Maximum Likelihood Training of Probabilistic Neural Networks

Roy L. Streit, Senior Member, IEEE and Tod E. Luginbuhl, Member, IEEE

Abstract—A maximum likelihood method is presented for training probabilistic neural networks (PNN's) using a Gaussian kernel, or Parzen window. The proposed training algorithm enables general nonlinear discrimination and is a generalization of Fisher's method for linear discrimination. Important features of maximum likelihood training for PNN's are: 1) it economizes the well known Parzen window estimator while preserving feedforward NN architecture, 2) it utilizes class pooling to generalize classes represented by small training sets, 3) it gives smooth discriminant boundaries that often are "piece-wise flat" for statistical robustness, 4) it is very fast computationally compared to back-propagation, and 5) it is numerically stable. The effectiveness of the proposed maximum likelihood training algorithm is assessed using nonparametric statistical methods to define tolerance intervals on PNN classification performance.

I. INTRODUCTION

CLASSIFICATION is the following decision problem: given an input vector \( X \), decide to which of several known classes the input \( X \) belongs. The classes are assumed to be mutually exclusive and exhaustive. Useful characterizations of the classes are assumed to be either unknown or unavailable and must be estimated from a given collection of labelled training samples (i.e., input vectors corresponding to each class). The absence of a priori class characterizations is the major difficulty in classification.

The training samples available for each class reflect the intrinsic variability of the class. Measurement errors are normally present in the training samples also, but such errors are subsampled here in the guise of class variability. Class variability models developed in this paper are based on the following fundamental assumptions:

1) Each class is a multivariate random variable with a continuous class conditional probability density function (PDF).
2) Every input vector \( X \) is a realization of one of the classes.
3) Each vector in the training sample set \( T \) is a realization of the random variable corresponding to its class label.

From the first two assumptions it follows that the well known Bayesian classifier [1, Chapter 13] is the optimum classifier in the sense of minimizing the overall misclassification risk. The implementation of a homoscedastic Gaussian mixture (defined in Section II) approximation to the optimum classifier in a probabilistic neural network (PNN) structure is discussed in Section II.

Obtaining meaningful class conditional PDF estimates for classes represented by only a few samples in the training set \( T \) is a difficult and thorny problem, but one that occurs often in practice. We treat the small sample size problem by a new sample pooling method that generalizes a classical statistical technique due to Fisher [2, Chapter 4]. We refer to this new method as Generalized Fisher (GF) training, and it yields (local) maximum likelihood estimates of the class conditional PDF's. GF training is discussed in Section III and derived in the Appendix. A discussion of the training of a priori class probabilities and misclassification costs is given in Section IV. Examples of GF training on small, moderate, and large size training sets \( T \) are presented in Section V, and the effectiveness of GF training for these examples is assessed in Section VI using a nonparametric statistical method called tolerance intervals. Concluding remarks are given in Section VII.

II. NEURAL NETWORK IMPLEMENTATION OF MIXTURE GAUSSIAN PNN'S

The purpose of this section is to show that a four layer feed-forward PNN using a general Gaussian kernel, or Parzen window, can implement exactly the general homoscedastic Gaussian mixtures used in this paper to approximate the optimum classifier. Maximum likelihood training of the PNN is discussed below in Sections III and IV. The structure of the required PNN is represented in Fig. 1. Each component of this PNN has a specific interpretation and, moreover, all the interconnection weights and nodal thresholds are given explicitly by mathematical expressions involving the defining parameters of the mixture Gaussian PDF estimates and the a priori class probabilities and misclassification costs.

Let \( N \) denote the dimension of the input vector \( X \), and let \( M \) denote the number of different class labels in the training set \( T \). For \( j = 1, \ldots, M \), let \( G_j \geq 1 \) denote the total number of different components in the \( j \)th class mixture PDF. Let \( p_{ij}(X) \) denote the multivariate PDF of the \( i \)th component in the mixture for class \( j \), and let \( \pi_{ij} \) denote the proportion of component \( i \) in class \( j \). The “within-class” mixing proportions \( \pi_{ij} \) are non-negative and satisfy the equations

\[
\sum_{i=1}^{G_j} \pi_{ij} = 1, \quad j = 1, \ldots, M.
\]
The PDF of class $j$, denoted by $f_j(X)$, is approximated by a general mixture PDF, denoted by $g_j(X)$, that is,

$$f_j(X) \approx g_j(X) = \sum_{i=1}^{G_j} \pi_{ij} p_{ij}(X), \quad j = 1, \ldots, M. \quad (2.2)$$

In this paper, the PDF $p_{ij}(X)$ is assumed to have the form

$$p_{ij}(X) = (2\pi)^{-N/2} |\Sigma|^{-1/2} \exp \left\{ -\frac{1}{2} (X - \mu_{ij})^\top \Sigma^{-1} (X - \mu_{ij}) \right\}, \quad (2.3)$$

where $\mu_{ij}$ is the mean vector and $\Sigma$ is the positive definite covariance matrix of $p_{ij}(X)$, and where superscript $\top$ denotes transpose. The PDF $g_j(X)$ is a homoscedastic Gaussian mixture because the covariance matrix $\Sigma$ is chosen independent of the class index $j$ and the component index $i$, for reasons discussed in Section III. The results presented in this Section are readily extended to the general heterogeneous case of different covariance matrices. For details, see [3] and [4].

Let $\alpha_l$ denote the a priori probability of class $l$, that is, $\{\alpha_l\}$ are the "between-class" mixing proportions. Let $c_{jl}$ denote the loss associated with classifying an input vector $X$ into class $j$ when the correct decision should have been class $l$. The risk $r_j(X)$ of classifying the input $X$ into class $j$ is the expected loss, so that

$$r_j(X) \approx \sum_{l=1}^{M} c_{jl} \alpha_l g_l(X). \quad (2.4)$$

The decision risk $r_j(X)$ is, thus, approximated by a mixture of Gaussian PDF's, as is seen by substituting (2.2) into (2.4). The minimum risk decision rule is to classify $X$ into that class $j$ having the minimum risk, that is, $j = \arg \min \{r_j(X)\}$. The decision $j$ is the optimum classification decision [1, Chapter 13] if $g_j(X) \equiv f_j(X)$ for all $j$, that is, provided the approximation (2.2) is an equality. (Ties for minimum risk occur with probability zero, so they can be decided arbitrarily in practice.) The simple task of selecting the minimizing index $j$ can be performed in many ways, and it has been pointed out [5] that a NN structure can be used for this task if desired.

The nodes in each of the four layers play specific roles in the PNN. The output of a fourth layer node is the risk $r_j(X)$ of choosing class $j$, as given by the approximation (2.4). The fourth layer therefore requires as many nodes as classes, namely $M$. A fourth layer node is conceptually equivalent to a decision risk. The output of a third layer node is $g_j(X)$, the approximate class conditional PDF given by equation (2.2). A node is needed for each class, so there are $M$ third layer nodes. A third layer node is conceptually equivalent to a statistical hypothesis. The output of a second layer node is the likelihood $p_{ij}(X)$ of a component in a class mixture. A second layer node is needed for each Gaussian component in each class, so there are

$$G \equiv G_1 + G_2 + \ldots + G_M$$

second layer nodes. A second layer node is conceptually equivalent to a multivariate Gaussian random variable. A first layer node is needed for each degree of freedom in the $\chi^2$ distributed exponent (the expression in braces in equation (2.3)) of every multivariate Gaussian component. There are $N$ degrees of freedom and $G$ components, so there are $NG$ nodes in the first layer.

The activation function appropriate for a node depends upon the layer in which it resides. All fourth and third layer nodes
use the identity function with a zero threshold, or bias. The second layer nodes use the function $exp(-x/2)$ with a zero bias. The first layer nodes all use the activation function $|x|$, but the biases vary from node to node across the layer. Explicitly, the first layer biases are given by

$$
\tau_{ijk} = \begin{bmatrix} L^{-1} & \mu_{ij} \end{bmatrix}_k,
$$

$i = 1, \ldots, G$, $j = 1, \ldots, M$, $k = 1, \ldots, N,$

where $L$ is any square root matrix factor of the covariance matrix $\Sigma$, that is $\Sigma = LL^T$. The Cholesky factor is one such square root. The bias $\tau_{ijk}$ depends on the destination second layer node via the mean vector $\mu_{ij}$. Further discussion of the activation functions and biases of the first three layers of the PNN are given in [3].

The description of the trained PNN is completed by defining interconnection weights between the layers, and giving their specific roles in the PNN. We begin with the top two layers and work down the NN. The interconnection weight between node $l$ in the fourth layer and node $j$ in the third layer is the product $\alpha_{l,j}$. These weights characterize decision risk formation. The interconnection weight between a third layer node (class mixture) and a second layer node (Gaussian component) is zero if the component does not belong to the class mixture, and is the mixing proportion $\pi_{ij}$ if it is component $i$ of the class $j$ mixture. These weights characterize mixture formation. The interconnection weight between a second and a first layer node is either 1 or 0, depending on whether or not a given degree of freedom (first layer node) belongs to a given Gaussian random variable (second layer node). These weights characterize $\chi^2$ random variable formation. Finally, the interconnection weights between the first and input layers are given by the entries of the inverse of the square root matrix factor $L$ of the covariance matrix $\Sigma$. There are a total of $G$ components, and $L^{-1}$ is $N \times N$, so this gives $G N^3$ interconnection weights. If $L$ is chosen to be the Cholesky factor of $\Sigma$, then $L^{-1}$ is lower triangular and nearly half the weights between the first and input layers are zero. Alternatively, the matrix $L$ can be chosen so that it characterizes the discrete Karhunen-Loève transformation corresponding to $\Sigma$, that is, $L^{-1} = \Lambda^{-1/2}U^T$, where $\Sigma = U \Lambda U^T$ is the singular value decomposition of $\Sigma$. In this case, the sparsity of $L^{-1}$ is not immediately evident. A more detailed description of interconnection weights is given in [3].

