Pattern Recognition, Vol. 24, No. 12, pp. 1197–1209, 1991
Printed in Great Britain

0031-3203/91 $3.00 + .00
Pergamon Press plc
Pattern Recognition Society

ADAPTIVE MIXTURES: RECURSIVE NONPARAMETRIC PATTERN RECOGNITION

CAREY E. PRIEBE* and DAVID J. MARCHETTE
Naval Oceans Systems Center, 421, San Diego, CA 92152-5000, U.S.A.

(*Received* 31 *August* 1990; *in revised form* 14 *May* 1991; *received for publication* 22 *May* 1991)

Abstract—We develop a method of performing pattern recognition (discrimination and classification) using a recursive technique derived from mixture models, kernel estimation and stochastic approximation.

Unsupervised learning Density estimation Kernel estimator Mixture model
Stochastic approximation Recursive estimation

1. INTRODUCTION

A large number of applications require the ability to recognize patterns within data, where the character of the patterns may change with time. Example applications include remote sensing, autonomous control, and automatic target recognition in a changing environment. (Titterington *et al.*,[1] Chapter 2, gives a list of applications to which mixture models have been applied. Many of these problems, and their variants, fall into the above categories.) These applications have a common requirement: the need to recognize new entities as they enter the environment. A pattern recognition system in this type of environment must be able to change its representation of the classes dynamically in order to conform to changes in the classes themselves, as well as recognize, and develop a representation for, a new class in the environment.

The adaptive mixtures approach presented herein uses density estimation to develop decision functions for supervised and unsupervised learning. Much work in performing density estimation in supervised and unsupervised situations has been done. For the most part, this research has centered on approaches that use a great deal of a priori information about the structure of the data. In particular, parametric assumptions are often made concerning the underlying model of the data. While these approaches yield impressive results, nonparametric approaches[2,3] free of a priori assumptions can be considered more powerful due to their increased generality and therefore wider applicability. Developing a system for performing unsupervised learning nonparametrically (that is, devoid of restricting assumptions) is a daunting task. In fact, there are many instances in which no system can be assured of proper performance. For example, two classes with identical distributions cannot be identified as such based on purely unsupervised learning. Nevertheless, a nonparametric density estimation approach to unsupervised learning can, in many cases, lead to a general and powerful pattern recognition tool.

(The adaptive mixtures approach considered herein is described as nonparametric. While there is some blurring of distinction between parametric, semiparametric, and nonparametric approaches, an estimation approach which intermittently changes the list of parameters to be estimated, based on the incoming observations, and which has no a priori upper bound on this parameter list, can rightfully be called nonparametric.)

In addition to the nonparametric assumption, we also consider the problem of recursive estimation (reference (1), Chapter 6).[4,5] That is, it is assumed that, due to high data rates or time constraints, we must develop our estimates in such a way that they do not require the storage or processing of all observations to date. This also limits the ability to develop optimal estimates, but often is the only approach for a given application.

By virtue of addressing the types of applications that can be termed recursive and nonparametric, we have at once made the problem more difficult and more interesting. The recursive assumption eliminates the possibility of using iterative techniques. It is necessary, by hypothesis, to develop our estimate at time t only from our previous estimate and the newest observation. The nonparametric assumption implies that we cannot make any but the simplest assumptions about our data. Realistic restrictions on processing and memory, as might be imposed on automatic target recognition, remote sensing, and automatic control applications, in conjunction with

* Author to whom correspondence should be addressed at Naval Surface Warfare Center, K12, Dahlgren, VA 22448-5000, U.S.A.

high data rates, make such applications, and the procedure discussed herein, an important subject in pattern recognition.

In this work, we apply statistical pattern recognition concepts to the problem of recursive nonparametric pattern recognition in dynamic environments. We begin with a description of pattern recognition in this context. Adaptive mixtures, a method for performing both supervised and unsupervised learning, is then developed. Simulation results are provided to show the performance of the system for a few examples.

Similarities exist between adaptive mixtures and potential functions,[6] maximum penalized likelihood,[2,7] and reduced kernel estimators.[8] The two category problem is considered throughout, with the exception of some of the examples. The results can be extended to multi-category problems easily (e.g. successive dichotomies). In addition, univariate assumptions are made in places for clarity. The sequel will, we hope, allow for meaningful discussion of recursive nonparametric learning as well as provide a useful problem definition and approach from which to begin addressing specific applications.

2. PATTERN RECOGNITION

Learning techniques are useful in a broad class of pattern recognition problems. In this section we motivate their application to problems requiring recursive and nonparametric processing.

Let $\Omega = \{C^{(1)}, C^{(2)}, \ldots, C^{(N)}\}$ be a set of classes, or patterns. Given an observation (measurement of a set of features) $\mathbf{x}_t$, indexed by time, of an object from one of the classes $C^{(i)}$, we wish to determine which class, or pattern, is represented by the observation.

To this end, we construct an estimate of the probability density functions associated with the individual classes, and then take our decision based on their relative height. Thus we consider the density of the overall distribution to be

$$\mathbf{x}_t \sim D = \sum_{i=1}^{N} \pi^{(i)} D^{(i)},$$

were $\pi^{(i)}$ is the prior probability for the class and the $D^{(i)}$ is the density for the individual class. The system will respond with class i, where i is chosen so that

$$\pi^{(i)} D^{(i)} = \max_j \pi^{(j)} D^{(j)}.$$

For pattern recognition, we are concerned with the problem of constructing estimates of the individual class densities $D^{(i)}$, and the prior probabilities $\pi^{(i)}$.

Following Kendall,[9] we define two distinct approaches to the tasks to be performed: classification and discrimination. Discrimination can be described as a supervised task. Based on a set of observations for which the true class of origin is known (the teaching set), we wish to construct a method for assigning a new observation of unknown origin to the correct class, that is, we wish to construct the individual densities $D^{(i)}$ and probabilities $\pi^{(i)}$. Classification, on the other hand, is an inherently unsupervised task. Based on a set of observations of unknown class, one decides whether groups exist within this data set. If so, one attempts to construct a method of assigning new observations to the correct class, again constructing a decision function. This corresponds to constructing the density D from observations for which the true class is unknown, and determining a partitioning of the density into individual class densities $D^{(i)}$. This is, as one would imagine, a much more difficult (and under some conditions, impossible) task.

Much of what follows pertains to classification and discrimination. While the nature of tasks to be performed becomes more complicated as we build to the dynamic environment scenario, the requirements of our pattern recognition system S can normally be thought of as analogous to these two tasks. In general, we have available a teaching data set $\{\mathbf{x}_t\}_{t=1}^{\alpha}$ for which the true class is known and untagged observations $\{\mathbf{x}_t\}_{t=\alpha+1}^{\infty}$, for which the true class is unknown. We wish to perform discrimination based on $\{\mathbf{x}_t\}_{t=1}^{\alpha}$ and use the decision function $d(\cdot)$ derived during this process to assign a class to the observations $\{\mathbf{x}_t\}_{t=\alpha+1}^{\infty}$. We would like, if possible, to use $\{\mathbf{x}_t\}_{t=\alpha+1}^{\infty}$ to update (and improve) $d(\cdot)$. Using this data for which the true class is unknown entails unsupervised learning.

It should be noted that in the simpler case of stationary distributions, it is the case that convergent estimates, $\hat{D}^{(i)}(\mathbf{x}) \to D^{(i)}(\mathbf{x})$ as $n \to \infty$, can yield $P_d(e) \to P_B(e) = P_{\text{opt}}(e)$, that is, the probability of error approaches the Bayes optimal (see, e.g. references (10) and (11)). This adds justification to the use of density estimates in constructing decision functions.

