TESTING FOR ACCEPTANCE–REJECTION

Acceptance-rejection testing, or *hypothesis testing,* deals with the problems of taking measurements and then estimating in which of a finite number of states an underlying system resides. Over the years, numerous theories and algorithms for hypothesis testing have been proposed and studied, since the problems they intend to solve often play a vital role in many scientific or engineering fields. The following are examples where hypothesis testing has been successfully applied:

- Image classification or segmentation
- Object or person recognition in computer vision systems
- Vector quantization for low data-rate systems
- Analog information decoding or equalization in digital communication systems
- Speech recognition
- Sonar or radar signal detection
- Resonance detection in physical systems

Since many textbooks and journals have discussed hypothesis testing in many aspects, the intention of this article is not to give another general survey of this widely studied topic. Instead, the discussion will focus on the applications of data classification. After giving a basic understanding of hypothesis testing, an efficient and systematic way to build up a state-of-the art testing scheme for classification applications will be provided. The article is organized in the following way: the fundamental theory for hypothesis testing, the Bayes decision theory, will be illustrated in the next section. After Bayes theory is introduced, the discussion of how to build up a Bayesian classifier is in order. The section Statistical Modeling will address this issue by describing a recently very popular model, the probabilistic modular network. Methods and algorithms for realizing this classification model, that is, *statistical model selection, model parameter estimation,* and *parameter modification for minimizing classification error,* will be discussed in the following sections. In the final section of this article will be presented a face recognition system, for the purpose of showing how the proposed technique is applied to real applications.

BAYES DECISION THEORY: AN EXAMPLE

Bayes decision theory is a fundamental statistical approach to the problem of hypothesis testing. This approach poses the decision problem in probabilistic terms, assuming that all the involved probability values are available. To illustrate some of the types of problems to be addressed, consider the following imaginary and somewhat whimsical example: Suppose that a supermarket wants to automatically pick up misplaced oranges from a pile of apples. A system to perform this very specific task might well have the form of the following: the camera takes a picture of the fruit and passes the picture on to a *feature extractor,* whose purpose is to reduce the data by measuring certain “features” or “properties” that distinguish pictures of oranges from pictures of apples. These features (or, more precisely, the values of these features) are then passed to a *classifier,* which evaluates the evidence presented and makes a final decision about the fruit type.

Now suppose the system designer chooses color information as features. To be more specific, the feature extractor takes the average of pixel values in the picture over red, green, and blue channels, and forms a three-dimensional vector $\boldsymbol{x} = [x_r, x_g, x_b]^T$. Moreover, after taking several pictures of oranges and apples in the supermarket, the designer finds out that, while the red and blue values for both apples and oranges are similar (high red values and low blue values), the green values for oranges tends to be higher than the ones for apples. therefore, the designer makes the feature extractor to send only the green values to the classifier. Now what the classifier sees is a one-dimensional feature space $x = x_g$.

Our purpose now is to partition the feature space into two regions, where all the points in one region correspond to orange, and all points in the other correspond to apple. Since now the feature space is only one-dimensional, one might classify the fruit to orange if the green channel value x exceeds a certain threshold T, and to apple if the value is below T. To choose T, one can take pictures all the oranges and apples in the supermarket, and inspect the result.

While this rule appears to do a good job of separating fruits in the store, one has no guarantee that it will perform as well on new samples. It would certainly be prudent to obtain some more samples and see how many are correctly classified. This suggests that the problem has a statistical component, and that perhaps one should look for a classification procedure that minimizes the probability of error, or, if some errors are more costly than others, the average cost of errors. Using the decision-theoretic terminology, one might say that, as each piece of fruit emerges, nature is in one or the other of the two possible states: either it is an apple or it is an orange. Let ω denote the *state of nature,* with $\omega = \omega_0$ for apple and $\omega = \omega_1$ for orange. Because the state of nature is so unpredictable, consider ω to be random variable.