III. GENERALIZED FISHER TRAINING OF PROBABILISTIC NEURAL NETWORKS

The PNN proposed by Specht [6] is a special case of the PNN described in Section II, as is seen by setting the costs $c_{ij} = 0$ for all $l$ and $c_{ij} = 1$ for $j \neq l$, and noting that the fourth layer is essentially superfluous in this case. Specht’s PNN implements the Parzen window PDF estimator [7] using the so-called product Gaussian (i.e., uncorrelated Gaussian) window. The Parzen window sets the interconnection weights and nodal activation functions. Specht’s PNN is thus a three layer feedforward NN that uses mixtures of uncorrelated Gaussians to estimate the class conditional PDF’s.
Specht's PNN is an excellent tool for initial exploration of new large training sets. Nonetheless, its usefulness in practice is limited by two factors. Firstly, because it is based on the Parzen window estimator, the total number $G$ of Gaussian components must equal the number of samples in the training set $T$. Therefore, it requires large amounts of data storage when extensive training sets are available. Secondly, an intrinsic smoothing parameter must be estimated on the basis of classification performance. Since robust estimates of classification performance are difficult to establish for small sample size, estimates of the smoothing parameter may be unreliable in practice. Both factors can often be mitigated by heuristics suited to the particular application. The contribution of generalized Fisher (GF) training is that it successfully treats both these problems without the need of heuristics.

The GF trained PNN requires significantly fewer nodes and interconnection weights than Specht's PNN in most problems of practical interest. A careful comparison of the two architectures below the third layer shows that the GF trained PNN is more efficient in both nodes and weights if

$$G \leq \frac{T}{\beta N^2},$$

where $\beta$ is the sparseness index of the inverse square root matrix factor of the covariance matrix $\Sigma$, that is, $L^{-1}$ has $\beta N^2$ nonzero entries. $L^{-1}$ is fully dense if $\beta = 1$ and least dense (diagonal) if $\beta = 1/N$. By choosing $L$ to be the Cholesky factor, the index $\beta$ can always be made at least as small as $(N + 1)/(2N) \approx 1/2$. From inequality 3.1, it is clear that the trained PNN is most effective in reducing node and weight requirements in large training set problems. For small training sets, the reduction in the number of nodes and weights depends on the sparseness index $\beta$ of $L^{-1}$.

We begin the discussion of GF training by reviewing a classical treatment of the two class discrimination problem: Fisher's linear discriminant (FLD). FLD is based on the premise that both classes are multivariate Gaussian random variables with a common covariance matrix $\Sigma$, but different mean vectors $\mu_1$ and $\mu_2$. The available training set $T$ is assumed to be correctly labelled, and we write $T = T_1 \cup T_2$, where $T_j$ denotes the subset of $T$ with class label $j$. The sample means

$$\hat{\mu}_j = \frac{1}{\#(T_j)} \sum_{X \in T_j} X, \quad j = 1, 2,$$

estimate the means $\mu_1$ and $\mu_2$, where $\#(\cdot)$ denotes the cardinality (number of samples) of a training set. The covariance matrix $\Sigma$ is estimated by Fisher's within-class scatter matrix

$$\hat{\Sigma} = \frac{1}{\#(T)} \sum_{j=1}^{2} \sum_{X \in T_j} (X - \hat{\mu}_j)(X - \hat{\mu}_j)^T.$$

The estimation error for $\Sigma$ is reduced by pooling the sample data, i.e., by using all samples in the training set $T$. Given these estimates, the log-likelihood ratio is evaluated for an unknown vector $X$ to be classified. The classification decision is obtained by comparing this ratio to an appropriate threshold. Contours of constant log-likelihood ratio are in this case
hyperplanes in the feature space (i.e., \( R^N \)), and the hyperplane corresponding to the threshold value is the FLD.

The FLD is known to be robust in the sense that linearly separable classes are often successfully discriminated in practice when neither class is truly Gaussian. Even when both classes are Gaussian but have different covariance matrices, some authors have observed that the FLD is often a better classifier than the optimum quadratic discriminator. For fixed training set size, the increased estimation error in the two covariance matrices that results from not pooling the training samples is, presumably, the cause of the relatively greater robustness of the FLD in this case.

Pooling the training samples provides a natural way of developing PDF estimates for classes that have few samples in the training set \( T \). In applications where samples from different classes have broadly similar correlational structure, it is reasonable to pool the training samples when the sample size is small. Moreover, in practice, in the absence of a priori information to the contrary, it is probably inevitable that pooling will be used to generalize small sample set classes. Pooling is the basic strategy adopted in this paper.

GF training is a generalization of the FLD methodology. It uses a homoscedastic “mixture of mixtures” assumption to formulate a posterior likelihood function \( L \) for the entire training set \( T \). An ordinary homoscedastic mixture PDF is a mixture in which the components share a common covariance matrix \( \Sigma \). By the term homoscedastic mixture of mixtures, we mean that a common covariance matrix \( \Sigma \) is used within each class mixture and also across all classes represented in the training set \( T \). The likelihood function \( L \) is highly nonlinear in the defining parameters of the mixture of mixtures, and it is not generally possible to factor it into terms depending on only one class label. Maximum likelihood parameter estimates for each class PDF are therefore jointly dependent on training samples across all classes.

Maximum likelihood parameter estimates for the mixture of mixtures are obtained numerically by utilizing an algorithm based on the Expectation-Maximization (EM) method [8]. The derivation of the GF training algorithm is given in the Appendix. The amount of training data required for GF training is also discussed in the Appendix (see Theorem). The remainder of this Section is devoted to formulating the likelihood function \( L \), to stating the most important properties of GF training, and to interpreting its maximum likelihood solution. Equations (A.63) – (A.67) summarize the GF algorithm iteration from step \( n \) to step \( n + 1 \).

The parameters \( \lambda \) defining the homoscedastic mixture of mixtures comprises the following variables:

- \( \alpha_j \) = the a priori probability of class \( j \),
- \( \pi_{ij} \) = the mixing proportion of component \( i \) in class \( j \),
- \( \mu_{ij} \) = the mean vector of component \( i \) in class \( j \), and
- \( \Sigma \) = the common covariance matrix of all Gaussians.

Thus, \( \lambda \) comprises a total of \( M + G + NG + N^2 \) real variables, though not all of them are independent (e.g., \( \Sigma \) is symmetric and mixing proportions sum to 1). For \( j = 1, \ldots, M \), let

\[
\lambda_j = \{ \pi_{ij}, \mu_{ij}, \Sigma \}_{i=1}^{G}
\]
denote the parameters defining the homoscedastic Gaussian mixture for class \( j \). The labelled training set \( T \) is partitioned into the disjoint subsets

\[ T = T_1 \cup T_2 \cup \ldots \cup T_M, \]

where \( T_j \) comprises those samples in \( T \) with class label \( j \). The posterior likelihood function \( \mathcal{L}(T|\lambda) \) is defined on \( T \) by assuming that the samples in \( T_j \) are independent for each \( j \), and that the class labels are assigned correctly. From these assumptions, it follows from equation (A.12) in the Appendix that the GF log-likelihood function is

\[
\log \mathcal{L}(T|\lambda) = \sum_{j=1}^{M} \sum_{X \in T_j} \log \left[ \alpha_j g_j(X|\lambda_j) \right],
\]

where the function \( g_j(X|\lambda_j) \) is identical to the class PDF \( g_j(X) \) defined by equation (2.2). Estimating the parameter set \( \lambda \) is the central task of GF training.

The GF training algorithm converges to a local maximum likelihood estimate \( \hat{\lambda}_{ML} \) for \( \lambda \). The EM method for mixtures is derived in [9] for one class, i.e., the special case \( M = 1 \). It is extended to the GF likelihood function in the Appendix. GF training is an iterative procedure that computes stationary points of the posterior likelihood function \( \mathcal{L} \) without taking gradients or derivatives. It begins with an initial guess, say \( \lambda \), for the optimum parameters, and each iteration gives a new parameter estimate, say \( \lambda^+ \), that is guaranteed to increase the value of the posterior likelihood function \( \mathcal{L} \) unless \( \lambda \) is a stationary value of \( \mathcal{L} \), in which case \( \lambda \equiv \lambda^+ \). Consequently, if the GF algorithm iterates are bounded above, as they typically are in applications (see the Theorem in the Appendix), the GF training algorithm must converge to a stationary point. In practice, stationary points of \( \mathcal{L} \) are also points of local maxima. By restarting the GF training algorithm with different initial guesses \( \lambda \) and choosing the best of the local maxima so obtained, a satisfactory maximum likelihood estimate for \( \lambda \) can be found.

An intuitive understanding of the shape of the decision (discriminant) surface can be gained in certain instances. Consider the two class problem. The FLD always has a linear decision boundary, as remarked above, but GF training will not result in linear decision boundaries in general. The nonlinear decision boundary will be very flat (linear) wherever the input vector \( X \) lies “close” to only one component in each of the two classes. The reason is that the log-likelihood ratio behaves locally like the FLD in this case. Intuitively, then, the GF decision surface comprises several nearly flat sections that are joined together by smoothly varying transitional surfaces. This intuitive image suggests that GF training may be robust against overtraining on the sample set \( T \) when \( G \) is in some sense small compared to the size of the training set \( T \). The image also suggests a “decision directed” method for obtaining piecewise linear discriminants, and this is discussed briefly in Section VII.

Finally, GF training has an important translation property that shows clearly that GF training is based on PDF estimation and not on class discrimination. To be explicit, for \( j = 1, \ldots, M \), let \( \Psi_j \) denote the training set \( T_j \) after translation.
by a given vector \( \phi_j \), and let \( \Psi \) denote the union of all sets \( \Psi_j \). Suppose GF training applied to the translated training set \( \Psi \) converges to the parameter set \( \lambda_\Psi \), and that GF training applied to the set \( T \) converges to the parameter set \( \lambda_T \). Then the parameter sets \( \lambda_\Psi \) and \( \lambda_T \) are translates, that is, they are identical, except that the mean vector in \( \lambda_\Psi \) of the \( i \)th component in the \( j \)th mixture is \( \mu_{ij} + \phi_j \), where \( \mu_{ij} \) is the corresponding mean vector in \( \lambda_T \). This result assumes that the initial parameter sets are also translates.

It follows from the translation property that the estimated class conditional PDF’s are independent of the between-class sample separations. Classification performance of GF training is therefore determined by two independent factors: 1) the separation of the class means, and 2) the detailed shape of the individual class conditional PDF’s.