Let us now consider the extended problem in which the total number of classes, and thus the distributions $D^{(i)}$ from which our $\mathbf{x}_t$ can be drawn, is finite but not constant over time. For simplicity of exposition, we will assume that the densities $D^{(i)}$ are stationary. We will consider Ω to be the set of all classes which appear during the operation of the classifier, with $|\Omega| = N$. Let N_t be the number of classes present at time t, that is the number of classes $C^{(i)}$ for which the class probability $\pi^{(i)}$ is nonzero. Then a new class entering the environment at time $t = \tau$ corresponds to $N_\tau = N_{\tau-1} + 1$. Let class $C^{(N_\tau)}$ enter into Ω at time $t = \tau$, and remain a member of Ω until time $t = \tau'$. Then

$$\mathbf{x}_t \sim \sum_{i=1}^{N_\tau} \pi^{(i)} D^{(i)}$$

for $t \in (\tau, \tau')$. That is to say, the observations $\mathbf{x}_t$ can be drawn from distribution $D^{(N_\tau)}$ for $\tau \le t < \tau'$. For

$$t \notin (\tau, \tau'), \mathbf{x}_t \sim \sum_{i=1}^{N_{\tau-1}} \pi^{(i)} D^{(i)},$$

and $\mathbf{x}_t$ will not be drawn from distribution $D^{(N_\tau)}$. Note that, since we are assuming the $\pi^{(i)}$ sum to 1, the proportions $\pi^{(i)}$ $(i = 1, \ldots, N_\tau)$ must be adjusted during the period of time $C^{(N_\tau)}$ is in our environment $(t \in (\tau, \tau'))$. For the simplest case, the class probabilities $\pi^{(i)}$ remain constant in the regions $t \in (0, \tau - 1]$, and $t \in [\tau, \tau')$. This corresponds to a simple kind of nonstationarity in the overall distribution **D** that can be termed a "jump" nonstationarity. A good deal of work has been done in detecting changes such as these in stochastic processes (see, e.g. references (12) and (13)).

Let us now consider additionally that the individual $D^{(i)}$ be nonstationary (that is time dependent, or drifting). Thus, $D^{(i)}$ is a function of time, and is allowed to change with time, and therefore the capability to track such a change (or drift) is necessary. This condition, together with a dynamic N_t, yields what will be termed a dynamic environment.

It should be noted that it is impossible to design a system that both recognizes when a new class has appeared and tracks the nonstationarity of existing classes, unless some assumptions are made. Either there must be some model of the densities, in order to decide if an existing class is starting to violate its model, or there must be a model of the nonstationarity, or a measure of distance from classes must be used to identify new classes as masses "far enough away" from existing classes. Since the approach taken here is a nonparametric one, we do not wish to make assumptions about the character of the class densities. Instead, we will use a measure of distance to assign a new class to points "far enough away" from existing classes, as will be described below. This assumes a measure of separation between classes, which may not be desirable in some applications. It also assumes that the drift is slow enough to distinguish between a new class and the movement of an old class.

These two assumptions are a result of our restriction to problems requiring recursive, nonparametric techniques. While it may be possible in many situations to make distributional assumptions, or assumptions about the character of the nonstationarity, or to retain a collection of data points for iterative processing, we are concerned with problems which do not allow these assumptions. Thus, though we must make some assumptions about the problem, the above seem to us to be the last restrictive within the context of pattern recognition.

3. DEVELOPMENT OF ADAPTIVE MIXTURE APPROACH

We now introduce an approach capable of performing recursive nonparametric learning in each of the categories described above: adaptive mixtures.

We will develop the adaptive mixture from density estimation techniques of finite mixture modelling and stochastic approximation (s.a.). The extension of the adaptive mixture beyond these techniques will allow for the modelling of dynamic environments. For simplicity of exposition, we will focus first on the estimation of a single density. This should be thought of as one of the class densities $D^{(i)}$.

Finite mixtures

Consider for the moment the problem of estimating the components of a Gaussian mixture. That is, we assume that our density is of the form

$$D(\mathbf{x}) = \sum_{i=1}^{n} \lambda_i \Phi(\mathbf{x}; \mu_i, \sigma_i), \tag{1}$$

where n is known, and the λ_i sum to 1 for each t. We are implicitly assuming here that the data come from a single class, and we are trying to estimate the density for that class. We wish to estimate the parameter vector θ^T which consists of λ, μ, and σ. Let us also assume for the moment that D is stationary. A standard technique for estimating the parameter vector θ^T is to maximize the (log)likelihood. We will write an estimate for $D(\mathbf{x})$ with parameter vector θ as $D(\mathbf{x}; \theta)$. Following Titterington,[14] we set

$$S(\mathbf{x}, \theta) = \frac{\partial}{\partial \theta} \log(D(\mathbf{x}; \theta)), \tag{2}$$

and use these likelihood equations to obtain the update formula

$$\hat{\theta}_{t+1} = \hat{\theta}_t + \alpha_t S(\mathbf{x}_{t+1}; \hat{\theta}_t). \tag{3}$$

This can be seen to be a gradient ascent on the loglikelihood surface, and under certain conditions on α_t and D we will have convergence to the target density (see reference (14)). In theory, α_t should be $(tI(\hat{\theta}_t))^{-1}$, where I is the Fisher information matrix, but in practice an approximation of I is used. An example of this kind of approximation formula which will be used below is the following set of recursive update equations:

$$\rho_{t+1}^{(i)} = \lambda_t^{(i)} \frac{\hat{D}_t^{(i)}(\mathbf{x}_{t+1})}{\hat{D}_t(\mathbf{x}_{t+1})} \tag{4}$$

$$\lambda_{t+1}^{(i)} = \lambda_t^{(i)} + a_{t+1}^{(i)}(\rho_{t+1}^{(i)} - \lambda_t^{(i)}) \tag{5}$$

$$\mu_{t+1}^{(i)} = \mu_t^{(i)} + a_{t+1}^{(i)}\rho_t^{(i)}(\mathbf{x}_{t+1} - \mu_t^{(i)}) \tag{6}$$

$$\sigma_{t+1}^{(i)} = \sigma_t^{(i)} + a_{t+1}^{(i)}\rho_t^{(i)}((\mathbf{x}_{t+1} - \mu_t^{(i)})(\mathbf{x}_{t+1} - \mu_t^{(i)})^T - \sigma_t^{(i)}). \tag{7}$$

We will call this "update rule" (Equations (4)–(7)) $U_t(\mathbf{x}_{t+1}; \hat{\theta}_t)$. The idea behind this update rule is to proportion the new data point out to all the components, in proportion to their respective likelihoods. The mean and covariance are then updated by this proportion. In the case of a single component, these update rules are just recursive versions of the sample mean and sample covariance calculations.

An obvious choice for $a_{t+1}^{(i)}$ (in the stationary case) is

$$\left(\sum_{j=1}^{t-1} \rho_j^{(i)}\right)^{-1}.$$

If n, the number of components in the mixture, is 1, this is just $1/(t+1)$, which is the inverse of the number of data points. In general, this can be thought of as the "number of points" used to updatw component i. If the density D is not known to be a mixture of Gaussians, however, one might still wish to use the above formulation to find an approximation to the density by such a mixture. In some sense, the kernel estimator[15] is an extreme of this point of view. Thus, one could choose m "large enough", start the estimate with some initial θ, and then recursively update the estimate using the above formula. Assuming that the density is well approximated by such a mixture (which is the case of m is large) and a reasonable initial estimate is used, this procedure will result in a good estimate of the density.

If an approximation of the density D by a mixture as above is used, the number of components, m, and an initial estimate must be chosen. It would be helpful (and in fact is essential in the nonstationary case) if the algorithm could choose m and the initial estimate recursively from the data. It is this which motivates the algorithm described below.

In order to develop an estimate of the form in Equation (1), we will use a combination of the above finite mixture modelling algorithm (the update rule) and a dynamic allocation procedure which allows the algorithm to increase the number of terms in our model if our current estimate fails to account for the current observation. That is, we will add a new term to the mixture, with mean $\mu_t = \mathbf{x}_t$ if circumstances indicate this is necessary. (It is this process which lends the process its "nonparametric" label.) Otherwise, we will update our estimate $\hat{\theta}_t$ (and hence $\hat{D}$). We will call this "create rule" $C_t(\mathbf{x}_{t+1}; \hat{\theta}_t)$, and will describe it shortly. Our s.a. procedure now becomes

$$\hat{\theta}_{t+1} = \hat{\theta}_t + [1 - P_t(\mathbf{x}_{t+1}; \hat{\theta}_t)]\, U_t(\mathbf{x}_{t+1}; \hat{\theta}_t) + P_t(\mathbf{x}_{t+1}; \hat{\theta}_t)\, C_t(\mathbf{x}_{t+1}; \hat{\theta}_t). \quad (8)$$

$P(\cdot)$ in Equation (8) is the "decision-to-add-component" function, and takes on values 1 or 0, depending on whether the decision is to add a component or not.