If there are more apples than oranges, one might say that, in the next picture, it is more likely to be an apple than an orange. More generally, assume that there is some *a priori* probability $P(\omega_0)$ that the next one is an apple, and some a priori probability $P(\omega_1)$ that it is an orange. These a priori probabilities reflect prior knowledge of how likely one is to see ash or birch before the lumber actually appears. In this example, it goes without saying that $P(\omega_0)$ and $P(\omega_1)$ are nonnegative and sum to one. However, a priori probabilities can be generalized to negative value (1).

Suppose for a moment that one was forced to make a decision about the type of fruit that will appear next without being allowed to see it. The only information one is allowed to use is the value of the a priori probabilities. If a decision must be made with so little information, it seems reasonable to use the following *decision rule:* Decide ω_0 if $P(\omega_0) > P(\omega_1)$; otherwise decide ω_1.

This may seem like a strange procedure, in that one always makes the same decision, even though one knows that both types of fruit will appear. How well it works depends upon the values of the a priori probabilities. If $P(\omega_1)$ is very much greater than $P(\omega_0)$, the decision in favor of ω_1 will be right most of the time. If $P(\omega_1) = P(\omega_0)$, one has only a fifty-fifty chance of being right. In general, the probability of error is the smaller of $P(\omega_1)$ and $P(\omega_0)$, and it shall be seen later that, under these conditions, no other decision rule can yield a smaller probability of error.

In most circumstances, one is not asked to make decisions with so little evidence. In the example, one can use the green color measurement x as evidence. Different samples of fruit will yield different green color readings, and it is natural to express this variability in probabilistic terms; one considers x to be a continuous random variable, whose distribution depends on the state of nature. Let $p(x|\omega_j)$ be the *state-conditional probability density* function for x, the probability density function for x given that the state of nature is ω_j. Then the difference between the mean of $p(x|\omega_0)$ and that of $p(x|\omega_1)$ describes the average difference in brightness between apples and oranges.

Suppose both the a priori probabilities $P(\omega_j)$ and the conditional densities $p(x|\omega_j)$ are known. Suppose further that one measures the average green color value in the fruit picture and discover the value of x. How does this measurement influence one's attitude concerning the true state of nature? The answer to this question is provided by *Bayes Rule:*

$$P(\omega_j|x) = \frac{p(x|\omega_j)P(\omega_j)}{p(x)} \tag{1}$$

where

$$p(x) = \sum_{j=1}^{2} p(x|\omega_j)P(\omega_j) \tag{2}$$

Bayes rule shows how observing the value of x changes the a priori probability $P(\omega_j)$ to the *a posteriori* probability $P(\omega_j|x)$. If one has an observation x, for which $P(\omega_1|x)$ is greater than $P(\omega_0|x)$, one would be naturally inclined to decide that the true state of nature is ω_1. To justify this procedure, calculate the probability of error whenever one makes a decision. Whenever one observes a particular x,

$$\begin{aligned} P(\text{error}|x) &= P(\omega_1|x) \qquad \text{if one decides } \omega_0 \\ P(\text{error}|x) &= P(\omega_0|x) \qquad \text{if one decides } \omega_1 \end{aligned} \tag{3}$$

Clearly, in every instance in which one observes the same value for x, one can minimize the probability of error by deciding ω_1 if $P(\omega_1|x) > P(\omega_0|x)$, and ω_0 if $P(\omega_0|x) > P(\omega_1|x)$.

$$\text{Decide } \omega_1 \text{ if } P(\omega_1|x) > P(\omega_0|x); \qquad \text{otherwise decide } \omega_0 \tag{4}$$

The above is the *Bayes decision rule.* Figure 1 illustrates the decision boundary generated by Bayes rule. Bayes rule can easily be extended to handle cases with more than two states of nature; if there are M states ω_j, $j \in 1, 2, \ldots, M$,

$$\text{Decide } \omega_i \text{ if } P(\omega_i|x) > P(\omega_j|x); \qquad \forall j \neq i \tag{5}$$

