Figs. 5 and 6 show the classification error obtained using a GM model with an increasing number of Gaussian nodes per class over two differently overlapped distributions (two-class and three nodes/class with 30/class labeled data and 3000 unlabeled data). The test set was composed by 1500/class samples. Both figures show the sample distributions (100 data/class) at the top and the classification error using boxplot diagrams (10 runs each) training with the 30/class labeled data (left) and with the 3060 mixed data (right) at the bottom. The horizontal line represents the Bayes error for every data set.

With three or more Gaussian nodes per class the GM can fit the artificially generated distributions and the performance of the combined re-estimation always improves the classification performance, even if there is an important overlap effect. With one and two Gaussian nodes per class the behavior depends on the distribution because the model cannot fit it.

5. Classification of land cover from Landsat TM images

In this paper we focus on a real application involving the classification of remote-sensing data.
This section is dedicated to show the results using the algorithm proposed in the paper, and to make comparisons with MLP classifiers.

Classification of data from remote-sensors is widely recognized as a difficult problem. It requires the assignment of each pixel from a multi-channel image to a terrain class. Usually, the number of classes to be recognized is large, to accurately represent the diversity of land cover; moreover, there exists extensive overlapping areas among the classes. This overlap effect is mainly due to the spatial resolution of the sensors and/or their inability to capture the discriminant spectral characteristics of some pre-defined classes.

The most classical technique to build a classifier for a remotely-sensed image is based on the assumption of a multi-variate normal distribution for every class; the problem is then to estimate their means and covariance matrices from pre-classified data. The implementation of the maximum likelihood classifier under the Gaussian assumption leads to quite good classification rates for weakly overlapped and ‘well-defined’ classes; nevertheless, the assumption of normality does not hold in most practical cases; consequently, the equal likelihood boundary between two classes does not match the Bayes boundary.

Multilayer perceptrons are currently used to tackle this drawback of the ML classifier. The lack of assumptions about the probabilistic model is the main rationale to this non-parametric technique. It is known that a one-hidden layer perceptron is capable of producing any convex region in the feature space [13] and a two-hidden layer any form of connected or disconnected decision function (if the number of nodes is large enough). Bischof [5] has compared classification results over five classes in Landsat TM images using an MLP with one-hidden layer and an ML Gaussian classifier, obtaining significantly better results with the NN approach. Similar conclusions were extracted by Decatour [7] when using SAR data (Synthetic Aperture Radar). Most of the work done by different authors comparing both techniques resulted in higher classification rates using MLPs. Only two drawbacks of MLPs have been found: the difficulty of guessing the number of nodes needed in the hidden layer(s) to obtain the best architecture in terms of classification rate, and the large training time needed to guarantee the convergence of the algorithm. RBF networks have also been used [6] for classification of SAR data; no comparisons with other methods have been reported though.

The GGMN can be used to cope with the characteristic non-normality of the classes. The specific spectral characteristics of the class-distributions suggest the use of a different number of Gaussian nodes or mixtures for each class, so ‘guessing’ these numbers is the task of GGMN.

The Landsat database we have used for the comparison among ML Gaussian, MLPs and the proposed GGMN has been extracted from a Landsat mini-scene (47.2 × 44 km) of the northwest of Spain; it shows the city of Vigo and surrounding areas. The image contains a large variety of land covers which make the problem more challenging. The study has been divided in two parts. In the first part we have chosen 5 classes following a similar experiment as the one by Bischof [5]: built-up land, agricultural land, forest, water and unproductive land. In the second experiment we have sub-divided these classes into 12 categories: built-up land was divided into downtown and industry; agricultural was divided into pure agricultural exploitation and areas of sparse houses with agricultural exploitation around; forest was divided into pine, eucalyptus, oak and bush; water was divided into water bodies and swamp; unproductive land was divided into sand and rock. Pixels used for training and test were pre-classified according to ground truth.

Landsat satellites contain 7 sensors specially designed to help in the discrimination of natural terrains; 3 of them correspond to the visible spectrum, 3 to the infrared and 1 to the thermal region. Every pixel in the image represents a 30 × 30 meters area, except for the thermal band (120 × 120 m). All the experiments we have performed made use of another artificially preprocessed band to include contextual information. The value of this band was calculated as the convolution of a 5 × 5 window with the infrared image corresponding to band 5. This ‘textural’ filter calculates a normalized infrared dispersion centered on each pixel.