Assuming that the system has decided to add a component, the create rule $C_t(\cdot)$ is then, for the single-class case:

$$\mu_{t+1}^{(m+1)} = \mathbf{x}_{t+1} \quad (9)$$

$$\sigma_{t+1}^{(m+1)} = \sigma_t^0 \quad (10)$$

$$\lambda_{t+1}^{(i)} = \lambda_t^{(i)}(1 - a_t) \qquad (i = 1, \ldots, m) \quad (11)$$

$$\lambda_{t+1}^{(m)} = a_t \quad (12)$$

$$m = m + 1. \quad (13)$$

Thus, the new component is centered at the observation, given an initial covariance (which may be user-defined, or derived from the components in the neighborhood of the observation) and a small proportion. All the other proportions must be updated so that they sum to 1, but otherwise the other components are unaffected. For the multi-class modelling case, $C(\cdot)$ becomes a bit more involved. This situation will be discussed below.

In the case where the decision is made to add a component for each data point, the estimate is similar to the kernel estimator (1D densities are used for clarity, and the explicit dependence on time is indicated):

$$D_t(\mathbf{x}) = \frac{1}{t}\sum_{i=1}^{t} \Phi(\mathbf{x}; x_i, \sigma_i). \quad (14)$$

Putting this into a more standard kernel estimation notation, we have

$$D_n(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h_i} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h_i}\right), \quad (15)$$

where K is the Gaussian with mean 0 and variance 1.

$D_n(\mathbf{x})$ is the estimator considered in Wolverton and Wagner[10] and Wegman and Davies.[16] Its consistency is easily established. Thus, in this extreme case, the algorithm is consistent for reasonable choices of the system variables (in this case a, σ_t and K). It is reasonable, therefore, that since the update rule is a recursive maximum likelihood estimator,[14,17,18] and so in some sense improves the estimate between the addition of new components, that if the decision to add a component is properly chosen the overall system will be consistent. The performance of the estimator obtained by using recursive updates, as opposed to merely always adding another term, is important. The reduction in the number of terms required in the estimate is a storage and computational advantage.

The decision to add a component $P(\cdot)$ can be made in a number of ways. The simplest way is to check the Mahalanobis distance from the observation to each of the components, and if the minimum of these exceeds a threshold (called the create threshold T_C), then the point is in some sense "too far away" from the other components, and a new component should be created. Recall that the Mahalanobis distance between a point $\mathbf{x}$ and a component with mean $\mu^{(i)}$ and covariance $\sigma^{(i)}$ is defined by

$$M^{(i)}(\mathbf{x}) = (x - \mu^{(i)})^T \sigma^{(i)-1}(x - \mu^{(i)}). \quad (16)$$

(Note: this is actually the square of the Mahalanobis distance, but this is unimportant to the discussion.) Thus, if the create threshold is T_C, then we create a new component at the point $\mathbf{x}_{t+1}$ if

$$M(\mathbf{x}_{t+1}) \min_i (M^{(i)}(\mathbf{x}_{t+1})) > T_C. \quad (17)$$

Other approaches would be to create stochastically with probability inversely proportional to $M(\mathbf{x}_{t+1})$ (scaled appropriately so that the probabilities lie in the range $[0, 1]$), or use the estimated density directly rather than the individual components. The stochastic threshold T_C is used in the results described below. In this case, the "Mahalanobis distance" is scaled by the exponential: $\Lambda^{(i)}(x) = \exp(-\frac{1}{2}M^{(i)}(x))$. This is the "distance" used to compare with T_C in the sequel.

Windowing

The most common technique for modifying a recursive system to allow the estimation of a nonstationary distribution is to use a window on the observations. This amounts, in the simple case, to setting a_t to some small constant. This puts an exponential window on the data, forcing the system to always treat the newest observation with a certain amount of respect. This approach obviously precludes a system from being consistent, but consistency in modelling nonstationary distributions is generally unobtainable. Consider the general update formula mentioned above, namely

$$\hat{\theta}_{t+1} = \hat{\theta}_t + \alpha_t S(\mathbf{x}_{t+1}; \hat{\theta}_t). \tag{18}$$

The conditions placed upon α_t in order for Equation (18) to be a consistent estimator are basically (a) $\Sigma\alpha_t = \infty$, and (b) $\Sigma\alpha_t^2 < \infty$. For example, $\alpha_t = t^{-1}$. To implement a windowed estimator and address nonstationarities, consider the perturbation of Equation (18) to

$$\hat{\theta}_{t+1} = \hat{\theta}_t + \beta_t S(\mathbf{x}_{t+1}; \hat{\theta}_t), \tag{19}$$

where the β_t are such that $B > 0$ is a lower bound for β_t. Then Equation (19) is a windowed s.a. scheme suitable for modelling nonstationarity densities. Note that $\Sigma\beta_t^2 = \infty$, so consistency is unobtainable. However, this provides a window on the data, allowing the estimator to adapt to changes in the underlying density. As an example, let $\beta_t = \max\{t^{-1}, B^{-1}\}$ for constant $B > 0$.

It is important to note that asymptotic considerations, as detailed above, become moot when dealing with nonstationarities. What is of importance is the level of performance that can be expected under given conditions. For instance, what variance and bias can be expected for given window widths under stationary assumptions. This information allows one to evaluate the output of the system.

Extension to the multi-class case

The preceding discussion involved modelling a single-class distribution. For the general pattern recognition problem this is clearly insufficient. We must have a method for modelling N classes within the framework of Equation (1). Consider N separate distributions of the form (1). That is,

$$D(\mathbf{x}) = \sum_{j=1}^{N} \pi^{(j)} D^{(j)}(\mathbf{x}). \tag{20}$$

Each of the component densities will be modelled as a mixture as above. Assume that a supervised training set is available, so that an initial estimate can be made for each of the classes. Assume further that all the classes are represented in the initial training set. We can then model the individual class densities as mixtures as above. However, in the general case, where there may be classes which are not represented by the initial training set, we need a mechanism for determining whether a point belongs to an existing class, or whether a new "unknown" class should be created.

Let Λ be the scaled normal, or scaled Mahalanobis distance as above. We now require that the create function $C(\cdot)$ utilize an inclusion test $\mathbf{I}(\Lambda^{(i)}, \mathbf{x}_{t+1})$, which will be used to allow Equation (20) to develop new, unknown classes $\hat{U}^{(j)}$ recursively. (Note that, for our purposes, a decision to update ($P(\cdot) = 0$) implies no need for any consideration of inclusion: proportional update takes care of itself. $\mathbf{I}(\cdot)$ can be thought of as a "coveredness-coefficient", and is used to determine if the present observation is predicted by one of the terms in the summation (20). (This is analogous to a tail-test, with the proviso that we are testing individual terms in the mixture (20) rather than the classes.) If the model (20) fails this test for all components of all classes, and the creation of a new term is indicated by Equation (17), then the newly created term will be considered the first member of a new unknown class $\hat{U}^{(N+1)}$. In this case, $C(\cdot)$ is the same as in the single-class case. If, on the other hand, the model passes the inclusion test for the current observation for one or more classes, then the new term will be incorporated into the class(es) for which the observation passes the test. This case will be discussed further.

Specifically, we let $\mathbf{I}(\Lambda^{(i)}, \mathbf{x}_{t+1})$ be a random variable such that $\mathbf{I}(\Lambda^{(i)}, \mathbf{x}_{t+1}) = 1$ if $\Lambda^{(i)}(\mathbf{x}_{t+1}) \geq T_I$ and $\mathbf{I}(\Lambda^{(i)}, \mathbf{x}_{t+1}) = 0$ if $\Lambda^{(i)}(\mathbf{x}_{t+1}) < T_I$ for some include threshold $T_I \leq 1$. If $\Sigma_i \mathbf{I}(\Lambda^{(i)}, \mathbf{x}_{t+1}) = 0$, an "unknown class" will be created, as described above. If, on the other hand, $\Sigma_i \mathbf{I}(\Lambda^{(i)}, \mathbf{x}_{t+1})$ is nonzero, $C(\cdot)$ then is as in the single-class case (Equations (9)–(13)) with the following exception: $C_{t+1} = \Sigma_i \mathbf{I}(\Lambda^{(i)}, \mathbf{x}_{t+1})$ terms are created, one corresponding to each class $C^{(i)} \ni \mathbf{I}(\Lambda^{(i)}, \mathbf{x}_{t+1}) = 1$, each with $\lambda = \lambda = a_t(C_{t+1})$ (compare Equation (12)).