IV. TRAINING A PRIORI CLASS PROBABILITIES AND MISCLASSIFICATION COSTS

The GF training algorithm gives explicit maximum likelihood estimates for the class a priori probabilities \( \{ \alpha_j \} \) without iteration. From equation (A.63), the maximum likelihood estimate of \( \alpha_j \) is \( \hat{\alpha}_j = \#(T_j)/\#(T) \), or, in words, \( \hat{\alpha}_j \) represents the relative abundance of class \( j \) in the training set \( T \). Clearly, the training set \( T \) contains significant a priori class probability information only if it has been carefully compiled.

Standard statistical practice requires screening the training set to eliminate outliers and other anomalies. Moreover, it may also be necessary to screen the training set to ensure correctly labelled samples. If careful attention is not given to these important tasks, the resulting training set \( T \) will contain little or no meaningful information regarding the class a priori probabilities. In this case, the likelihood function \( \mathcal{L} \) must be modified slightly. It becomes

\[
\log \mathcal{L}(T|\lambda \setminus \{ \alpha_j \}) = \sum_{j=1}^{M} \sum_{x \in T_j} \log g_j(x|\lambda_j)
\]

where \( \lambda \setminus \{ \alpha_j \} \) denotes the parameter set \( \lambda \) with \( \{ \alpha_j \} \) removed.

It is straightforward to show, using the methods of the Appendix, that the GF training algorithm modified for this likelihood function is identical to equations (A.64)–(A.67). The sole difference is that \( \{ \alpha_j \} \) are no longer estimated. We will refer to both algorithms as GF training algorithms.

The preceding comments should not obscure the fact that it is still necessary to train class a priori probabilities in applications in which these quantities cannot be estimated from the training set \( T \). Although one may resort to information theoretic concepts such as entropy [10], a more appropriate recourse for many applications is to exploit a priori information not immediately available from within the training set \( T \), as it is defined in Section I. The same is true as well concerning the misclassification costs \( \{ c_{ji} \} \). Training these fundamental quantities is important if near-optimum classification performance is to be attained in practice.

The essential difficulty is that the likelihood function \( \mathcal{L}(T|\lambda \setminus \{ \alpha_j \}) \) is independent of the misclassification costs \( \{ c_{ji} \} \) and the probabilities \( \{ \alpha_j \} \). No maximum likelihood
training algorithm can estimate factors missing from the fundamental likelihood structure. Thus, \( \mathcal{L} \) must be modified to include dependence on a priori information. This task requires intimate knowledge of the particular application together, perhaps, with additional observation time history. Although time history can be included in \( T \), it is clear that training \( \{e_{ij}\} \) and \( \{\alpha_j\} \) may require an extensive modification of \( \mathcal{L}(T|\lambda \setminus \{\alpha_j\}) \) and involve information and methods outside the scope of the present paper.

V. GF TRAINING EXAMPLES

Three examples are presented to illustrate the effectiveness of GF training on different size training sets. To focus clearly on the small training set problem, the same classes are used in all the examples, and training is performed on different size subsets of the available training set \( T \). The effect of using different subsets of \( T \) on classification performance is assessed in Section VI.

Each example comprises three classes defined on \( \mathbb{R}^2 \), that is, the dimension of the input vector \( X \) is \( N = 2 \). The samples in \( T \) are measured data, not simulated. Because of the way \( T \) was gathered and screened in the application, the relative abundance of sample data for each class does not reflect the a priori class probabilities \( \{\alpha_j\} \). The class prior probabilities \( \alpha_j \) are chosen equal to 1/3. The misclassification costs \( c_{ij} \) are defined by equation (2.4) and are chosen equal to 0 if \( l = j \) and equal to 1/2 if \( l \neq j \). The optimum classifier is, with these choices, equivalent to the well known maximum likelihood classifier.

Given model orders \( \{G_j\} \), GF training is defined for these examples by equations (A.64)-(A.67). The best choice of \( G_j \), the number of (bivariate) Gaussian components in the mixture PDF for class \( j \), is a model order selection problem, and its solution is application dependent. Typically, \( G_j \) should be chosen as small as possible without losing classification performance. The study of the order selection problem is greatly facilitated by the numerical robustness of GF training; however, this important problem is outside the scope of the present paper. Overtraining is the only aspect of this problem upon which we will comment.

GF training requires initial values for the class mixture parameters. For class \( j \), the initial mixing proportions \( \pi_{ij} \) were chosen equal to \( 1/G_j \). The initial covariance matrix was the within-class scatter matrix (c.f., (3.3)) for two classes) for the training set. The initial mean vectors for Example 1 were chosen randomly within a square containing the appropriate class samples. The initial mean vectors for Example 2 were a subset of those for Example 1, and Example 2 contained those for Example 3. No effort was made to restart GF training to determine if the local maximum likelihood solutions obtained were globally optimum.

To facilitate the discussion of the examples, we define the decision risk by

\[
\rho(X) = \min \{\rho_1(X), \rho_2(X), \rho_3(X)\},
\]

where the decision risks \( \{\rho_j(X)\} \) are approximated by the right hand side of (2.4). The decision assurance is defined by
where the estimated class PDF’s \( g_j(X) \) are defined by equation (2.2). The optimum class decision is identified by the index \( j^* \), where
\[
j^* = \arg \min \{ \rho_1(X), \rho_2(X), \rho_3(X) \}.
\]
Because of the particular choice of costs and class priors, we also have the equivalent expression \( j^* = \arg \max \{ g_1(X), g_2(X), g_3(X) \} \).

GF training is very efficient in the examples presented in detail below. The convergence criteria required a relative increase of \( 10^{-4} \) in the log-likelihood function, that is, iteration ceased when the current value of \( \log \mathcal{L} \) increased by a factor less than or equal to \( 1+10^{-4} \) times the previous value of \( \log \mathcal{L} \). GF training for Example 1 converged in 53 iterations and used approximately two minutes of wall-clock time (including all file handling and I/O operations). Example 2 converged in 26 iterations in about five seconds, while Example 3 converged in 18 iterations in well under one second. The GF algorithm is implemented in single precision FORTRAN on a Sun SPARCstation I.

**Example 1: Large-Size Training Set, \( T \):** The training set \( T = T_1 \cup T_2 \cup T_3 \) comprises 1960, 720, and 500 two-dimensional training samples in classes 1, 2, and 3, respectively. Example 1 trains using all the available samples in \( T \) with the choices \( G_1 = 8, G_2 = 4 \), and \( G_3 = 2 \) for the number of class mixture components. This choice for \( \{ G_j \} \) reflects the relative diffuseness (as compared to an uncorrelated Gaussian distribution) of the training samples in the various classes. The trained mixing proportions and mean vectors of the class mixture PDF’s are listed in Table I. The inverse of the trained covariance matrix is
\[
\Sigma^{-1} = 10^{-3} \times \begin{bmatrix} 17.590 & -0.76978 \\ -0.76978 & 6.0786 \end{bmatrix}.
\]

The eigenvalues of \( \Sigma \) are 56.6850 and 165.911, so \( \Sigma \) is numerically stable.

**Fig. 2** is a scatter plot of the training set \( T \), together with a graph of the three-class discriminant function obtained after GF training is completed. The graph of the discriminant function is the boundary line between the regions of the input plane that map into the three different classes under the optimum decision rule (5.3). Inspection of Fig. 2 indicates that good generalization of the training set \( T \) has taken place. Overtraining has not occurred, since overtraining is characterized by highly convoluted discriminant curves and the nonlinear discriminant curves in Fig. 2 are smooth.

Figs. 3, 4, and 5 depict the level curves, or contours, of the estimated class PDF’s. The likelihood levels corresponding to the contours in these and subsequent figures are given in decibels referenced to the maximum PDF level, or dB/max. Generally, to plot a function in dB/max, it is normalized by its maximum value, and then 10 times its base 10 logarithm is taken. Thus, \(-20 \) dB/max is equivalent to a level that is \( 10^{-2} \) times the referenced maximum PDF level. The use of decibels is justified by the dynamic range of the likelihood functions involved. The maximum values of the class PDF’s are \( 4.29 \times 10^{-4}, 7.91 \times 10^{-4} \), and \( 16.41 \times 10^{-4} \) for classes 1, 2, and 3, respectively. Class 3 has the largest maximum because it has the most compact PDF.

The large \( \times \) marked on Figs. 3–5 is the approximate location of the point of maximum likelihood. The locations of the mean vectors of the trained Gaussian components are marked with squares. Note that in Class 1 the maximum likelihood does not occur at the mean vector of one of the Gaussian components. Scattered plots of the training data have been superimposed on the contours in Figs. 3–5. Sample data are marked with simple dots. In each class virtually all the training data lies within the \(-20 \) dB/max contour. Since the \(-20 \) dB/max contours of the classes intersect and their maxima do not greatly differ, perfect separation of the three classes is not achieved.

Although two Gaussians were permitted for class 3, GF training merged them by superimposing their means, as is seen from Table I. Merging indicates that class 3 is overmodeled, that is, \( G_3 = 2 \) is too large. The common covariance matrix structure is easily seen in the PDF level curves for class 3, depicted in Fig. 5. Fitting a single Gaussian to only class 3 samples would give a slightly different covariance matrix. The divergence is due to pooling samples across classes, i.e., the covariance matrix \( \Sigma \) depends jointly on the entire training set \( T \), not just the samples for any one class.

The decision risk \( \rho(X) \) gives much more insight into the class structure than the simple discriminant curve alone. Fig. 6 depicts the risk \( \rho(X) \) in dB/max, where the maximum risk is \( 4.94 \times 10^{-5} \). The region of greatest decision risk occurs between classes 2 and 3, and this fact agrees very well with the good visual separation between the scatter plots of class 1 and the other two classes. The risk \( \rho(X) \) thus not only confirms, but also quantifies, our intuition in the matter. Note that the discriminant curve runs along the ridges of the graph of \( \rho(X) \).