Note that $p(x)$ in Eq. (1) is unimportant, as far as making a decision is concerned. It is basically just a scaling factor that assures that $P(\omega_1|x) + P(\omega_0|x) = 1$. By eliminating this scaling factor, one obtains the following completely equivalent decision rule:

$$\text{Decide } \omega_1 \text{ if } p(x|\omega_1)P(\omega_1) > p(x|\omega_0)P(\omega_0); \qquad \text{otherwise decide } \omega_0 \tag{6}$$

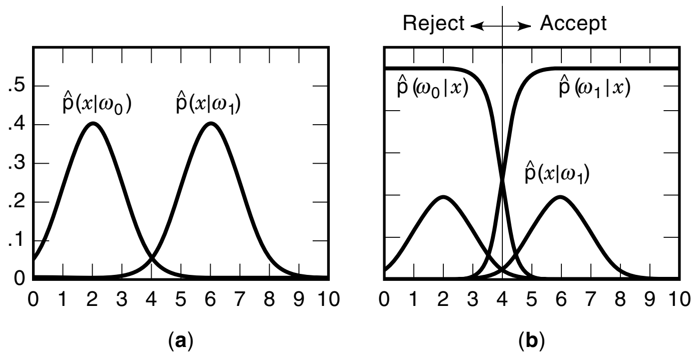

Figure 1. The estimated probability distributions of hypothesis ω_0 ($\hat{p}(\mathbf{x}|\omega_0)$) and hypothesis ω_1 ($\hat{p}(\mathbf{x}|\omega_1)$). The x axis indicates the normalized green color intensity value ranging from 0 to 10. Here $\hat{p}(\mathbf{x}|\omega_0) = N(2; 1)$ and $\hat{p}(\mathbf{x}|\omega_1) = N(6; 1)$. (b) The decision boundary (the thick straight line at $\mathbf{x} = 4$) generated by estimated posterior probabilities $\hat{P}(\omega_0|\mathbf{x})$ and $\hat{P}(\omega_1|\mathbf{x})$. Here, assume that the prior probabilities $P(\omega_0) = P(\omega_1)$. The input pattern is classified as $\omega 1$ if $\hat{P}(\omega_1|\mathbf{x}) \geq \hat{P}(\omega_0|\mathbf{x})$; otherwise it is classified to ω_0.

Some additional insight can be obtained by considering a few special cases. If for some x, $p(x|\omega_1) = p(x|\omega_0)$, then that particular observation gives no information about the state of nature; in this case, the decision hinges entirely upon the a priori probabilities. On the other hand, if $P(\omega_1) = P(\omega_0)$, then the states of nature are equally likely a priori; in this case the decision is based entirely on $p(x|\omega_j)$, the *likelihood* of ω_j with respect to x. In general, both of these factors are important in making a decision, and the Bayes decision rule combines them to achieve the minimum probability of error.

STATISTICAL MODELING

Now that it is known that posterior probability (or, more generally, any monotonically increasing functions of posterior probability) is theoretically the best candidate to serve as the discriminant function for classifying data points, the next task is to estimate that probability that lies under the system. Over the past years many parametric or nonparametric methods have been proposed and utilized to perform this task (2–7). Recently, a method called *finite mixture distributions* or *probabilistic modular networks* has been reported to have considerable success in data quantification and classification (3,5,6,8,9). Finite mixture distribution model assumes that the data points x_i in a database come from M classes $\{\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_r, \ldots, \boldsymbol{\omega}_M\}$, and the data distribution of each class consists of K_r clusters $\{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_k, \ldots, \boldsymbol{\theta}_{K_r}\}$, where $\boldsymbol{\omega}_r$ is the model parameter vector of class r, and $\boldsymbol{\theta}_k$ is the kernel parameter vector of cluster k within class r. It further assumes that, in the training data set (which should be a representative subset of the whole database), each data point has a one-to-one correspondence to one of the classes, denoted by its class label l_{ir}^{*}, defining a supervised learning task, but the true memberships of the data to the local clusters are unknown, defining an unsupervised learning task.