$\mathbf{I}(\Lambda^{(i)}, \mathbf{x}_t)$ attempts to recursively identify the modes or terms in the unsupervised data, and as such cannot be perfect. In implementation, it is possible to use a number of parameters to develop $\mathbf{I}(\cdot)$. In particular, a "minimum variance" parameter, σ^*, can be used to aid in making the inclusion decisions. For nonasymptotic reasons, a minimum distance

(Mahalanobis) can be useful. (Note that these parameters need not be constant over the entire feature space!) Finally, uncertainty considerations can be made. For instance, when many observations (supervised or unsupervised) have been made in a given sector of feature space, we have more confidence in our estimate. We can therefore reduce our dependence on unsupervised learning (u.l.) in these cases. This is for the stationary case. In the nonstationary case, we may indeed wish to suspend u.l. in certain instances until a "change detector" (such as in references (12) and (13)) indicates that our estimate is no longer valid. While these considerations are indeed important, they deal mainly with application-specific issues. The point of mentioning them is that they are imperative precisely because there can be no "perfect" u.l. machine!

As an aside, a *discrimination threshold* $\mathbf{T}_D$ may be beneficial for the ultimate response of the system, although this is technically unnecessary. The problem arises when no known class is "close" to the current observation. In this case (which, by the way, may or may not imply the creation of a new, unknown class) the probability that the observation originates from the closest class (in a Mahalanobis sense) can be high while the likelihood is quite low. This scenario, not uncommon in practice, can yield misleading system responses. However, a discrimination threshold $0 \leq \mathbf{T}_D$ can be implemented such that $\max_i \hat{D}^{(i)}(\mathbf{x}_t) < \mathbf{T}_D$ implies that the system response will include, along with the probabilistic ranking of the classes of origin, the proviso that the observation is either an outlier or originates from an unknown and as yet unmodelled class.

4. RESULTS

The algorithm

We now present a pseudo-code algorithm for the Adaptive Mixture Model (AMM) pattern recognition paradigm for performing both supervised and unsupervised learning recursively and nonparametrically in a potentially nonstationary environment. The simulations presented are all univariate, although the algorithm presented is not restricted to this case.

Assumptions:

(A1) Gaussian kernel.

(A2) There are (or will be) C known classes, or choose C so large that there will be no more than C known classes. That is, $\Omega = \{C^{(1)}, C^{(2)}, \ldots, C^{(C)}\}$ is the set of known classes.

(A3) Choose U so large that there will be no more than U unknown classes. That is, $\Omega' = \{U^{(C+1)}, U^{(C+2)}, \ldots, U^{(C+U)}\}$ is the set of unknown classes. (If $U = 0$, no new classes will be created.)

The model (estimate) we develop for each class i is of the form

$$\hat{f}^{(i)}(\mathbf{x}) = \sum_{j=1}^{n_i} \pi_j^{(i)} \phi(\mathbf{x}; \mu_j^{(i)}, \sigma_j^{(i)}),$$

where n_i is the number of terms in the model for class i and f is the standard normal (Gaussian) p.d.f. The overall estimate is of the form

$$\begin{aligned}\hat{f}(\mathbf{x}) &= \sum_{i=1}^{C+U} \sum_{j=1}^{n_i} \pi_j^{(i)} \phi(\mathbf{x}; \mu_j^{(i)}, \sigma_j^{(i)}) \\ &= \sum_{i=1}^{C} \sum_{j=1}^{n_i} \pi_j^{(i)} \phi(\mathbf{x}; \mu_j^{(i)}, \sigma_j^{(i)}) \\ &\quad + \sum_{i=C+1}^{C+U} \sum_{j=1}^{n_i} \pi_j^{(i)} \phi(\mathbf{x}; \mu_j^{(i)}, \sigma_j^{(i)}) \\ &= \sum_{i=1}^{C} \hat{C}^{(i)}(\mathbf{x}) + \sum_{i=C+1}^{C+U} \hat{U}^{(i)}(\mathbf{x}) \\ &= \sum_{i=1}^{C} \hat{C}^{(i)}(\mathbf{x}) + \hat{U}(\mathbf{x}),\end{aligned}$$

with the requirement that

$$\sum_{i=1}^{C+U} \sum_{j=1}^{n_i} \pi_j^{(i)} = 1.$$

The $\hat{C}^{(i)}$ represent known classes and the $\hat{U}^{(i)}$ represent unknown classes (classes the AMM has developed on its own, in the unsupervised mode). $\hat{U}$ represents the overall "unknown" distribution. Let $p_j^{(i)}$ be the "unnormalized weight". We write $\phi_j^{(i)} = \phi(\mathbf{x}; \mu_j^{(i)}, \sigma_j^{(i)})$. Let Λ be the scaled normal: $\Lambda_j^{(i)} = (2\pi)^{1/2} \det[\Sigma] \phi_j^{(i)}$. Let N be the weighted normal: $N_j^{(i)} = \pi_j^{(i)} \phi_j^{(i)}$. The input $\{x_t, c_t\}_{t=1}^{\infty}$ to the system is in the form of {observation, class} pairs, where the observation is assumed to be a univariate independent random sample and the class is an element of $\{0, 1, \ldots, C\}$. A value of 0 for class ($c_t = 0$) implies an untagged observation, while any other value implies an observation for which the true class is known and reported.

Begin

Step 0: Choose C = total number of supervised classes

Choose parameters $\sigma_0^{(i)}$ $(i = 1, \ldots, C)$, T_C, T_I, T_D.

Set $n_i = 0$ for $i = 1, \ldots, C + U$.

Choose U = total number of unsupervised classes possible.

Choose B = window size.

While input $\{x_t, c_t\}$ is available

Step 1: Classify observation x_t.

Step 2: If $c_t > 0$ Then

SupervisedLearn

Else

UnsupervisedLearn

Endif

Step 3: Normalize

EndWhile
End

Classify:
Step 1: For each class $C^{(i)} \in \Omega \cup \Omega'$ determine value

$$D^{(i)}(x_t) = \sum_{j=1}^{n_i} \pi_j^{(i)} \phi(\mathbf{x}; \mu_j^{(i)}, \sigma_j^{(i)})$$

Step 2: If max $D^{(i)}(x_t) < T_D$ Then class is unknown (no class is above discrimination threshold)
Step 3: Calculate probabilities $D^{(i)}(x_t)/D(x_t)$ and $P_{\text{"single-unknown"}}$ if desired
Step 4: Return vector of probabilities.

SupervisedLearn:
If $\max_{1 \le i \le n_{c_t}} \Lambda_i^{(c_t)}(x_t) > T_c$ Then
SupervisedUpdate
Else
SupervisedCreate

UnsupervisedLearn:
If $U = 0$ or $\max_{C^{(i)} \in \Omega \cup \Omega'} \max_{1 \le i \le n_j} \Lambda_i^{(j)}(x_t) > T_c$ Then
UnsupervisedUpdate
(same condition as SupervisedLearn, only taken over all classes)
Else
UnsupervisedCreate

SupervisedUpdate:
For each component of class c_t ($i = 1, \ldots, n_{c_t}$) update $\mu_i^{(c_t)}, \sigma_i^{(c_t)}, p_i^{(c_t)}$ proportionately (Equations (4)–(7)), incorporating window

$$\left(a_{t+1}^{(i)} = \max\left(\left(\sum_{j=1}^{t+1} \rho_j^{(i)}\right)^{-1}, B^{-1}\right)\right).$$

SupervisedCreate:
Allocate component $N_{n_{c_t+1}}^{c_t}$ to class $C^{(c_t)}$ with
$\mu_{n_{c_t+1}}^{(c_t)} = x_t$,
$\sigma_{n_{c_t+1}}^{(c_t)} = \sigma^{(c_t)}$ = weighted average (over $i = 1, \ldots, n_{c_t}$) of all $\sigma_i^{(c_t)}$
(if $n_{c_t} = 0$, $\sigma_{n_{c_t+1}}^{(c_t)} = \sigma^0$).
$n_{c_t+1} = n_{c_t} + 1$.
$p_{n_{c_t+1}}^{(c_t)} = p_{n_{c_t}}^{(c_t)} + 1$
(if $n_{c_t-} = 0$, $p_{n_{c_t+1}}^{(c_t)} = 1$).

UnsupervisedUpdate:
For all comonents, update $\mu_i^{(j)}, \sigma_i^{(j)}, p_i^{(j)}$ proportionately.
(Same as SupervisedUpdate, except over all classes, not just c_t).