The decision assurance \( \delta(X) \) is depicted in Fig. 7. The maximum assurance is \( 5.47 \times 10^{-4} \). The assurance function \( \delta(X) \) is useful as an indicator of the correctness of the
TABLE II

<table>
<thead>
<tr>
<th>Decision</th>
<th>Class 1</th>
<th>Class 2</th>
<th>Class 3</th>
<th>Sample size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 1</td>
<td>98.32%</td>
<td>1.67%</td>
<td>0.20%</td>
<td>1960</td>
</tr>
<tr>
<td></td>
<td>(1297)</td>
<td>(12)</td>
<td>(1)</td>
<td>1960</td>
</tr>
<tr>
<td>Class 2</td>
<td>0.97%</td>
<td>93.61%</td>
<td>0.80%</td>
<td>720</td>
</tr>
<tr>
<td></td>
<td>(19)</td>
<td>(674)</td>
<td>(4)</td>
<td>720</td>
</tr>
<tr>
<td>Class 3</td>
<td>0.71%</td>
<td>4.72%</td>
<td>99.00%</td>
<td>500</td>
</tr>
<tr>
<td></td>
<td>(14)</td>
<td>(34)</td>
<td>(405)</td>
<td>500</td>
</tr>
</tbody>
</table>

optimum decision. An outlier is easily classified by noting its relation to the discriminant curve, and the risk \( \rho(X) \) associated with this decision is very small, as can be seen from Fig. 6. Nonetheless, one should not feel too comfortable with any decision concerning an outlier. Screening outliers is accepted statistical practice, and screening can be facilitated by setting a threshold on the assurance \( \delta(X) \). If \( \delta(X) \) is not sufficiently large, then no classification decision is made. This is equivalent to postulating an additional "null" class having an appropriate diffuse PDF. Note that the discriminant curve runs down the valleys of the pdf graph of \( \delta(X) \).

Classification performance estimates derived from the training set are optimistically biased, as is well known. However, overtraining has not occurred, so such estimates should not in this instance be significantly biased. Classification performance estimates are given in Table II, and are presented primarily for comparison with the next two examples. Note that the largest off-diagonal entry of the confusion matrix corresponds to misclassifying class 2 samples as class 3. This also corresponds to the region of greatest risk \( \rho(X) \).

Example 2: Moderate-Size Training Set, \( T/10 \): Example 2 trains on the set \( T/10 \), a fixed (randomly selected) subset of \( T \) with 196, 72, and 50 training samples representing classes 1, 2, and 3, respectively. The choices \( G_1 = 4 \), \( G_2 = 2 \), and \( G_3 = 1 \) are made in this example to reflect the reduced training set size, given the choices of \( \{G_j\} \) in Example 1. The trained mixing proportions and mean vectors of the class mixture PDF's are listed in Table III. The inverse of the trained covariance matrix is

\[
\Sigma^{-1} = 10^{-3} \times \begin{bmatrix} 12.345 & 3.2255 \end{bmatrix}.
\]

The eigenvalues of \( \Sigma \) are 73.164 and 223.24, so \( \Sigma \) is numerically stable.

The discriminant boundary and decision risk \( \rho(X) \) are depicted in Fig. 8. The maximum risk is \( 4.23 \times 10^{-2} \). The discriminant boundary is effectively piecewise linear in this example because the reduced model orders \( \{G_j\} \) make the GF discriminant more prone to have locally flat behavior, as described above in Section III. The risk function \( \rho(X) \) has two discernable peaks (local maxima) that correspond to binary decision problems (i.e., two-class problems). There was only one peak in Example 1.

The decision assurance in dB/\text{max} is depicted in Fig. 9. The maximum assurance is \( 4.15 \times 10^{-4} \). The covariance matrix structure, discernable in the elliptically shaped likelihood contours of Fig. 9, is slightly rotated from that of Example 1, but this difference does not significantly alter the overall class likelihood distributions. It is interesting to note that the point of intersection of the three arms of the GF discriminant lies in a small valley (i.e., local minimum).

Classification performance was estimated on the full training set \( T \). The confusion matrix is given in Table IV. As is evident from Table IV, GF training on the reduced size set \( T/10 \) gives excellent classification performance. Note that class 3 is never misclassified as class 1. Correct classification rates are slightly less than those for Example 1, possibly because the larger testing set has reduced the small positive bias evident in the confusion matrix of Example 1. Note that the largest off-diagonal entry in the confusion matrix corresponds to the largest peak in the risk \( \rho(X) \), and the second largest off-diagonal entry corresponds to the second largest peak.

Example 3: Small-Size Training Set, \( T/100 \): Example 3 trains on the set \( T/100 \), a fixed (randomly selected) subset of \( T/10 \) with 20, 7, and 5 training samples representing classes 1, 2, and 3, respectively. The number of components per class are further reduced to \( G_1 = 2 \), \( G_2 = 1 \), and \( G_3 = 1 \). The trained mixing proportions and mean vectors of the class mixture PDF's are listed in Table V. The inverse of the trained covariance matrix is

\[
\Sigma^{-1} = 10^{-3} \times \begin{bmatrix} 11.230 & 4.9816 \end{bmatrix}.
\]

The eigenvalues of \( \Sigma \) are 66.010 and 204.149, so \( \Sigma \) is numerically stable.

The discriminant boundary and decision risk are depicted in Fig. 10. The three arms of the discriminant boundary are nearly
linear in this example because of the small model orders \( \{G_j\} \). The class decision regions are unbounded in this example. The class 3 region is bounded in the other two examples. The risk function \( \rho(X) \) has only one peak, and it lies in the same location as that of Example 1. The maximum decision risk is \( 8.72 \times 10^{-5} \).

The decision assurance in dB/\text{max} is depicted in Fig. 11. The maximum assurance is \( 4.57 \times 10^{-4} \). The covariance matrix structure, clearly visible in Fig. 11, shows that the covariance matrix \( \Sigma \) is significantly rotated from that of the other two examples. The reason is the small model order for class 1. It happens during training that one Gaussian models the most densely clustered samples, and the other models the most significant portion of the remaining samples in class 1. Since class 1 samples dominate the within-class covariance matrix calculation and the densely clustered samples dominate (cf. the mixing proportions of Table V) in class 1, the covariance matrix \( \Sigma \) reflects the dense portion of class 1 samples. This effect is evident in Fig. 11.

Classification performance was estimated on the sample set \( T \) that was used as the training set for Example 1. The confusion matrix is given in Table VI. Note that class 3 is never misclassified as class 1. Correct classification rates are very similar to those of Example 2. The largest off-diagonal entry in the confusion matrix corresponds to the peak of \( \rho(X) \), just as in the other examples. Clearly, GF training on the greatly reduced size training set \( T/100 \) seems to be nearly as good as that attained by using the entire training set \( T \).

### Table V

<table>
<thead>
<tr>
<th>Component number</th>
<th>Class 1</th>
<th>Class 2</th>
<th>Class 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \pi )</td>
<td>0.757665</td>
<td>106.663</td>
<td>1.00000</td>
</tr>
<tr>
<td>( \mu )</td>
<td>61.286</td>
<td>1.00000</td>
<td>48.823</td>
</tr>
<tr>
<td>( \nu )</td>
<td>71.176</td>
<td>15.857</td>
<td>49.063</td>
</tr>
<tr>
<td>( \tau )</td>
<td>106.816</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Component number</th>
<th>Class 1</th>
<th>Class 2</th>
<th>Class 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \pi )</td>
<td>0.242335</td>
<td>106.816</td>
<td></td>
</tr>
<tr>
<td>( \mu )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \nu )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \tau )</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table VI

<table>
<thead>
<tr>
<th>Decision</th>
<th>Class 1</th>
<th>Class 2</th>
<th>Class 3</th>
<th>Input</th>
<th>Training</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>97.65%</td>
<td>1.11%</td>
<td>0.00%</td>
<td>1960</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>(1914)</td>
<td>(8)</td>
<td>(0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class 2</td>
<td>0.46%</td>
<td>91.81%</td>
<td>0.80%</td>
<td>720</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>(9)</td>
<td>(661)</td>
<td>(4)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class 3</td>
<td>1.89%</td>
<td>7.08%</td>
<td>99.20%</td>
<td>500</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>(37)</td>
<td>(51)</td>
<td>(496)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

VI. TOLERANCE INTERVALS FOR ASSESSING CLASSIFICATION PERFORMANCE

Example 3 is one realization of a larger “experiment” conducted by randomly selecting subsets of specified size from
the training set $T$. In this section we assess quantitatively how representative this example was of the larger experiment by using a nonparametric statistical method known as tolerance intervals. Tolerance intervals are similar to confidence intervals in their use, but they are defined and derived very differently.

We define a training trial $Z$ on a fixed size subset of a given labelled training set $T$ in the following manner. Firstly, a (uniform) random sample $S$ of the specified size is drawn from $T$ without replacement. The subset $S$ is returned to $T$ before the next training trial begins. Next, GF training is conducted on the set $S$. The initial parameter set $\lambda$ required by GF training may be fixed or generated randomly, but the same initialization procedure must be used in all training trials.

On convergence of the GF training algorithm, classification performance is assessed on the set $T \setminus S$. In the examples presented in Section V, classification performance is measured by a confusion matrix. In this Section we also consider the total error rate.

Training trials are independent trials of a multivariate discrete random variable. The trials are independent because of the independence of the subsets $S$ drawn from $T$, and the trials are discrete outcome because there are only a finite number of different possible subsets $S$ that can be drawn from $T$. Finally, the trials are multivariate because the outcome is the calculated confusion matrix together with the total error rate. In principle, the PDF of the training trials can be found exactly by systematically running through the entire list of all possible subsets $S$ of the training set $T$; however, except for very small training sets $T$, such a procedure is computationally prohibitive.

Suppose momentarily that the total error rate, denoted by $\epsilon$, is the only outcome of a training trial and that its PDF is continuous, not discrete. Denote the PDF of $\epsilon$ by $E(\epsilon)$, and let $n$ independent training trials with outcomes $\epsilon_1, \epsilon_2, \ldots, \epsilon_n$ be given. Then the population fraction or coverage, $u$, of the PDF $E(\epsilon)$ between $\min \epsilon_k$ and $\max \epsilon_k$ is given exactly by

$$u = \int_{\min \epsilon_k}^{\max \epsilon_k} E(\epsilon) d\epsilon.$$

Wilks shows [11] that the PDF of $u$, denoted $P_n(u)$, is independent of the PDF $E(\epsilon)$ and is equal to

$$P_n(u) = n(n - 1)u^{n-2}(1 - u).$$

Robbins shows [12] that order statistics are the only statistics which yield distribution free tolerance intervals. If we want coverage $u \geq 100\beta\%$ with probability $100\alpha\%$, then $n$ must be chosen so that

$$Pr\{u \geq \beta\} = \alpha = \int_{\beta}^{1} P_n(u) du. \quad (6.1)$$

The PDF of the total error rate for a training trial is discrete. The result by Wilks is applicable to univariate discrete PDF's, but (6.7) changes to $Pr\{u \geq \beta\} \geq \alpha$ in this case. For a proof of Wilks' result for discrete PDF's, see [13].
In general, a training trial outcome includes the confusion matrix and total error rate. The PDF of the confusion matrix is discrete and multivariate. Wald [14] derives distribution free tolerance intervals for continuous multivariate outcomes by computing order statistics on each vector component separately and by carefully choosing a multidimensional interval (block). Tukey extends Wald’s results to more general choices of multidimensional blocks [15] and to discrete multivariate PDF’s [16]. The curves given in [17] are valid for continuous multivariate PDF’s. These curves may be used to obtain lower bounds on the confidence $\alpha$ for discrete multivariate PDF’s [16], just as in the univariate case.