In this way, the input pattern is an 8-dimensional vector taking integer values [0,255]. Usually, some kind of coding is performed before presenting the
pattern to the MLP [5]. Coarse coding [21] has been frequently used with better results than the original inputs; however, this has the disadvantage of increasing the number of input nodes and, consequently, the training time and the difficulty to find a 'good' local minimum. Besides, this kind of coding is not useful to the Gaussian classifier (coarse coding is a sort of Gaussian preprocessing). Integer inputs caused some problems to the estimation of covariance matrices because it is easy to have a bad condition number when the corresponding Gaussian is located in a region with small or null variance in one of the dimensions (typically the thermal band). To cope with this problem we have added white noise with zero mean and small variance (less than 0.5) to all the components. The same inputs (normalized [0,1]) have been used for the MLP experiments. No difference has been noticed with respect to coarse coding.

All the experiments have been carried out with enough data to avoid over-fitting in the MLP network. In the 5-classes experiments, the training and tests sets were both composed of 2000 patterns per class. In the 12-classes experiments, we have used 1000 patterns per class for training as well as for testing. ML Gaussian classifiers have been trained with three different types of covariance matrices: full, diagonal and diagonal-tied (equal eigenvalues). We found it interesting to test another kind of restriction on the covariance matrix, a class-tied full covariance matrix weighted only by an independent factor for each Gaussian (it means that nodes from the same class share the same receptive-field shape). All the results are shown by means of boxplots. Every boxplot represents 10 runs of the corresponding algorithm; each run uses a new selection of $N_m$ training patterns and $N_m$ test patterns of the randomized $2N_m$ set.

5.1. Results with 5 classes

Two-dimensional projections of data from the 5 classes defined above show the non-normality of the class-distributions. Nevertheless we have tested ML Gaussian classifiers to get an idea of the improvement in classification rate when using Gaussian mixtures to capture these distributions. Fig. 7 (top) shows results from the three different variance restrictions. It is interesting to note the large difference between full-covariance results and the other ones; this points to a likely high correlation between the spectral features of the data in every class.

The Growing Gaussian Mixture Network has been trained for these data; the results are shown in Fig. 7 (bottom). The final average number of nodes per class is indicated in the x-axis. It is interesting to verify here the large improvement over the ML Gaussian classifier for all covariance types. The difference between error rates for full and class-tied full covariance matrices suggests that the defined classes are composed of various subclasses with different spectral correlations.

MLP classifiers have been trained with increasing number of nodes in the hidden layer. The
sigmoid non-linearity has been applied to every hidden and output node. Training has been performed using the standard backpropagation algorithm [21] with a learning rate $\mu = 0.2$. An independent validation set of 1000 samples has been used to stop the training when the MSE increased 1% from any previous value. Results of training and test are shown in Fig. 8. The only GGMN with performance comparable to the MLP is based on independent full covariance matrices for every node. MLP outperforms full matrix GGMN if the number of nodes in the hidden layer is bigger than 24. However, there is no significant improvement on the classification error above that number of nodes.

5.2. Results with 12 classes

The same experiments have been performed by considering the 12 subclasses previously defined. Projections in 2 dimensions have shown more ‘compact’ and Gaussian-shaped class-distributions than in the 5 classes case. Results for ML Gaussian classifiers, as shown in Fig. 9, highlight an interesting comparison between the 5 classes and the 12 (sub)classes classifiers: the error rate difference between diagonal and full covariance has been reduced. This effect means that the class subdivision achieved more uncorrelated spectral features inside every subclass.

GGMN has been trained with various insertion thresholds. The results for threshold = 5, 20 are shown at Fig. 10. The best results are obtained with the smallest insertion threshold; at the expense of a larger number of nodes. A detailed theoretical analysis of the optimum insertion threshold is still missing. As in the ML Gaussian case, it is interesting to observe the difference between 5 and 12 classes for the class-tied covariance matrix test. The behavior when the covariance matrices are shared within the same class is almost as good as when the matrices are independent. This effect clearly indicates that the 5 classes contained subclasses with different spectral variance; the subdivision into 12 classes has captured these differences and produced more Gaussian-shaped classes.

MLPs performance for 12 classes classification is shown in Fig. 11. The best performance obtained with GGMN is equalized or slightly outperformed if the number of nodes in the hidden layer is larger than 20. Nevertheless, it is interesting to highlight that MLP’s error rate decrease only a little bit more with the addition of new hidden nodes.

The experiments performed adding unlabeled data to the training process did not reduce the error rate. The high number of labeled data used to make a fair comparison with MLP was enough to avoid the Hughes effect so unlabeled data were not useful.
The computational complexity of the GGMN is larger than MLP with an adequate fixed number of hidden nodes, but the comparison is not fair because we do not usually know the correct MLP architecture and have to check out several network configurations. Every configuration is usually stopped through a validation set that increases the overall computational complexity. In the Landsat experiments we have used an independent validation set for MLP composed by 1000 samples (5 and 12 classes).