UnsupervisedCreate:
If $\max_{C^{(i)} \in \Omega \cup \Omega'} \max_{1 \le i \le n_j} \Lambda_i^{(j)}(x_t) \ge T_l$ Then
For each $\hat{X}^{(j)}$ such that $\max_{1 \le i \le n_j} \Lambda_i^{(j)}(x_t) \ge T_l$,
Proportionately Allocate node $N_{n_j+1}^{(j)}$ to class $\hat{X}^{(j)}$ with
$\mu_{n_j+1}^{(j)} = x_t$,
$\sigma_{n_j+1}^{(j)} = \sigma^{(j)}$ = weighted average (over i) of all $\sigma_i^{(j)}$,
$n_{j+1} = n_j + 1$,
$p_{n_j+1}^{(j)} = p_{n_j}^{(j)} + 1$.
Else Allocate node $N_{n_j+1}^{(j)}$ to new unknown class $C^{(j)}$ with
$\mu_{n_j+1}^{(j)} = x_t$,
$v_{n_j+1}^{(j)} = \sigma^{(j)}$ = weighted average (over i and j) of all $\sigma_i^{(j)}$,
Set $n_j = 1$,
Set $p_{n_j+1}^{(j)} = 1$.

Normalize: Normalize the priors $\pi_j^{(i)}$. Choices include
(a) making the overall estimate a density, so that the sum over all classes is unity, assuming estimated priors (ONED), or
(b) making each class estimate a density, so that the sum over each class is unity, assuming equal priors (EACHD).

Simulations

We consider the four situations of interest: (1) approximation of an arbitrary probability density; (2) discrimination in a stationary environment of supervised and unsupervised data; (3) approximation of a nonstationary density; and (4) the appearance of a new class. The simulations below show the abilities and limitations of the adaptive mixtures approach. It should be noted that, while rates of convergence are not yet available for this algorithm, simulations indicate that processing requirements are similar to those for a straightforward mixture approximation, implying the practicability of the algorithm.

In the examples below, $T_C \sim U[0, 1]$, $T_l = 0.5$, $T_D = 0.0$, $\sigma_0 = \sigma_0^{(i)} = 1.0$ ($i = 1, \ldots, C$). In addition, normalization is performed via ONED.

Some explanation of T_C is needed. Instead of fixing a create threshold, a "stochastic create threshold" is used. For each point, a uniform random number is drawn to be used as the create threshold. This is equivalent to deciding to create with probability equal to $1 - \max(\Lambda_i^{(j)}(x_t))$. So the system is more likely to create a component if the data point is far from any current component, and less likely to create for points close to current components. The absence of a hard threshold allows the system to create even for points close to a current component.

Example 1: *stationary single-class distribution—example of density estimation.* The data are drawn from a mixture of two normals, $p\phi(x; \mu_1, \sigma_1) + (1 - p)\phi(x; \mu_2, \sigma_2)$, with parameters $p = 0.3$, $\mu_1 = 0.25$, $\sigma_1 = 1.25$, $\mu_2 = 2.0$, $\sigma_2 = 1.0$. (See Fig. 1(a).) Note that the components do not show up as distinct modes. In this simulation, $U =$

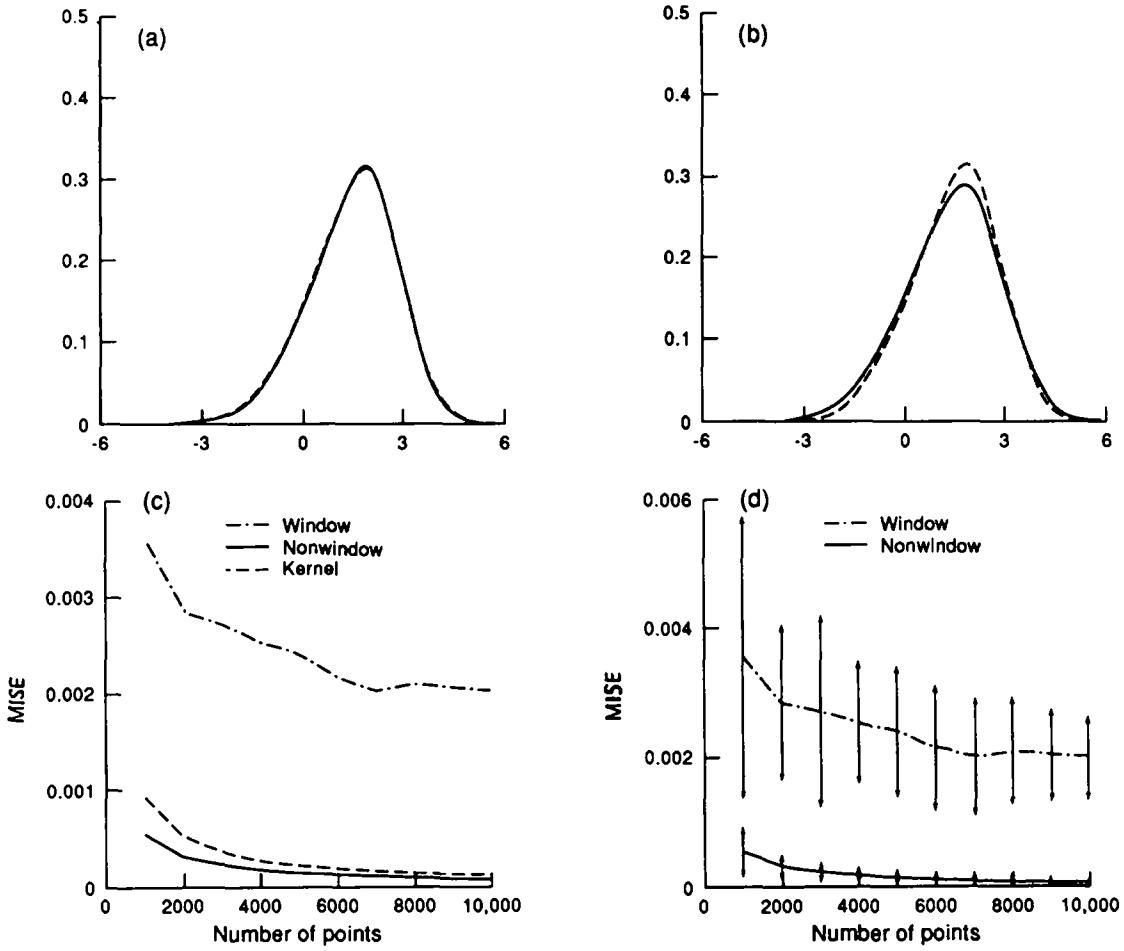

Fig. 1. Mixture of two normals: 0.3 $\phi(x; 0.25, 1.25) + 0.7\ \phi(x; 2.0, 1.0)$, $U = 0$, $C = 1$: (a) $B = \infty$. The dotted line represents the true distribution, the solid line represents the estimate after 10,000 supervised observations. 50 components were created; (b) $B = 10$. The dotted line represents the true distribution, the solid line represents the estimate after 10,000 supervised observations. 72 comonents were created; (c) mean integrate square-error (MISE). 50 runs were made and the average MISE is plotted. The solid line corresponds to $B = \infty$, the dash–dot line corresponds to $B = 10$, and the dashed line corresponds to a kernel estimator; (d) standard deviations for the MISE curves in (c). Only the adaptive mixtures are shown.

0, so no unknown classes are created, and $C = 1$, so only one class is possible. This is the problem of estimating the stationary density of a single class. It can also be thought of as supervised learning in a stationary environment.

If no distributional assumptions can be made, nor iterative processing performed, adaptive mixtures is a viable approach to density estimation. While much in the algorithm is superfluous to simple stationary density estimation, this example is presented due to its fundamental importance to the adaptive mixtures approach to pattern recognition. It must be (and is) necessary to do a credible job in basic density estimation.

Figure 1(a) shows the adaptive mixture model after 10,000 observations drawn from the distribution. The estimate, made up of 50 components, is quite good. Analysis indicates that, under suitable conditions, the single density estimation procedure is, in fact, consistent. Figure 1(b) is analogous to the first example, but with a windowed system. Here we see that, as expected, the estimate reaches a minimum-error limit, based on the window, around which it oscillates. Pattern recognition based on such estimates gain the ability to track nonstationarities (see below), but are limited in their asymptotic performance. An iterative reduced kernel estimation algorithm might produce a similar estimate with fewer terms, but under the recursive nonparametric assumptions we know of no analogous approach.

Note that a reasonably large number (10,000) of data points is used. This is typical of recursive systems: they tend to require large data sets, and are typically used in applications where the amount of data precludes the use of iterative techniques.