For the model of local class distribution, since the true cluster membership for each data point is unknown, one can treat cluster labels of the data as random variables, denoted by l_{ik} (1). By introducing a probability measure of a multinomial distribution with an unknown parameter π_k to reflect the distribution of the number of data points in each cluster, the relevant (sufficient) statistics are the conditional statistics for each cluster and the number of data points in each cluster. The class conditional likelihood density for any data point inside the class r, that is, the standard finite mixture distribution (SFMD), can be obtained by writing down the joint probability density of the x_i and l_{ik}, and then summing it over all possible outcomes of l_{ik}, as a sum of the following general form:

$$p(\boldsymbol{u}|\boldsymbol{\omega_r}) = \sum_{k=1}^{K_r} \pi_k g(\boldsymbol{u}|\boldsymbol{\theta}_k) \tag{7}$$

where $\pi_k = P(\boldsymbol{\theta}_k|\boldsymbol{\omega}_r)$ with a summation equal to one, and $g(\boldsymbol{u}|\boldsymbol{\theta}_k)$ is the kernel function of the local cluster distribution. Several observations are worth reiterating: (1) all data points in a class are identically distributed from a mixture distribution; (2) the SFMD model uses the probability measure of data memberships to the clusters in the formulation instead of realizing the true cluster label for each data point.

In the finite mixture distribution model, the Bayesian prior $P(\boldsymbol{\omega}_r)$ in Eq. (1) is an intrinsically known parameter and can be easily estimated by $P(\boldsymbol{\omega}_r) = \sum_{i=1}^{N} l_{ir}^{*}/N$, since defining a supervised learning requires information of l_{ir}^{*}. Therefore, the only uncertainty comes from class likelihood function $p(\boldsymbol{u}|\boldsymbol{\omega}_r)$, which should be the key issue in the follow-on learning process. For simplicity, in the following context, omit class index r in the discussion, when only single class distribution model is concerned, and use $\boldsymbol{\theta}$ to denote the parameter vector of regional parameter set $\{(\pi_k, \boldsymbol{\theta}_k)\}$.

METHODS AND ALGORITHMS

There are mainly two issues in the design of the finite mixture distribution model: what is the proper statistical model (i.e., number of kernels, the shape of the kernel), and how to estimate the parameters in the model. These two issues will be addressed in the following sections. The motivation of selecting the proper statistical model is driven by various objectives and requirements in the real applications. For example, in the application of medical image quantification, the structure of the disease patterns for a particular patient or for a particular type of cancer may be arbitrarily complex and, moreover, the prior knowledge on the true database structure is generally unknown, that is, the number and the kernel shape of the local clusters are not available beforehand. In such cases, statistical model selection is required and particularly critical in the procedure of data classification (4). Statistical model selection will be discussed in the section Statistical Model Selection. Once the model is selected, one can apply parametric estimation technique to obtain cluster parameters. The section Model Parameter Estimation will describe several estimation approaches. Sometimes the estimated parameters, although they reach optimal values in the information theoretic sense, do not generate a satisfactory classification result, due to the reason of insufficient training samples or the nonperfectly selected statistical model. For such cases, the parameters in the classifier need to be further fine-tuned, so that a better classification result and generalization performance can be achieved. The section Parameter Modification for Minimizing Classification Errors will address this issue.

STATISTICAL MODEL SELECTION

One conventional approach for doing statistical model selection is to use a sequence of hypothesis tests (7,10,17). The problem in this approach, however, is the subjective judgement in the selection of the threshold for different tests. Recently there has been a great deal of interest in using information-theoretic criteria, such as Akaike information criterion (AIC) (10) and minimum description length (MDL) (11) to solve this problem. The major thrust of these approaches has been the formulation of a model-fitting procedure, in which an optimal model is selected from the several competing candidates, such that the selected model best fits the observed data. For example, AIC considers the number of local clusters as an adjustable parameter. While maximizing the likelihood function, AIC will penalize the models that contain too many clusters. From a quite different point of view, MDL reformulates the problem explicitly as an information coding problem, in which the best model fit is measured, such that it assigns high probabilities to the observed data, while at the same time the model itself is not too complex to describe. Different from AIC, the penalty term in MDL not only has a term for number of clusters, but it also takes into account the number of observed samples (11). The drawbacks of MDL and AIC are that: (a) the justifications for the optimality of these two criteria, with respect to data quantification or classification, are somewhat indirect and remain unresolved (8); and (b) none of these approaches have directly addressed the problem of kernel shape learning (4).