From the curves in [17], taking $n$ equal to 50 ensures coverage 90% = 100\% and confidence 95% = 100\%α%. Results for $n = 50$ independent trials of the experiments $T/10$ and $T/100$ are shown in Table VII and in Table VIII, respectively. The corresponding tolerance intervals for the total error rate in percentages are $3.6 \pm 1.1$ and $8.4 \pm 5.3$ for $T/10$ and $T/100$, respectively. The tolerance intervals for the confusion matrices were obtained using Wald’s method [14] for selecting the multidimensional block. Table VII shows that GF training yields good class characterization when a $T/10$ training set is used. The largest tolerance intervals are for class 2 because class 2 data overlaps both class 1 and class 3 data. Table VIII shows that GF training does not perform as well on $T/100$ training sets. In particular, large performance variations are possible when trying to distinguish class 2 data from class 3 data. Table VIII clearly shows that the excellent results obtained in Example 3 are at the high end of the tolerance interval (block) for the experiment $T/100$.

### TABLE VII

<table>
<thead>
<tr>
<th>Tolerance Intervals for $T/10$ Experiment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input class</td>
</tr>
<tr>
<td>Decision (%)</td>
</tr>
<tr>
<td>Class 1</td>
</tr>
<tr>
<td>97.1 ± 1.2</td>
</tr>
<tr>
<td>Class 2</td>
</tr>
<tr>
<td>1.5 ± 0.7</td>
</tr>
<tr>
<td>Class 3</td>
</tr>
<tr>
<td>1.4 ± 0.9</td>
</tr>
</tbody>
</table>

### TABLE VIII

<table>
<thead>
<tr>
<th>Tolerance Intervals for $T/100$ Experiment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input class</td>
</tr>
<tr>
<td>Decision (%)</td>
</tr>
<tr>
<td>Class 1</td>
</tr>
<tr>
<td>96.0 ± 3.0</td>
</tr>
<tr>
<td>Class 2</td>
</tr>
<tr>
<td>1.4 ± 1.4</td>
</tr>
<tr>
<td>Class 3</td>
</tr>
<tr>
<td>2.5 ± 1.6</td>
</tr>
</tbody>
</table>

VII. CONCLUDING REMARKS

The examples of Sections V and VI give convincing evidence of the utility of GF training. Although the classes in these examples are nearly linearly separable (cf. Fig. 10 and Table VI), the important translation property discussed in Section IV implies that linear separability is not the central issue for GF training because we can always train on widely
separated translates of the individual class training sets \( T \). The examples show that GF training has obtained very reasonable estimates of the class conditional PDF’s, that is, the available class training samples have been generalized in some sense. Highly nonlinear discriminants are by-products of good class conditional PDF estimates.

A “decision directed” generalized Fisher (DDGF) method can be used, if desired, to obtain a strictly piecewise linear approximation to the GF discriminant surface. The DDGF method classifies an input vector \( X \) into the class \( j^* \) such that \( (i^*, j^*) = \arg \max \{ a_j \pi_{ijj}(X) \} \). The component decision \( i^* \) is also part of the DDGF decision. If the required maximum is taken first over the component index \( i \), and then over the class index \( j \), it is seen that the DDGF method is equivalent to a two stage decision and is implementable in a feed-forward NN structure that avoids using exponential nonlinearities. In the first stage, a “within-class” decision determines which component generated the given input vector. There are as many within-class decisions as there are classes. In the second stage, a final “between-class” decision is made using the representative class components determined by the first stage. Both the within-class and between-class decisions have piecewise linear discriminants because of the Gaussian PDF structures in each instance. The within-class mixing proportions \( \{ \pi_{ij} \} \) are the a priori probabilities for the within-class decisions, while the between-class mixing proportions \( \{ \alpha_j \} \) are used for the between-class decision.

An interesting aspect of GF training is that, because only one covariance matrix \( \Sigma \) is used across all classes, the principal components analysis (PCA) based on \( \Sigma \) is common to all \( M \) classes. A common PCA, taken together with the spread-of-the-means of the components, are potentially useful tools for investigating dimensional reduction of the feature space in all classes simultaneously. This unique aspect of GF training merits further study.

The GF training algorithm derived in the Appendix easily accommodates several useful extensions. Three extensions are mentioned here. Supervised/unsupervised GF training can be undertaken on training sets in which some of the training samples are unlabelled. Unsupervised GF training is the special case of all unlabelled training data. This extension is especially useful for applications in which the cost or difficulty of correctly labelling all the training samples is prohibitive. GF training iterations can be made adaptive and run “closed loop” if the class PDF’s are time-varying. This extension requires reformulating the GF likelihood function with Bayesian prior distributions \( [9] \) for the mixing proportions, mean vectors, and covariance matrix of the class conditional mixture PDF’s. Adaptive GF training is potentially useful in applications in which class statistics are either non-stationary or are treated as non-stationary to ensure robustness and a degree of fault tolerance. GF training can be extended to mixtures of discrete PDF’s and continuous non-Gaussian PDF’s \( [8] \). These extensions may enable reduced PNN size (because of the increased modelling efficiency) in applications requiring discrete feature
vectors or continuous non-Gaussian feature vectors. These extensions are not mutually exclusive. For example, adaptive GF training is possible with supervised/unsupervised training sets \( T \).

**APPENDIX**

**DERIVATION OF THE TRAINING ALGORITHM**

The Generalized Fisher (GF) training algorithm is based on the Expectation-Maximization (EM) method described in [8]. The EM method consists of two steps: The first step is called the expectation step or E-step, and the second is called the maximization step or M-step. The E-step extends the likelihood function \( \mathcal{L} \) to the unobserved or “missing” data, and then computes an expectation of over the missing data to obtain an auxiliary function \( Q \). The M-step maximizes the function \( Q \) with respect to the parameter set to be estimated. Reference [8] describes the conditions required for the EM method to converge to a local maximum of the likelihood function \( \mathcal{L} \).

Suppose independent samples of a random vector \( X \) with dimension \( N \) are observed, where \( X \) has a mixture of mixtures conditional PDF given by

\[
X \sim \sum_{j=1}^{M} \alpha_j g_j(X|\lambda_j) \tag{A.1}
\]

where

\[
\sum_{j=1}^{M} \alpha_j = 1, \tag{A.2}
\]

\[
g_j(X|\lambda_j) = \sum_{i=1}^{G_j} \pi_{ij} p_{ij}(X|\lambda_{ij}) \tag{A.3}
\]

\[
= \sum_{i=1}^{G_j} \frac{\pi_{ij}}{(2\pi)^{N/2}|\Sigma|^{1/2}} \exp \left[ -\frac{1}{2} (X - \mu_{ij})^T \Sigma^{-1} (X - \mu_{ij}) \right] \tag{A.4}
\]

\[
\sum_{i=1}^{G_j} \pi_{ij} = 1, \tag{A.5}
\]

and the symbol \( X^T \) denotes the transpose of \( X \). The number of components \( G_j \) can be different for different class mixtures. Note that the between-class mixing proportions \( \{\alpha_j\} \) and the within-class mixing proportions \( \{\pi_{ij}\} \) are contained in the interval \([0,1]\).

The parameter sets

\[
\lambda_{ij} = \{\mu_{ij}, \Sigma\} \tag{A.6}
\]

\[
\lambda_j = \{\pi_{ij}, \lambda_{ij}\}_{i=1}^{G_j} \tag{A.7}
\]

\[
\lambda = \{\alpha_j, \lambda_j\}_{j=1}^{M} \tag{A.8}
\]

are unknown. In this appendix the GF training algorithm is derived for estimating the unknown parameter set \( \lambda \) from the training set \( T \). The following discussion develops the expectation step (E-step) and the maximization step (M-step) of the EM method applied to the mixture of mixtures PDF model in equation (A.1). The training set \( T \) is partitioned (labelled) so that for each component \( g_j(X|\lambda_j) \) of the mixture \( f(X|\lambda) \), \( T_j \) of the observations of \( X \) are from \( g_j(X|\lambda_j) \):

\[
T = \{X_n\}_{n=1}^{T} = \left\{ \left( X_{kj} \right)_{k=1}^{T_j} \right\}_{j=1}^{M} \tag{A.9}
\]

where

\[
\sum_{j=1}^{M} T_j = T. \tag{A.10}
\]

The posterior likelihood function for the unlabelled training set \( T \) is

\[
\mathcal{L}_u(T|\lambda) = \left( \prod_{j=1}^{T} \prod_{k=1}^{T_j} \sum_{l=1}^{M} \alpha_j g_j(X_{nk}|\lambda_j) \right) \tag{A.11}
\]

using equation (A.1) and the independence of the training samples. The multinomial coefficient is required in equation (A.11) because the training set \( T \) is unordered. Since the multinomial coefficient is a constant and only scales the likelihood function, it is dropped for the rest of this discussion. The likelihood function of the labelled training set \( T \) becomes

\[
\mathcal{L}(T|\lambda) = \prod_{j=1}^{M} \prod_{k=1}^{T_j} \left( \sum_{l=1}^{M} \alpha_j g_j(X_{nk}|\lambda_l) \delta(l-j) \right) \tag{A.12}
\]

where \( \delta(\cdot) \) is the Kronecker delta function. Substituting equation (A.3) in equation (A.12) yields

\[
\mathcal{L}(T|\lambda) = \prod_{j=1}^{M} \prod_{k=1}^{T_j} \alpha_j \left( \sum_{l=1}^{G_j} \pi_{lj} p_{lj}(X_{nk}|\lambda_{lj}) \right). \tag{A.13}
\]

If \( \alpha_j = 0 \) for some \( j \), then \( \mathcal{L}(T|\lambda) = 0 \). Therefore, we require that \( \alpha_j > 0 \), \( j = 1, \ldots, M \).