Figures 1(c) and (d) give mean integrated squared-error (MISE) results for the above examples. Here, 50 runs were made in which data were drawn from

the distribution, and the mean integrated squared-error was computed between the true distribution and the estimate. The average of these runs is plotted in Fig. 1(c), with comparison with a kernel estimator of the same data, and standard deviation bars for the windowed and nonwindowed AM estimators are shown in Fig. 1(d). The kernel estimator uses the bandwidth that would be optimal under the normal assumption (this is a reasonable conclusion to jump to since the data look normal, and so this is probably the bandwidth that would be chosen in practice). As expected, the windowed version has a higher error and variability, but as will be shown below, this variability allows the windowed estimator to track nonstationary distributions. This points out the trade-off between estimation of the distribution, and the ability to track nonstationary distributions.

Example 2: *stationary discrimination—supervised vs unsupervised vs fully supervised.* These data come from two classes, $C = 2$, two normals, $\phi(x; \mu_1, \sigma_1)$ and $\phi(x; \mu_2, \sigma_2)$, with parameters $\mu_1 = -3.0$, $\sigma_1 = 1.0$, $\mu_2 = 0.0$, $\sigma_2 = 1.0$. The data consist of both supervised and unsupervised points. As in Example 1 above, $U = 0$, rendering much of the algorithm unused. The basic tenet of unsupervised learning is that the model can be improved based on untagged observations. This simplified example illustrates that ability.

Figure 2(a) depicts the adaptive mixture estimate based on a small set of teaching observations, Equation (10). The estimate for the individual classes leaves something to be desired, a result of the small teaching set. Figure 2(b) shows this same model after it has been augmented by unsupervised learning. Finally, Fig. 2(c) shows the estimate produced when the entire data set is used in supervised learning. Compare Figs 2(a) and (b) with (c). We note that, while the model utilizing unsupervised learning may not be as good in general as the fully supervised estimate (in this case the unsupervised estimate is arguably as good as the supervised), the unsupervised learning has none the less improved the estimate. The adaptive mixtures procedure has (as it must) the ability to utilize information in untagged observations. It is of interest to note that the initial supervised estimate (Fig. 2(a)) uses 6 components (overall), while model Fig. 2(b), after unsupervised learning, uses 35 components. An additional 29 components were created by the algorithm to better model the observations.

Note: This performance can be improved, in this example, by using the information that the classes consist of single Gaussians. The AM can be set to allow only a single component for each class, and it will then produce (nearly) the optimal estimate (in the single normal case, the mean and covariance estimates are the sample mean and sample covariance, though the sample covariance is biased by the initial covariance). This is only true in the case of single normal distributions, of course.

Example 3: *nonstationary, single density.* The ability to follow nonstationarities in the data is examined in this example. While this example is limited to a single density, $C = 1$, with no unsupervised learning, it none the less illustrates a basic ability necessary for pattern recognition in nonstationary environments. The freedom to create a new term in a previously uncovered area of the input space when an observation is recorded in that area allows the model to drift based on the data. In addition, simple outliers will be largely ignored, due to the proportional updating of the priors, if there is little to support the hypothesis that they represent significant density mass. The window here is 10 ($B = 10$) and no unknown classes are allowed ($U = 0$).

Figure 3(a) shows the (windowed) adaptive mixtures for $D = p\phi(x; \mu_1, \sigma_1) + (1 - p)\phi(x; \mu_2, \sigma_2)$, with parameters $p = 0.6$, $\mu_1 = -1.5$, $\sigma_1 = 1.5$, $\mu_2 = 1.5$, $\sigma_2 = 1.0$, after 2000 observations (compare Fig. 1(c)). At this point, there is a jump in the distribution from which the observations are drawn, to $D' = q\phi(x; \mu_3, \sigma_3) + (1 - q)\phi(x; \mu_4, \sigma_4)$, with parameters $p = 0.3$, $\mu_3 = 0.25$, $\sigma_3 = 1.25$, $\mu_4 = 2.0$, $\sigma_4 = 1.0$. Figures 3(b)–(d) depict the drifting of the estimate to account for this jump. At Fig. 3(d) we are again approaching the minimum-error limit defined by the window size. (As noted above, work has been done in recognizing and following nonstationarities. Much of this work, however, requires a model of the nonstationarities to be encountered. Here, the window size incorporates the only assumptions made on the character of the nonstationarities.)

Example 4: *nonstationary, new class.* It is here, when a new class may enter the environment, that the need for an algorithm like adaptive mixtures can be most readily seen. The unsupervised learning part of the algorithm is finally given free reign and allowed to create new terms, allocated to new classes, based upon the data. The nonstationary abilities have been depicted above, but here we allow (based on the parameters, most notably T_C and T_I) the algorithm to indicate a new class has been recognized. In fact, numerous new classes have been recognized. Herein lies a major problem with unsupervised learning. There is no reliable way, devoid of a priori knowledge, to determine the connection between different unknown classes. In this example we consider two options: coalescing all unknown classes into a "single-unknown", and considering each unknown separately. The window width used is 10 ($B = 10$) and $U = 100$ to allow for the creation of unknown classes.

Figure 4(a) shows the two-class model after supervised learning based on observations drawn from $D_1 = \phi(x; \mu_1, \sigma_1)$, $D_2 = \phi(x; \mu_2, \sigma_2)$, with par-

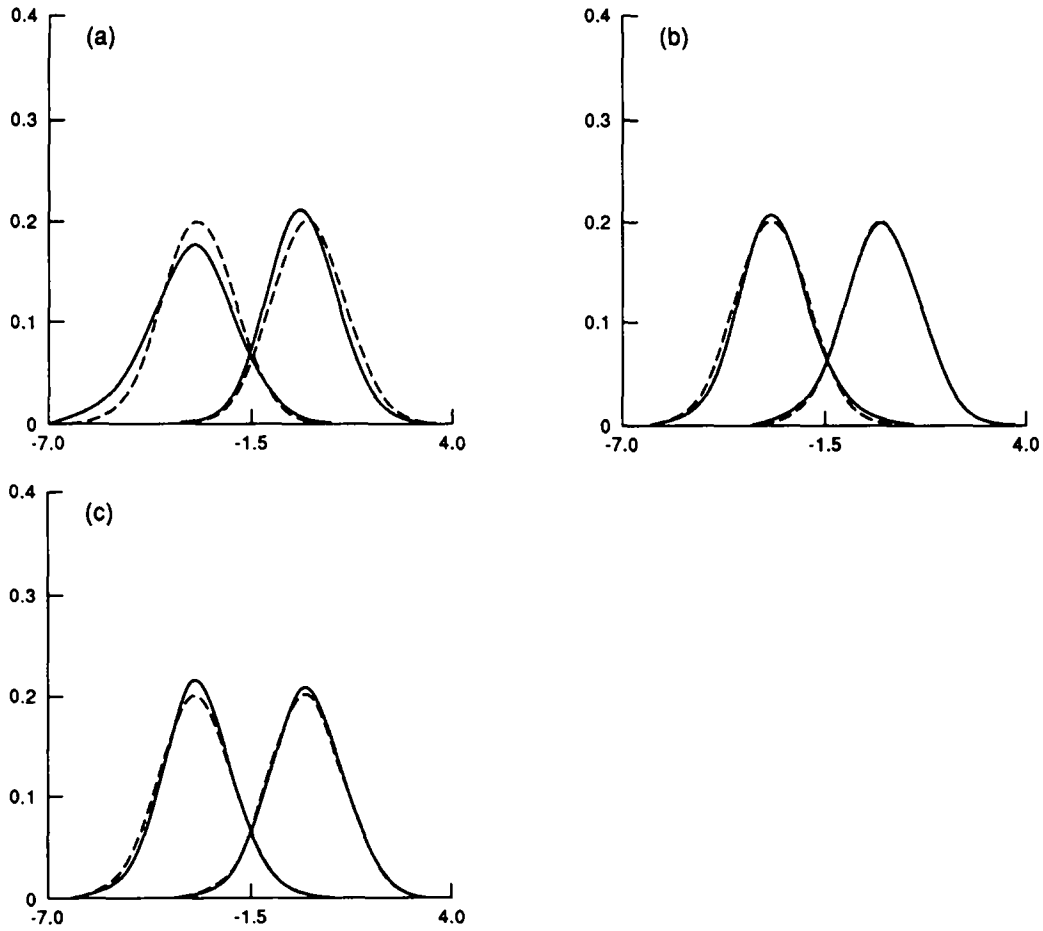

Fig. 2. Two classes ($C = 2$) from $\phi(x; -3.0, 1.0)$ and $\phi(x; 0.0, 1.0)$, $U = 0$, $C = 2$: (a) estimate after 10 supervised observations. The dotted lines correspond to the estimates, the solid lines to the true distributions. The left-most estimate consists of 4 components, the right most consists of 2; (b) the estimates from (a) after unsupervised learning on an additional 1000 observations (500 from each class). The left-most estimate consists of 18 components, the right-most consists of 17 components; (c) estimate after supervised learning on the full 1010 observations. The left-most estimate consists of 13 components, the right-most consists of 16 components.

ameters $\mu_1 = -3.0$, $\sigma_1 = 1.0$, $\mu_2 = 0.0$, $\sigma_2 = 1.0$. At this point, unsupervised learning is attempted, but the distribution from which observations are drawn jumps to the three class distribution: D_1, D_2 as above, and $D_3 = \phi(x; \mu_3, \sigma_3)$, with $\mu_3 = 5.0$, $\sigma_3 = 1.0$. Figure 4(b) depicts the estimate after unsupervised learning, with all the unknown classes shown as "single-unknown". This estimate has in fact recognized the unknown class, without significantly deggrading the model of the two known classes. Figure 4(c) shows this "single-unknown" as the combination of unknown classes that were created by the algorithm. It should be noted that coalescing these unknowns into one or more composite unknown class is impossible without prior knowledge or assumptions. Nevertheless, the model developed can allow the system to indicate that an observation is drawn, with high probability, from some class other than the two known classes.