Wang et al. (12) present another formulation of the information-theoretic criterion, minimum conditional bias/variance (MCBV) criterion, to solve model selection problem. The approach has a simple optimal appeal, in that it selects a minimum conditional bias and variance model, that is, *if two models are about equally likely, MCBV selects the one whose parameters can be estimated with the smallest variance.* The formulation is based on the fundamental argument that the value of the structural parameter cannot be arbitrary or infinite, because such an estimate might be said to have low 'bias,' but the price to be paid is high 'variance' (13). Inspired by the joint entropy of observations $\boldsymbol{x}$ and model parameter estimate $\hat{\boldsymbol{\theta}}$, the MCBV criterion is defined as

$$MCBV(K) = -\log(\mathscr{L}(\boldsymbol{x}|\hat{\boldsymbol{\theta}}_{ML})) + \sum_{k=1}^{K} H(\hat{\boldsymbol{\theta}}_{kML}) \tag{8}$$

where the subscript ML represents maximum likelihood and K is the structual parameter indicating the number of clusters. The log-likelihood $-\log(\mathscr{L}(\boldsymbol{x}|\hat{\boldsymbol{\theta}}))$ in the first term is the conditional bias, and the entropy of the parameter estimate $\sum_{k=1}^{K} H(\hat{\boldsymbol{\theta}}_k)$ is served as the conditional variance of the model. As both two terms represent natural estimation errors about their true models and should be treated on an equal basis, a minimization leads to the following characterization of the optimum estimation:

$$K_0 = \arg\{\min_{1\le K\le K_{MAX}} MCBV(K)\} \tag{9}$$

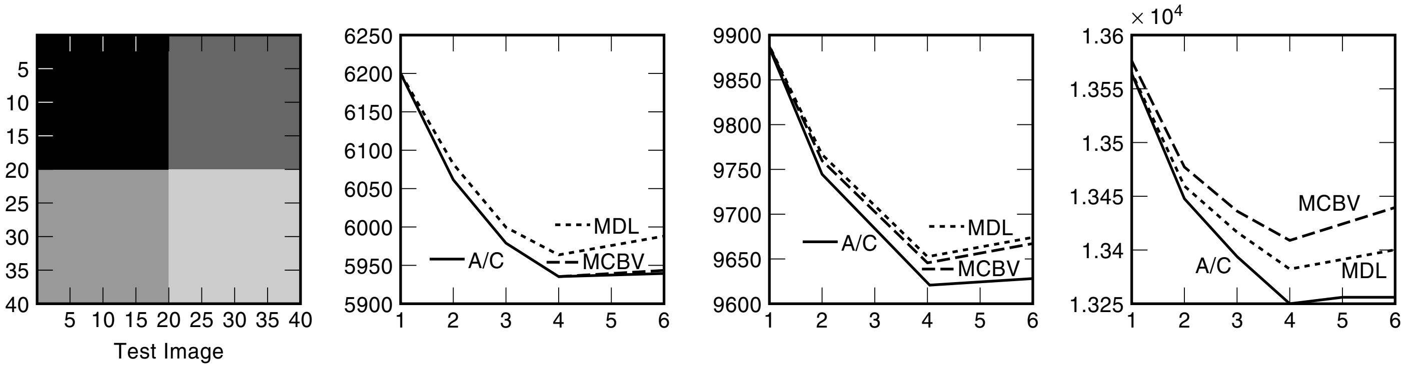

Figure 2. Original test image ($K_0 = 4$, SNR = 10 dB) and the AIC/MDL/MCBV curves in model selection (left to right: σ = 3, 30, 300). (Courtesy: Wang et al., Data Mapping by Probabilistic Modular Networks and Information Theoretic Criteria, *IEEE Trans. Signal Processing* (12)).