The missing data in this problem is the index, \( i \), of the PDF \( p_{ij}(\cdot|\lambda_{ij}) \) within the mixture \( g_j(\cdot) \) from which \( X_{kj} \) originated. The “complete data” in this case is the set

\[
T' = \left\{ \left( Y_{kj} \right)_{k=1}^{T_j} \right\}_{j=1}^{M} = \left\{ \left( X_{kj}, i_{kj} \right)_{k=1}^{T_j} \right\}_{j=1}^{M} = T \cup \mathcal{T}, \tag{A.14}
\]

where \( i_{kj} \) denotes the component index of the PDF from which \( X_{kj} \) was drawn. Note that \( i_{kj} \) is not observed and that \( 1 \leq i_{kj} \leq G_j \). The conditional PDF for \( T' \) is

\[
\mathcal{F}(T'|\lambda) = \prod_{j=1}^{M} \prod_{k=1}^{T_j} \alpha_j g_j(Y_{nk}|\lambda_{i_{kj}})
\]

\[
= \prod_{j=1}^{M} \prod_{k=1}^{T_j} \alpha_j \left( \sum_{l=1}^{G_j} \pi_{lj} p_{lj}(Y_{nk}|\lambda_{lj}) \right)
\]

\[
= \prod_{j=1}^{M} \prod_{k=1}^{T_j} \alpha_j \left( \sum_{l=1}^{G_j} \pi_{lj} p_{lj}(X_{nk}|\lambda_{lj}) \delta(l-i_{kj}) \right)
\]

\[
= \prod_{j=1}^{M} \prod_{k=1}^{T_j} \alpha_j \pi_{ij} p_{ij}(X_{nk}|\lambda_{ij}) \tag{A.15}
\]
where \( \delta(\cdot) \) is the Kronecker delta function. The PDF of \( T = \{i_k\} \) conditioned on \( T \) and \( \lambda \) is then
\[
\mathcal{K}(I|T, \lambda) = \frac{F(T'|\lambda)}{L(T'|\lambda)}.
\]
Substituting (A.12) and (A.15) into \( \mathcal{K} \) yields
\[
\mathcal{K}(I|T, \lambda) = \prod_{j=1}^{M} \prod_{k=1}^{T_j} \frac{\pi_{ij} p_{ij}(X_{k_j} | \lambda_{ij})}{g_j(X_{k_j} | \lambda_j)} \bigg|_{I=i_{k_j}}
\]
\[
= \prod_{j=1}^{M} \prod_{k=1}^{T_j} w_{ij}(X_{k_j}) \bigg|_{I=i_{k_j}}
\]
where
\[
w_{ij}(X_{k_j}) = \frac{\pi_{ij} \exp \left[-\frac{1}{2}(X - \mu_{ij})^{T} \Sigma^{-1}(X - \mu_{ij}) \right]}{\sum_{i=1}^{G_j} \pi_{ij} \exp \left[-\frac{1}{2}(X - \mu_{ij})^{T} \Sigma^{-1}(X - \mu_{ij}) \right]} \bigg|_{X=X_{k_j}}
\]
Note that \( w_{ij}(X_{k_j}) \) \( \geq 0 \) and \( w_{ij}(X_{k_j}) = 0 \) if and only if \( \pi_{ij} = 0 \). It is straightforward to verify that
\[
\sum_{I} \mathcal{K}(I|T, \lambda) = 1,
\]
where the sum over \( I \) is shorthand for the T-fold sum
\[
\sum_{I} = \sum_{i_1=1}^{G_1} \cdots \sum_{i_{T_j}=1}^{G_1} \cdots \sum_{i_{T_j}=1}^{G_1}
\]
\[
\cdots \sum_{i_{T_M}=1}^{G_1} \cdots \sum_{i_{T_M}=1}^{G_1}
\]
Similarly,
\[
\sum_{I \setminus i_{k_j}} \mathcal{K}(I|\lambda) = w_{ij}(X_{k_j}) \bigg|_{I=i_{k_j}}
\]
where the sum over \( I \setminus i_{k_j} \) is the same as the sum over \( I \) except that the sum over the index \( i_{k_j} \) is deleted. Note that \( \mathcal{K}(\cdot) \) defines a probability on the discrete space of indices \( I \).

The E-step of the EM method is defined to be
\[
Q(\lambda|\lambda') = E \{ \log F(T'|\lambda') | T, \lambda' \},
\]
where the expectation is taken over the set of all possible indices \( I = i_{k_j} \) and is conditioned on \( T \) and \( \lambda' \), where \( \lambda' \) is a given parameter set of the form (A.8). By definition of expectation,
\[
Q(\lambda|\lambda') = \sum_{I} \log [ F(T'|\lambda) ] \mathcal{K}(I|T, \lambda) + \sum_{j=1}^{M} \sum_{k=1}^{T_j} \log [ \pi_{ij} p_{ij}(X_{k_j} | \lambda_{ij}) ] \bigg|_{I=i_{k_j}} \mathcal{K}(I|T, \lambda')
\]
\[
= \sum_{j=1}^{M} \sum_{k=1}^{T_j} \log [ \pi_{ij} p_{ij}(X_{k_j} | \lambda_{ij}) ] \bigg|_{I=i_{k_j}} \mathcal{K}(I|T, \lambda')
\]
\[
= \sum_{j=1}^{M} \sum_{k=1}^{T_j} \log [ \pi_{ij} p_{ij}(X_{k_j} | \lambda_{ij}) ] w_{ij}(X_{k_j})
\]

The first term simplifies easily using equation (A.20) because
\[
\sum_{j=1}^{M} \sum_{k=1}^{T_j} \log [ \pi_{ij} p_{ij}(X_{k_j} | \lambda_{ij}) ] w_{ij}(X_{k_j})
\]
\[
= \left( \sum_{j=1}^{M} T_j \log \alpha_j \right) \sum_{I \setminus i_{k_j}} \mathcal{K}(I|T, \lambda')
\]
\[
= \sum_{j=1}^{M} T_j \log \alpha_j.
\]

Using equation (A.22), the second term in equation (A.26) simplifies to
\[
\sum_{I \setminus i_{k_j}} \sum_{j=1}^{M} \sum_{k=1}^{T_j} \log [ \pi_{ij} p_{ij}(X_{k_j} | \lambda_{ij}) ] \mathcal{K}(I|T, \lambda')
\]
\[
= \sum_{j=1}^{M} T_j \sum_{k=1}^{G_j} \log [ \pi_{ij} p_{ij}(X_{k_j} | \lambda_{ij}) ] w_{ij}(X_{k_j}) \bigg|_{I=i_{k_j}}
\]
\[
= \sum_{j=1}^{M} \sum_{k=1}^{T_j} \log [ \pi_{ij} p_{ij}(X_{k_j} | \lambda_{ij}) ] w_{ij}(X_{k_j})
\]
where
\[
w_{ij}(X_{k_j}) = \frac{\pi_{ij} \exp \left[-\frac{1}{2}(X - \mu_{ij})^{T} \Sigma^{-1}(X - \mu_{ij}) \right]}{\sum_{i=1}^{G_j} \pi_{ij} \exp \left[-\frac{1}{2}(X - \mu_{ij})^{T} \Sigma^{-1}(X - \mu_{ij}) \right]} \bigg|_{X=X_{k_j}}
\]
Therefore, from equations (A.26), (A.28) and (A.30)
\[
Q(\lambda|\lambda') = \sum_{j=1}^{M} T_j \log \alpha_j
\]
\[
+ \sum_{j=1}^{M} \sum_{k=1}^{T_j} \log [ \pi_{ij} p_{ij}(X_{k_j} | \lambda_{ij}) ] w_{ij}(X_{k_j}).
\]

Note that if \( \pi_{ij} = 0 \) for some \( i \) and \( j \), then \( Q(\lambda'|\lambda') = -\infty \). The M-step of the EM method maximizes \( Q(\lambda'|\lambda') \) with respect to the parameter set \( \lambda \) given the previous estimate \( \lambda' \). In (8), it is proved that maximizing \( Q \) is equivalent to maximizing \( \mathcal{L} \); hence, an iterative procedure for maximizing \( Q \) over the parameter set \( \lambda \) will cause the likelihood function \( \mathcal{L} \) to monotonically increase. This maximization problem is solved either by differentiating \( Q \) with respect to each parameter in the set \( \lambda \), setting the resulting partial derivatives equal to zero, and then solving for each parameter, or by the method of Lagrange multipliers if parameter constraints are necessary.
In the following development, it turns out that the parameters \( \{\alpha, \pi, \mu\} \) are uncoupled from each other and from \( \Sigma \) in the equations defining the necessary conditions for maximization, so they can be solved for separately. The parameter \( \Sigma \) is a function of the parameters \( \{\mu_{ij}\} \).

The expression for \( Q \) in equation (A.32) may be rewritten as

\[
Q(\lambda|x) = \sum_{j=1}^{M} T_j \log \alpha_j + \sum_{j=1}^{M} \sum_{i=1}^{G_j} T_j \log \pi_{ij}(X_{kj}) \log \pi_{ij}
+ \sum_{j=1}^{M} \sum_{i=1}^{G_j} \sum_{k=1}^{T_j} \log \pi_{ij}(X_{kj}) \log \pi_{ij}(X_{kj})
= Q_\alpha + \sum_{j=1}^{M} Q^*_j + Q_p,
\]

where \( Q_\alpha \) depends only on \( \{\alpha_j\} \), \( Q^*_j \) depends only on \( \{\pi_{ij}, i = 1, \ldots, G_j\} \), and \( Q_p \) depends jointly on \( \Sigma \) and the vectors \( \{\mu_{ij}, i = 1, \ldots, G_j, j = 1, \ldots, M\} \). In addition, to maximize \( Q^*_j \) for each \( j \), we require that \( \pi_{ij} > 0 \) for all \( i \). Consequently, if \( Q_\alpha, Q^*_j \) and \( Q_p \) are each maximized separately, then \( Q \) is also maximized (see Juang [18] for the special case of \( M = 1 \)).