A final note regarding Example 4: should the new class(es) be two close to existing classes, unsupervised learning techniques with no prior assumptions will be unable to distinguish the new class. A similar statement holds for Example 2: if the classes upon which unsupervised learning is being performed are too close, the results are unreliable. This will be true for any unsupervised learning algorithm. It is clear that without some prior knowledge, it is impossible to distinguish between a drifting distribution and the emergence of a new class. Thus, assumptions about the separation of classes (T_C, T_I) and the rate of change of the distributions (B) are made, and within reasonable bounds these are modifiable by the user to achieve the desired performance.

In all the examples above, emphasis is placed on the quality of the model developed. Implicitly, this translates into the quality of pattern recognition that can be performed, as a good model can be expected to yield reliable classification results. The four examples, taken as a whole, indicate the viability of

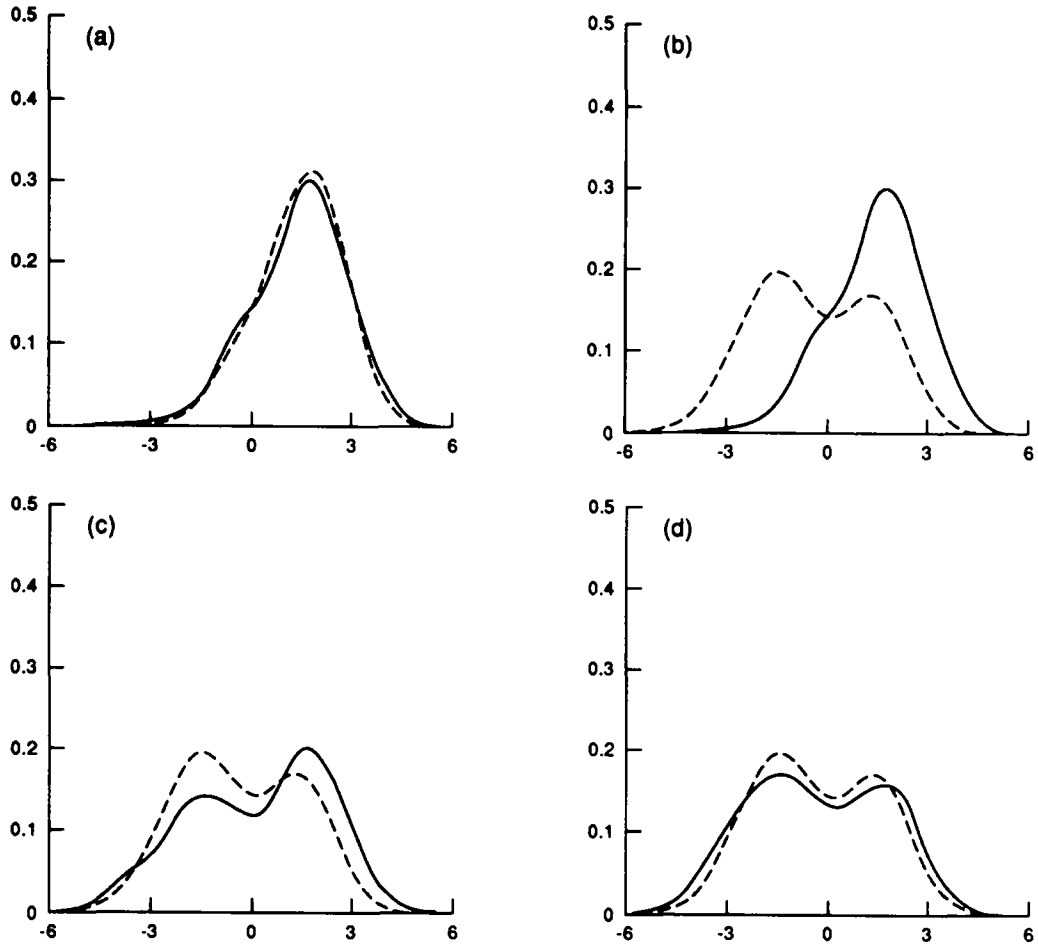

Fig. 3. Jump nonstationarity. $U = 0$, $C = 1$, $B = 10$: (a) estimate after 2000 observations from $0.6\phi(x; -1.5, 1.5) + 0.4\phi(x; 1.5, 1.0)$. Dotted line corresponds to estimate (37 components), solid line corresponds to true distribution; (b) the true distribution jumps to $0.3\phi(x; 0.25, 1.25) + 0.7\phi(x; 2.0, 1.0)$; (c) estimate after 1000 observations from the new distribution. The estimate consists of 52 components; (d) estimate after 10,000 observations from the new distributions. The estimate consists of 92 components.

using adaptive mixtures for developing the model when recursive nonparametrics are called for. The assumption of nonstationarity, and consequential selection of $B < \infty$, limits the estimation ability in the event of truly stationary data. However, for nonstationary data, this is irrelevant. The learning procedure allows the improvement of the model based on untagged (unsupervised) data, and the ability to create new terms based on the data allows both the tracking of nonstationarities and the development of new, previously unknown classes.

While comparisons between adaptive mixtures and conventional methods (such as kernel estimation) could be made for the simpler examples, no current method we know of attempts to recursively and nonparametrically perform the unsupervised learning in Example 4. The question "How well does it perform?" must therefore be answered in an *ad hoc* fashion, based on the cumulative abilities demonstrated. The basic density estimation capabilities (touched upon in Examples 1–3) together with the ability to detect new classes depicted in Example 4 indicate, we believe, the usefulness of the adaptive mixture procedure.

5. DISCUSSION

We have formulated the learning problem from a statistical pattern recognition viewpoint, motivated by developing the dynamic environment scenario, and presented an approach to both supervised and unsupervised learning which allows a system to update and improve its model of the dynamic environment based on observations for which truth is unknown. The adaptive mixtures presented herein are interesting in terms of basic density estimation as well as learning applications. While unsupervised learning can never be completely reliable, it is believed that the approach outlined above can be tailored to individual applications in such a way as