That is, if the cost of model variance is defined as the entropy of parameter estimates, the cost of adding new parameters to the model must be balanced by the reduction they permit in the ideal code length for the reconstruction error (the first term). A practical MCBV formulation with code-length expression is further given by

$$MCBV(K) = -\log(\mathscr{L}(\boldsymbol{x}|\hat{\boldsymbol{\theta}}_{ML})) + \sum_{k=1}^{K} \frac{1}{2}\log 2\pi e Var(\hat{\boldsymbol{\theta}}_{kML}) \qquad (10)$$

However, the calculation of $H(\hat{\boldsymbol{\theta}}_{kML})$ requires the true values of the model parameters that are to be estimated. It has been shown that, if the number of observations exceeds the minimal value, the accuracy of the ML estimation tends quickly to the best possible accuracy determined by the Cramer–Rao lower bounds (CRLB), as has been well studied theoretically in (3). Thus, the CRLB of the parameter estimates are used in the actual calculation representing the "conditional" bias and variance (3).

Experiments show that MCBV exhibits a very good performance consistent with both AIC and MDL. Figure 2 depicts the comparison of these three methods on a simulation that uses artifical data generated from four overlapping normal components. Each component represents one local cluster. The values for each component were set to a constant value, the noise of normal distribution was then added to this simulation digital phantom. Three noise levels with different variance were set to keep the same signal-to-noise ratio (SNR), where SNR is defined by

$$SNR = 10\log_{10}\frac{(\Delta\mu)^2}{\sigma^2} \qquad (11)$$

where $\Delta\mu$ is the mean difference between clusters, and σ^2 is the noise power. The AIC, MDL, and MCBV curves, as functions of the number of local clusters K, are plotted in the same figure. According to the information theoretic criteria, the minima of these curves indicate the correct number of the local cluster. From this experimental figure, it is clear that the number of local clusters suggested by these criteria are all correct. For larger noise level, the model selection based on the MCBV criterion provides more differentiable result than the other two criteria.

MODEL PARAMETER ESTIMATION

As the counterpart for adaptive model selection, there are many numerical techniques to perform ML estimation of cluster parameters (8). For example, EM algorithm first calculates the posterior Bayesian probabilities of the data through the observations and the current parameter estimates (E-step), and then updates parameter estimates using generalized mean ergodic theorems (M-step). The procedure cycles back and forth between these two steps. The successive iterations increase the likelihood of the model parameters. The following are the operations taken in an iteration of the EM algorithm for Gaussian mixture distribution: at iteration j,

(1) E-step: First compute the conditional posterior probabilities $h_k^{(j)}(t)$ for all clusters k in class $\boldsymbol{\omega}$, using training samples $\boldsymbol{x}(t)$, $t = 1, \ldots, N$:

$$h_k^{(j)}(t) = \frac{\pi_k^{(j)} g^{(j)}(\mathbf{x}(t)|\boldsymbol{\omega},\boldsymbol{\theta}_k)}{\sum_i \pi_i^{(j)} g^{(j)}(\mathbf{x}(t)|\boldsymbol{\omega},\boldsymbol{\theta}_i)} \qquad (12)$$