We run all the experiments in a Sparc Station 10. Training time for GGMN is quite dependent on the insertion threshold value. The largest computational burden corresponded to the 12-class full covariance matrices GGMN trained with an insertion threshold = 5 (top graphic at Fig. 10). GGMN lasted between 96 and 110 sec to create the average 45 nodes for each one of the 10 classifiers. By other hand, 20 hidden nodes MLP training lasted between 44 and 53 sec including validation set computations (Fig. 11). Obviously, if we sum up training times for the other MLP’s configurations (from 8 to 20 hidden nodes, for example), GGMN obtained its best architecture faster than MLP.

5.3. Discussion

The experiments we have performed clearly show the advantage of GGMN and MLP with respect to the standard ML Gaussian classifier when the definition of classes induces non-Gaussian class-distributions. The constructive algorithm we propose is able to create a Gaussian mixture classifier that fairly approximates arbitrary decision boundaries as MLP classifiers do.

The larger number of parameters required by Gaussian mixture classifiers compared to MLP is not followed by the necessity of a larger data set (we have used the same number of patterns for both experiments). The local influence of the parameters in the GGMN against the global influence in the MLP makes the difference. The growing conditions stop the algorithm before harmful over-fitting appears, but the obtained GGMN might be a little bit worse than an optimized MLP with the correct
number of hidden nodes. A careful study of the pruning effect of the insertion conditions is still missing.

The clear advantage of the proposed algorithm is the automatic distribution of nodes and the possibility of adding unlabeled data in the training process. The only parameter that must be selected beforehand is the insertion threshold, that will be indirectly related to the "final number of nodes and error rate."

6. Conclusions and future research

We have presented a novel growing technique for Gaussian mixture classifiers. The basis of the growing process rely on EM parameter re-estimation every time a new Gaussian node is inserted into a candidate class using a discriminative algorithm. An AIC complexity penalty term and an internal threshold for allowing insertion contribute to reduce overfitting and automatically stop the growing process. Another discriminative condition based on the MMI criterion ensures a balanced growth between representativeness and discriminant capabilities of the final GGMN classifier.

The possibility to include labeled and unlabeled data allows to take advantage of the information contained in large databases when the labeling process is expensive. The results obtained for artificial and real data have shown the reliability of the proposed method and the comparable performance to powerful non-Gaussian classifiers like MLP neural networks.

Our next step consist of adapting Miller’s generalized mixture model to the GGMN algorithm and make extensive tests jointly including labeled and unlabeled data. Another point of research is the pruning effect of the insertion conditions that still seems to slightly overfit the training data set.

Acknowledgements

We wish to acknowledge the help of Julio Martin who labeled the Landsat database and Simón Ruibal who made extensive tests with it. We wish to thank also D.J. Miller for his interesting comments about the generalized mixture model and to the reviewers of this paper for their useful comments and for pointing out a couple of well related references.

Appendix A

The maximization of the likelihood function of the data set with respect to the GM model in Eq. (5) is eased with the EM algorithm. When the data set is composed by labeled and unlabeled data we can define a new unobserved variable, $z$, that will associate each labeled or unlabeled $x$ with a single Gaussian component giving us a complete data set $Y = (X, Z)$. Let us see how $z$ can be linked to the likelihood function.

For each member of the unlabeled data set, $x_j$, let us define the random variable $z_j$ taking values on the set of bi-dimensional indexes, $(l,m), 1 \leq l \leq l_m, 1 \leq m \leq M$; $z_j$ will account for the Gaussian from the mixture of any class that actually generated the sample $x_j$, so that

$$P(z_j = (l, m) | x_j) = \pi_m w_{lm}, \quad l = 1, \ldots, l_m, \quad m = 1, \ldots, M.$$

Let now $z_j$ be an indicator random variable linked to $x_j$ in the following way:

$$z_j(l, m) = \begin{cases} 1, & x_j = (l, m), \\ 0, & x_j \neq (l, m). \end{cases}$$

The likelihood of $(x_j', z_j)$ can now be written as

$$P(x_j', z_j | \Xi) = \prod_{m=1}^{M} \prod_{l=1}^{l_m} (\pi_m w_{lm} W_{lm}(x_j'))^{z_j(l, m)}.$$
Let now \( P \in \mathcal{O}_c \) be the random variable taken values on the same set as \( z'_i \). \( z_i \) will account for the Gaussian from the mixture of the class \( \Omega_c \) that actually generated the sample \( x_i \), so that

\[
P(z_i = (l, m)) = w_{lm} \delta(m - c_i), \quad l = 1, \ldots, l_m, \quad m = 1, \ldots, M.
\]

Let \( z_i \) be an indicator random variable linked to \( z_i \) as before

\[
z_i(l, m) = \begin{cases} 1, & z_i = (l, m), \\ 0, & z_i \neq (l, m). \end{cases}
\]