Starting with the parameter set \( \{\alpha_j\} \), \( Q_\alpha \) is maximized subject to constraint (A.2). The appropriate Lagrangian for \( \{\alpha_j\} \) is

\[
\tilde{Q} \equiv Q_\alpha + \gamma \left( \sum_{j=1}^{M} \alpha_j - 1 \right),
\]

where \( \gamma \) is the Lagrange multiplier. Differentiating with respect to \( \alpha_j \) yields

\[
\frac{\partial \tilde{Q}}{\partial \alpha_j} = \frac{T_j}{\alpha_j} + \gamma = 0;
\]

hence,

\[
\alpha_j = -\frac{T_j}{\gamma}.
\]

Substituting into the constraint (A.2) yields

\[
1 = -\sum_{j=1}^{M} \frac{T_j}{\gamma} = -\frac{1}{\gamma} \sum_{j=1}^{M} T_j = -\frac{T}{\gamma}.
\]

Therefore, \( \gamma = -T \), and

\[
\hat{\alpha}_j = \frac{T_j}{T}.
\]

Note that the estimates \( \{\hat{\alpha}_j\} \) are an immediate consequence of the labelling (partitioning) of the training set \( T \) and that \( \hat{\alpha}_j > 0 \), as required. By Lemma 2 in [19], \( \{\hat{\alpha}_j\} \) is the unique global maximum of \( Q_\alpha \).

To estimate \( \{\pi_{ij}, i = 1, \ldots, G_j\} \), the term \( Q^*_j \) is maximized subject to the constraint (A.5). In this case, the appropriate Lagrangian is

\[
\tilde{Q} = Q^*_j + \gamma \left( \sum_{i=1}^{G_j} \pi_{ij} - 1 \right),
\]

where \( \gamma \) is the Lagrange multiplier. Taking the partial derivative with respect to \( \pi_{ij} \) yields

\[
\frac{\partial \tilde{Q}}{\partial \pi_{ij}} = \frac{T_j}{\pi_{ij}} \left( \frac{1}{\pi_{ij}} + \gamma \right) = 0
\]

or

\[
\pi_{ij} = -\frac{T_j}{\gamma} \sum_{k=1}^{G_j} w_{ij}(X_{kj}).
\]

Substituting (A.42) into constraint equation (A.5) results in

\[
-\frac{1}{\gamma} \sum_{i=1}^{G_j} \sum_{k=1}^{T_j} w_{ij}(X_{kj}) = 1.
\]

Since

\[
\sum_{i=1}^{G_j} \sum_{k=1}^{T_j} w_{ij}(X_{kj}) = 1,
\]

it follows that \( \gamma = -T_j \). Hence, the estimate of the mixing proportion is, from equation (A.42),

\[
\hat{\pi}_{ij} = \frac{T_j}{T} \sum_{k=1}^{T_j} w_{ij}(X_{kj}).
\]

Note that \( \hat{\pi}_{ij} > 0 \), as required. Again, by Lemma 2 of [19], the estimate \( \{\hat{\pi}_{ij}\} \) is the unique global maximum of \( Q^*_j \).

The new estimate of the covariance matrix \( \Sigma \) is found by differentiating \( Q_p \) with respect to \( \Sigma = [\Sigma_{ij}] \):

\[
\nabla_\Sigma Q_p = \sum_{j=1}^{M} \sum_{k=1}^{G_j} \frac{T_j}{2} \left[ -\Sigma^{-1} + \Sigma^{-1}(X_{kj} - \mu_{ij})(X_{kj} - \mu_{ij})^T \Sigma^{-1} \right] = 0,
\]

where \( \nabla_\Sigma \) is defined as the matrix operator

\[
\nabla_\Sigma = \left[ \frac{\partial}{\partial \Sigma_{ij}} \right]^{N}_{j=1,i=1}.
\]

From equation (A.31),

\[
\nabla_\Sigma Q_p = \sum_{j=1}^{M} \sum_{i=1}^{G_j} \sum_{k=1}^{T_j} w_{ij}(X_{kj}) = \sum_{j=1}^{M} \sum_{i=1}^{G_j} T_j = T,
\]

and the estimate of the covariance matrix is therefore

\[
\hat{\Sigma} = \frac{1}{T} \sum_{j=1}^{M} \sum_{i=1}^{G_j} \sum_{k=1}^{T_j} w_{ij}(X_{kj})(X_{kj} - \mu_{ij})(X_{kj} - \mu_{ij})^T.
\]

Note that \( \hat{\Sigma} \) is in the convex hull of outer products of vectors, and that the mean vectors \( \mu_{ij} \) in equation (A.49) are the maximum likelihood estimates of the true means.

The estimate of the mean vector \( \mu_{ij} \) is also found by differentiating \( Q_p \):

\[
\nabla_{\mu_{ij}} Q_p = \sum_{k=1}^{T_j} w_{ij}(X_{kj}) \Sigma^{-1}(X_{kj} - \mu_{ij}) = 0.
\]
Note that in this case, $\mu_{ij}$ is defined as a vector of length $N$, and that $\nabla_{\mu}$ is defined for general vectors $\mu = (\mu_1, \ldots, \mu_N)$ as
\[
\nabla_{\mu} \equiv \left[ \frac{\partial}{\partial \mu_j} \right]_{j=1}^{N}.
\]

Hence, from equation (A.50) the estimate of the mean vector $\hat{\mu}_{ij}$ is
\[
\hat{\mu}_{ij} = \frac{m \sum_{k=1}^{T_i} w_{ij}^k(X_{kj}) X_{kj}}{\sum_{k=1}^{T_i} w_{ij}^k(X_{kj})},
\]

(A.52)

Note that the estimate $\hat{\mu}_{ij}$ is in the convex hull of the training set for label class $j$, and that $\hat{\mu}_{ij}$ is independent of the covariance matrix estimate $\hat{\Sigma}$.

Equations (A.49) and (A.52) produce the unique global maximum of $Q_p$, provided a regularity condition is imposed on the training set $T$. For vectors $c_j \in \mathbb{R}^N, j = 1, \ldots, M$, define the set
\[
\Gamma(c_1, \ldots, c_M) = \{ X_j + c_j \}_{j=1}^{M} \subset \mathbb{R}^N,
\]

(A.53)

where $\Gamma[c]$ denotes the closed convex hull. The training set $T$ is defined to be “full rank” if the set $\Gamma(c_1, \ldots, c_M)$ has at least $N+1$ extreme points for every choice of the vectors $\{c_j\}$. The full rank assumption on $T$ guarantees that the estimated covariance matrix $\hat{\Sigma}$ given by equation (A.49) is positive definite.

The following theorem proves that each iteration step of the EM method is well defined for GF training. The theorem does not imply convergence of the GF training algorithm to a global maximum likelihood parameter estimate.

**Theorem:** If the training set $T$ is full rank, and if the mixing proportions $\pi_j \neq 0$ for all $i$ and $j$, then $Q_p$ has a unique global maximum as a function of the covariance matrix $\Sigma$ and the mean vectors $\{\mu_{ij}, i = 1, \ldots, G_j, j = 1, \ldots, M\}$.

Before proving this theorem, note that the full rank requirement on $T$ is not very restrictive. For example, if any one class has $N+1$ training samples, and after arbitrary translation, any $N$ of the translated samples are linearly independent, then the pooled training set $T$ is full rank. However, the set $T$ can also have full rank even if none of the individual classes in the pool have full rank. This is an important consequence of pooling across classes. For instance, if $M = N$ and each class has two samples, that is, $T_j = \{X_{1j}, X_{2j}\}$, then the set $T$ is full rank if and only if the vectors $\{X_{1j} - X_{2j}\}_{j=1}^{M}$ span $\mathbb{R}^N$.

Also, note that class training sets that contain only one sample do not contribute to the rank of $T$. Clearly, the geometric condition on the convex hull embodies many equivalent algebraic statements.

The following proof of the theorem is in the spirit of Liporace’s proof of a similar result [20, Appendix]. From the definition of $Q_p$ in (A.34) and $p_{ij}(X | \lambda_{ij})$ in (A.4),

\[
Q_p = \frac{1}{2} \sum_{j=1}^{M} \sum_{i=1}^{G_j} \sum_{k=1}^{T_i} w_{ij}^k(X_{kj}) \left[ -N \log(2\pi) + \log |\Lambda| - (X_{kj} - \mu_{ij})^T \Lambda^{-1} (X_{kj} - \mu_{ij}) \right].
\]

(A.54)

where $\Lambda = \Sigma^{-1}$. Parameterizing $Q_p$ in terms of the precision matrix $\Lambda$, instead of the covariance matrix $\Sigma$, allows the development of an explicitly negative expression for the second derivative of $Q_p$ at a critical point. Let the point $\lambda = \{\lambda_1, \ldots, \lambda_{G_j}, j = 1, \ldots, M\}$ satisfy the necessary conditions (A.49) and (A.52) for a critical point of $Q_p$. Expressing $\lambda$ as a convex combination of two arbitrary points $\lambda^1$ and $\lambda^2$ in the domain of $Q_p$ such that $\lambda^1 \neq \lambda^2, \lambda^1 \neq \lambda$ and $\lambda^2 \neq \lambda$ yields
\[
\mu_{ij} = \theta \mu^1_{ij} + (1 - \theta) \mu^2_{ij}
\]

(A.55)

and
\[
\lambda^1 = \theta \lambda^1 + (1 - \theta) \lambda^2.
\]

(A.56)

Note that $\theta$ in (A.55) and (A.56) is uniquely defined, is independent of the indices $i$ and $j$, and satisfies $0 < \theta < 1$. Substituting (A.55) and (A.56) into (A.54) and differentiating twice with respect to $\theta$ yields (after tedious calculation)
\[
\frac{\partial^2 Q_p}{\partial \theta^2} = -R - \frac{1}{2} \sum_{j=1}^{M} \sum_{i=1}^{G_j} \sum_{k=1}^{T_i} w_{ij}^k(X_{kj}) \left( R_{ij} + R_{ijk} \right),
\]

(A.57)

where
\[
R = \frac{1}{2} \sum_{i=1}^{N} \left[ \frac{D^2_i}{\theta D^2_i + (1 - \theta) D^2_i} \right]^2,
\]

(A.58)

\[
R_{ij} = 2(\mu^1_{ij} - \mu^2_{ij})^T \Lambda (\mu^1_{ij} - \mu^2_{ij}),
\]

(A.59)

\[
R_{ijk} = 4(\mu^1_{ij} - \mu^2_{ij})^T \Lambda (X_{kj} - \theta \mu^1_{ij} - (1 - \theta) \mu^2_{ij}),
\]