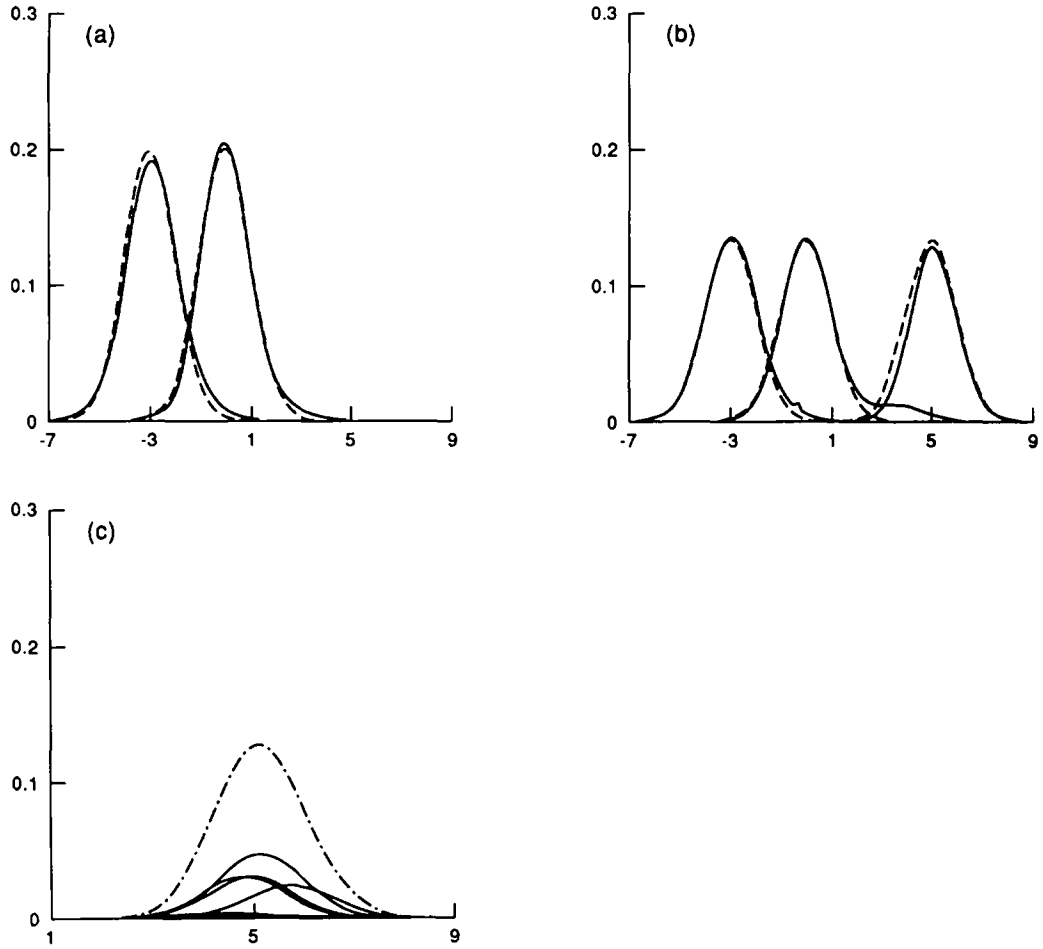

Fig. 4. $U = 100, c = 2, B = 10$: (a) estimate after 200 supervised observations (100 each) from two classes distributed as $\phi(x; -3.0, 1.0)$ and $\phi(x; 0.0, 1.0)$. The left-most estimate consists of 12 components, the right-most consists of 10 components; (b) the distribution jumps to a three class problem with the third class distributed as $\phi(x; 5.0, 1.0)$. The estimate from (b) has been taught on 10,000 unsupervised observations from the three classes. The left-most estimate consists of 27 components, the middle consists of 28 components and the right-most consists of 15 components; (c) the new estimate and its components.

to allow a new dimension in pattern recognition systems.

For the supervised case, analysis of the asymptotic performance of Equation (8) as a density estimation technique is performed in Priebe and Marchette[19]. The formal relationship of this density estimation to discrimination is the subject of ongoing work. Analysis of the extension of this work to the more general problem that we call classification (unsupervised learning) is also ongoing. While general performance statements in this area will require severe restrictions on the true densities $D^{(i)}$, we have shown that in specific instances Equation (8) is capable of the three stages of unsupervised learning necessary for dynamic environments: (i) updating density estimates $\hat{C}^{(i)}$ associated with known classes $C^{(i)}$, (ii) developing estimate $\hat{\mathbf{U}}$ for the overall density of unknown observations; and (iii) partitioning $\hat{\mathbf{U}}$ into $\hat{U}^{(j)}$ corresponding to individual unknown classes $U^{(j)}$. However, in (i) and (iii) especially, there are no guarantees, based solely on unsupervised learning, that the estimated model corresponds to the true $D^{(i)}$.

We have stressed throughout that the problem addressed here is in some sense the most difficult of all. We are attempting to do recursive nonparametric classification. Thus, we have made assumptions about the environment in which the estimation is taking place which may not hold in many applications. If information is available about the distributional families of the classes, the character of the nonstationarity (if any), or if there is time to retain samples and do distributional tests to determine, for instance, that a new class has appeared, then this information should certainly be used in the estimator. The adaptive mixtures approach shows that in the absence of this information, a creditable estimator can be produced in many situations.

6. SUMMARY

The nonparametric estimation of probability density functions allows the computation of decision regions without explicit assumptions about the character of these regions. There are many techniques for nonparametric density estimation. Some have large storage requirements (kernel estimators), some are non-smooth (histograms), and many are not recursive in nature. We have developed a recursive, nonparametric method for performing density estimation derived from mixture models, kernel estimation and stochastic approximation. This technique, called adaptive mixtures, requires less storage than kernel estimators, yet does not make the distributional assumptions about the density that mixture models do.

The adaptive mixtures estimator recursively fits a mixture of Gaussian densities to the data. This mixture is not constrained to be of a given size, and in fact the number of components of the mixture is allowed to grow with the data. Thus the estimator is truly nonparametric.

The asymptotic performance of the adaptive mixtures estimator has been investigated using the method of sieves. The estimator has been shown to be consistent for a large class of densities.

REFERENCES

1. D. M. Titterington, A. F. M. Smith and U. E. Makov, *Statistical Analysis of Finite Mixture Distributions*. Wiley, New York (1985).
2. R. A. Tapia and J. R. Thompson, *Nonparametric Probability Density Estimation*. Johns Hopkins University Press, Baltimore (1978).
3. B. L. S. Prakasa Rao, *Nonparametric Functional Estimation*. Academic Press, Orlando (1983).
4. M. B. Nevel'son and R. Z. Has'minskii, *Stochastic Approximation and Recursive Estimation*, Translations of Mathematical Monographs, Vol. 47. American Mathematical Society, RI (1973).
5. T. Y. Young and T. W. Calvert, *Classification, Estimation and Pattern Recognition*. Elsevier, New York (1974).
6. M. A. Aizerman, E. M. Braverman and L. T. Rozonoer, The probability problem of pattern recognition learning and the method of potential functions, *Automn remote Control* **26**, 1175–1190 (1966).
7. I. J. Good and R. A. Gaskins, Nonparametric roughness penalties for probability densities, *Biometrika* **58**, 255–277 (1971).
8. K. Fukunaga and R. R. Hayes, The reduced parzen classifier, *IEEE Trans. Pattern Anal. Mach. Intell.* **II**, 423–425 (1989).
9. M. G. Kendall, Discrimination and classification, *Multivariate Analysis: Proceedings of an International Symposium held in Dayton, Ohio, June 14–19*, P. R. Krishnaiah, ed. Academic Press, New York (1966).
10. C. T. Wolverton and T. J. Wagner, Asymptotically optimal discriminant functions for pattern classification, *IEEE Trans. Info. Theory* **IT-15**, 258–265 (1969).
11. W. Greblicki and M. Pawlak, Almost sure convergence of classification procedures using Hermite series density estimates, *Pattern Recognition Lett.* **2**, 13–17 (1986).
12. R. K. Bansal and P. Papantoni-Kazakos, An algorithm for detecting a change in a stochastic process, *IEEE Trans. Info. Theory* **IT-32**, 227–235 (1986).
13. R. K. Bansal and P. Papantoni-Kazakos, Outlier-resistant algorithms for detecting a change in a stochastic process, *IEEE Trans. Info. Theory* **IT-35**, 521–535 (1989).
14. D. M. Titterington, Recursive parameter estimation using incomplete data, *J. R. Statist. Soc., Ser. B* **46**, 257–267 (1984).
15. B. W. Silverman, *Density Estimation for Statistics and Data Analysis*. Chapman & Hall, London (1986).
16. E. J. Wegman and H. I. Davies, Remarks on some recursive estimators of a probability density, *Ann. Statist.* **7**, 316–327 (1979).
17. A. P. Dempster, N. M. Laird and D. B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, *J. R. Statist. Soc., Ser. B* **39**, 1–38 (1977).
18. R. A. Redner and H. F. Walker, Mixture densities, maximum likelihood and the EM algorithm, *SIAM Rev.* **26**(2) (1984).
19. C. E. Priebe and D. J. Marchette, Adaptive mixture density estimation, *Pattern Recognition* (forthcoming).

About the Author—CAREY E. PRIEBE received his B.S. degree in Mathematics from Purdue University in 1984 and his M.S. degree in Computer Science from San Diego State University in 1988. Since 1985, Mr Priebe has been working in adaptive systems and recursive estimators for the Naval Ocean Systems Center, San Diego.

About the Author—DAVID J. MARCHETTE received his B.S. and M.S. degrees in Mathematics from the University of California, San Diego, in 1980 and 1982, respectively. Since 1985, Mr Marchette has been working for the Naval Ocean Systems Center, San Diego, in the field of pattern recognition.