The likelihood of \((x'_i, z'_i)\) can now be written as

The likelihood of \((x_i, z_i)\) can be expressed as

\[
P(x_i, z_i; \Xi_c) = \prod_{l=1}^{l_i} (w_{lc} \Psi_{lc}(x_i))^{z_i(l, c)}.
\]

The likelihood of the complete data (ordering the products for every class and taking logarithms) is the random variable

\[
L_c(\Xi; Y) = \sum_{m=1}^{M} \sum_{c_i = c} \left( \ln \pi_m + \sum_{l=1}^{l_m} z_i(l, m) \ln(w_{lm} \Psi_{lm}(x_i)) \right) + \sum_{m=1}^{M} \sum_{j=1}^{N} \sum_{l=1}^{l_m} z'_j(l, m) \ln(\pi_m w_{lm} \Psi_{lm}(x'_j)).
\]

Step E in the EM algorithm takes the expectation of \( L_c(\Xi; Y) \) given the distribution of the observed variables, using the estimated parameters from the previous iteration \((p)\). This expectation is a deterministic function

\[
U(\Xi; \Xi^{(p)}) = E\{L_c(\Xi; Y) \mid X; \Xi^{(p)}\} = \sum_{m=1}^{M} \ln \pi_m^{(p)} \left( N_m + \sum_{j=1}^{N} \sum_{l=1}^{l_m} q_{lm}^{(p)}(x'_j) \right) + \sum_{m=1}^{M} \ln w_{lm} \left( \sum_{c_i = m} \sum_{l=1}^{l_m} f_{lm}^{(p)}(x_i) + N' \sum_{j=1}^{N} q_{lm}^{(p)}(x'_j) \right)
\]

\[
+ \sum_{m=1}^{M} \sum_{l=1}^{l_m} \left( \sum_{c_i = m} f_{lm}^{(p)}(x_i) \ln \Psi_{lm}(x_i) + \sum_{j=1}^{N} q_{lm}^{(p)}(x'_j) \ln \Psi_{lm}(x'_j) \right),
\]

where we have expressed the conditional expectations of \( z_i(l, m) \) and \( z'_j(l, m) \) as

\[
E\{z_i(l, m) \mid x_i; \Xi^{(p)}_m\} = \frac{P\{x_i \mid z_i(l, m) = 1; \Xi^{(p)}\} P\{z_i(l, m) = 1; \Xi^{(p)}\}}{P(x_i; \Xi^{(p)}_m)} = \frac{w_{lm} \Psi_{lm}(x_i)}{\Phi_{lm}(x_i)} = f_{lm}^{(p)}(x_i),
\]

\[
E\{z'_j(l, m) \mid x'_j; \Xi^{(p)}\} = \frac{P\{x'_j \mid z'_j(l, m) = 1; \Xi^{(p)}\} P\{z'_j(l, m) = 1; \Xi^{(p)}\}}{P(x'_j; \Xi^{(p)}_m)} = \frac{\pi_m^{(p)} w_{lm} \Psi_{lm}(x'_j)}{\sum_{m=1}^{M} \pi_m^{(p)} w_{lm} \Psi_{lm}(x'_j)} = q_{lm}^{(p)}(x'_j).
\]

Step M in the EM algorithm maximizes \( U(\Xi; \Xi^{(p)}) \) to obtain the \((p + 1)\)th estimation of the model parameters

\[
\pi_{lm}^{(p+1)} = \frac{N_m + \sum_{i=1}^{N} \sum_{l=1}^{l_m} q_{lm}^{(p)}(x'_j)}{N + N'}, \quad w_{lm}^{(p+1)} = \frac{\sum_{i=1}^{l} f_{lm}^{(p)}(x_i) + \sum_{j=1}^{N} q_{lm}^{(p)}(x'_j)}{(N + N') \pi_{lm}^{(p+1)}},
\]

\[
\bar{c}_{lm}^{(p+1)} = \frac{\sum_{i=1}^{l} f_{lm}^{(p)}(x_i) x_i + \sum_{j=1}^{N} q_{lm}^{(p)}(x'_j) x'_j}{(N + N') \pi_{lm}^{(p+1)} w_{lm}^{(p+1)}},
\]

\[
\Sigma_{lm}^{(p+1)} = \frac{\sum_{i=1}^{l} f_{lm}^{(p)}(x_i) (x_i - \bar{c}_{lm}^{(p)}) (x_i - \bar{c}_{lm}^{(p)})^T + \sum_{j=1}^{N} q_{lm}^{(p)}(x'_j) (x'_j - \bar{c}_{lm}^{(p)}) (x'_j - \bar{c}_{lm}^{(p)})^T}{(N + N') \pi_{lm}^{(p+1)} w_{lm}^{(p+1)}}.
\]
References