(A.60)

$D^2_i$ and $D^2_i$ are the diagonal entries of $U^T \Lambda^2 U$ and $U^T \Lambda U$ respectively, and where $U$ is a nonsingular matrix diagonalizing $\Lambda^1$ and $\Lambda^2$ simultaneously. Note that $R \geq 0$, and that $R = 0$ if and only if $\lambda^1 = \lambda^2$. (N.B. Had we not reparameterized in terms of $\Lambda$, this term would be nonpositive). Because the training set $T$ is full rank, $\Lambda$ is positive definite; thus, the term $R_{ij} \geq 0$, and $R_{ijk} = 0$ if and only if $\mu^1_{ij} = \mu^2_{ij}$ for some $i$ and $j$. At least one of the terms $\{R, w_{ij}^k(X_{kj}) R_{ij}^2\}$ is strictly positive because $\lambda^1 \neq \lambda^2$ and because $w_{ij}^k \neq 0$ for all $i$ and $j$. This implies $w_{ij}^k(X_{kj}) > 0$. The term $R_{ijk}^2$ does not vanish, but the sum over its indices does vanish; that is,
\[
\sum_{j=1}^{M} \sum_{i=1}^{G_j} \sum_{k=1}^{T_i} w_{ij}^k(X_{kj}) R_{ijk}^2 = 0.
\]

(A.61)

because of the necessary condition (A.50) at a critical point (recall the definition of $\mu_{ij}$ at a critical point from equation (A.55)). It follows that
\[
\frac{\partial^2 Q_p}{\partial \theta^2} < 0.
\]

(A.62)
Hence, a critical point of $Q_p$ is a local maximum. Since $Q_p$ has a unique critical point, all that remains to be shown is that $Q_p$ attains its maximum. Because the training set $T$ is full rank, $Q_p$ is bounded above. But $Q_p \to -\infty$ uniformly as its defining parameter vector goes to the point at infinity, so the supremum of $Q_p$ must be a maximum (i.e., $Q_p$ attains its maximum).

To summarize the GF iteration, first note that the mixing proportions $\{q_{ij}\}$ may be computed at the beginning from equation (A.39):

$$ q_{ij} = \frac{T_j}{T}. \quad (A.63) $$

Now let $\lambda^{(n)}$ be available from the previous iteration, and define the weights

$$ w_{ij}^{(n)}(X_{kj}) = \frac{\pi_{ij}^{(n)} \exp \left[ -\frac{1}{2} (X_{kj} - \mu_{ij}^{(n)})^t (\Sigma^{(n)})^{-1} (X_{kj} - \mu_{ij}^{(n)}) \right]}{\sum_{i=1}^{G_j} \pi_{ij}^{(n)} \exp \left[ -\frac{1}{2} (X_{kj} - \mu_{ij}^{(n)})^t (\Sigma^{(n)})^{-1} (X_{kj} - \mu_{ij}^{(n)}) \right]} m, \quad (A.64) $$

The new intercomponent mixing proportions are updated using equation (A.45):

$$ \pi_{ij}^{(n+1)} = \frac{T_j}{\sum_{k=1}^{T_j} w_{ij}^{(n)}(X_{kj})}. \quad (A.65) $$

Since $\pi_{ij}^{(n+1)} = 0$ if and only if $\pi_{ij}^{(n)} = 0$, GF training can not be initialized with any zero mixing proportions. Specifically, we require that $\pi_{ij}^{(0)} \neq 0$ for all $i$ and $j$. The mean vectors are updated using equation (A.52):

$$ \mu_{ij}^{(n+1)} = \frac{\sum_{k=1}^{T_j} w_{ij}^{(n)}(X_{kj}) X_{kj}}{\sum_{k=1}^{T_j} w_{ij}^{(n)}(X_{kj})}. \quad (A.66) $$

The new covariance matrix is found from equation (A.49):

$$ \Sigma^{(n+1)} = \frac{1}{T} \sum_{j=1}^{M} \sum_{i=1}^{G_j} \sum_{k=1}^{T_j} w_{ij}^{(n)}(X_{kj}) (X_{kj} - \mu_{ij}^{(n+1)}) (X_{kj} - \mu_{ij}^{(n+1)})^t. \quad (A.67) $$

Convergence of the GF algorithm can be tested in two ways. Firstly, $Q(\lambda^{(n+1)}|\lambda^{(n)})$ can be computed using equation (A.32):

$$ Q(\lambda^{(n+1)}|\lambda^{(n)}) = \sum_{j=1}^{M} T_j \log q_{ij} - \log \left( (2\pi)^{N/2} |\Sigma^{(n+1)}|^{1/2} \right) $$

$$ + \sum_{j=1}^{M} \sum_{i=1}^{G_j} \sum_{k=1}^{T_j} w_{ij}^{(n)}(X_{kj}) \left\{ \log \pi_{ij}^{(n+1)} - \frac{1}{2} (X_{kj} - \mu_{ij}^{(n+1)})^t (\Sigma^{(n+1)})^{-1} (X_{kj} - \mu_{ij}^{(n+1)}) \right\}. \quad (A.68) $$

Then $Q(\lambda^{(n+1)}|\lambda^{(n)})$ can be compared to $Q(\lambda^{(n)}|\lambda^{(n-1)})$ to determine if the parameter estimates have stabilized. If the estimates have stabilized, then the algorithm is terminated.

Alternatively, GF training can be terminated if the likelihood function, $C(T|\lambda^{(n)})$, as a function of $n$, ceases to increase at a sufficient rate. As an aside, if the likelihood function $C(T|\lambda^{(n)})$ does not monotonically increase during GF training, then there are almost certainly errors in the code. This diagnostic property of the likelihood function has proved to be very reassuring in practice and useful during program development.

Implementation of the GF training algorithm is straightforward in many cases; however, with large training sets or with high dimensional small training sets, some attention to numerical details may be required. Firstly, updating $\Sigma$ using equation (A.67) above squares the condition number of the training data and may cause a loss of accuracy. This loss is avoided if $\Sigma$ is updated using the QR factorization of the data matrix (see chapter 9 of [21], page 9.2). The QR factorization of the data matrix formed by the stacked vectors

$$ \sqrt{w_{ij}^{(n)}(X_{kj})(X_{kj} - \mu_{ij}^{(n+1)})^t} $$

can be used throughout GF training. Secondly, computing the weights $w_{ij}^{(n)}(X_{kj})$ can cause numerical underflow problems. This is especially true when parameter initialization is poor. Underflow problems with the weights is mitigated by first computing all the Gaussian exponents in the mixture using the identity

$$ X^t \Sigma^{-1} X = ||R^{-1} X||^2_2 \quad (A.69) $$

where $R$ is the $N \times N$ upper triangular matrix obtained from the QR factorization of the data matrix (see chapter 9 of [21], page 9.10), and then finding and subtracting the minimum Gaussian exponent from all the Gaussian exponents. Scaling the exponents in this way does not affect the results because the weights are conditional probabilities (see equation (A.18)). Thresholding the scaled Gaussian exponents to prevent out of range arguments in the $\exp(\cdot)$ routine is also recommended. Thirdly, the implicit division by the updated mixing proportion, $\pi_{ij}^{(n+1)}$, in equation (A.66) may cause problems when one or more mixing proportions are small. By noting that $\pi_{ij}^{(n)}$ may be factored out of both the numerator and denominator of equation (A.66), this problem is easily avoided. Defining a new variable

$$ \phi_{ij}^{(n)}(X_{kj}) = \frac{\exp \left[ -\frac{1}{2} (X_{kj} - \mu_{ij}^{(n)})^t (\Sigma^{(n)})^{-1} (X_{kj} - \mu_{ij}^{(n)}) \right]}{\sum_{i=1}^{G_j} \pi_{ij}^{(n)} \exp \left[ -\frac{1}{2} (X_{kj} - \mu_{ij}^{(n)})^t (\Sigma^{(n)})^{-1} (X_{kj} - \mu_{ij}^{(n)}) \right]} \quad (A.70) $$

enables the equation (A.66) to be rewritten as

$$ \mu_{ij}^{(n+1)} = \frac{\sum_{k=1}^{T_j} \phi_{ij}^{(n)}(X_{kj}) X_{kj}}{\sum_{k=1}^{T_j} \phi_{ij}^{(n)}(X_{kj})}. \quad (A.71) $$

This removes the implicit division by $\pi_{ij}^{(n+1)}$. 


ACKNOWLEDGMENT

The authors thank Michael L. Rousseau of the Naval Undersea Warfare Center for pointing out the importance attached by the neural network community to pooling class sample data and for creating Fig. 1. The authors also thank the reviewers for their useful and constructive comments.

REFERENCES


Roy L. Streit (SM’84) received the B.S. degree (Honors) in physics and mathematics in 1968 from East Texas State University, the M.A. degree in mathematics in 1970 from the University of Missouri (Columbia), and the Ph.D. degree in mathematics in 1978 from the University of Rhode Island. From 1981 to 1982, he was a visiting Scholar in the Department of Operations Research at Stanford University, Stanford, California, and from 1987 to 1989 he was an exchange Scientist at the Defence Science and Technology Organisation, Adelaide, Australia. He joined the Navy Underwater Sound Laboratory, New London, Connecticut, in 1970, and he is now with the Naval Undersea Warfare Center (NUWC), Newport, Rhode Island. His previous research includes the first applications of hidden Markov models (HMM’s) to frequency line detection and tracking, diverse investigations in support of several NUWC towed array R&D programs, and neural network investigations that established general theoretical classification performance limits. His current research interests include distributed detection problems and the application of Bayesian inference networks for reasoning under uncertainty to sonar data fusion.

Tod E. Luginibuhl (S’88–M’92) earned the B.S. degree in electrical engineering from the University of Rochester in 1985, the M.S. degree in electrical engineering from the University of Connecticut in 1991, and expects to complete the Ph.D. degree in electrical engineering at the University of Connecticut in 1994. Mr. Luginibuhl joined the Naval Underwater Systems Center in New London, Connecticut in 1985, where his initial assignment was the development of passive sonar system simulation software. Subsequently, he worked on estimating individual and harmonic sets of sampled frequency modulated signals. His current research is on detection and classification of sampled, short-duration, nonstationary signals, and the application of maximum likelihood methods to estimate signal parameters.

Mr. Luginibuhl is a member of Eta Kappa Nu.