(2) M-step:

$$\boldsymbol{\pi}_{\boldsymbol{k}}^{(j+1)} = (1/N)\sum_{t=1}^{N} h_k^{(j)}(t)$$

$$\boldsymbol{\mu}_{\boldsymbol{k}}^{(j+1)} = \left(1\Big/\sum_{t=1}^{N} h_k^{(j)}(t)\right)\sum_{t=1}^{N} h_k^{(j)}(t)\mathbf{x}(t)$$

$$\boldsymbol{\sigma}_{\boldsymbol{k}}^{(j+1)} = \left(1\Big/\sum_{t=1}^{N} h_k^{(j)}(t)\right)\sum_{t=1}^{N} h_k^{(j)}(t)[\mathbf{x}(t)-\boldsymbol{\mu}_k^{(j)}][\mathbf{x}(t)-\boldsymbol{\mu}_k^{(j)}]^T \qquad (13)$$

where $\boldsymbol{\mu}_k$ is the mean vector for cluster k and $\boldsymbol{\sigma}_k^2$ is the variance vector. A neural network interpretation of EM procedure was first introduced by Perlovsky (3).

EM algorithm has the advantages of guaranteed maximum likelihood (ML) convergence and nonrequirement of learning rate parameter. However, as we have shown in Eqs. (12) and (13), EM needs to store all the incoming observations to update the statistical parameters. In other words, EM is preferably applied in off-line situations. An adaptive learning algorithm, called *probabilistic self-organizing mixture* (PSOM) algorithm (12) is proposed to alleviate the high memory demand and to change the parameters immediately after each data point, allowing for high data rates. Like EM algorithm, PSOM also provides winner-takes-in probability (Bayesian "soft") splits of the data, hence allowing the data to contribute simultaneously to multiple clusters. For the sake of simplicity, assume the kernel shape of local cluster to be a Gaussian with mean $\boldsymbol{\mu}_k$ and variance $\boldsymbol{\sigma}_k^2$. The learning rule of PSOM is derived from a stochastic gradient descent scheme for minimizing the relative entropy (the Kullback-Leibler distance) (4,12,14), with respect to the unconstrained parameters, $\boldsymbol{\mu}_k$ and $\boldsymbol{\sigma}_k^2$ (15): given N randomly ordered training samples $\boldsymbol{x}(t)$, $t = 1, \ldots, N$,

$$\boldsymbol{\mu}_{\boldsymbol{k}}^{(t+1)} = \boldsymbol{\mu}_{\boldsymbol{k}}^{(t)} + a(t)(\mathbf{x}(t+1) - \boldsymbol{\mu}_{\boldsymbol{k}}^{(t)})z_{(t+1)k}^{(t)}, \qquad k = 1, \ldots, K \tag{14}$$

$$\boldsymbol{\sigma}_{\boldsymbol{k}}^{2(t+1)} = \boldsymbol{\sigma}_{\boldsymbol{k}}^{2(t)} + b(t)[(\mathbf{x}(t+1) - \boldsymbol{\mu}_{\boldsymbol{k}}^{(t)})^2 - \boldsymbol{\sigma}_{\boldsymbol{k}}^{2(t)}]z_{(t+1)k}^{(t)}, \qquad k = 1, \ldots, K \tag{15}$$

similar to the $h(t)$ in Eq. (12), $z_{(t+1)k}^{(t)}$ is the posterior Bayesian probability, defined by

$$z_{(t+1)k}^{(t)} = \frac{\boldsymbol{\pi}_{\boldsymbol{k}}^{(t)} g(\mathbf{x}(t+1)|\boldsymbol{\mu}_{\boldsymbol{k}}^{(t)}, \boldsymbol{\sigma}_{\boldsymbol{k}}^{2(t)})}{p(\mathbf{x}(t+1)|\theta)} \tag{16}$$

Note that $a(t)$ and $b(t)$ are introduced as the learning rates, two sequences converging to zero, ensuring unbiased estimates after convergence. The idea behind this update rule is motivated by the principle that every weight of a network should be given its own learning rate and that these learning rates should be allowed to vary over time (15). Based on generalized mean ergodic theorem (16), updates can also be obtained for the constrained regularization parameters, $\boldsymbol{\pi}_k$, in the SFMD model. For simplicity, given an asymptotically convergent sequence, the corresponding mean ergodic theorem, that is, the recursive version of the sample mean calculation, should hold asymptotically (8). From the *M-step* of EM algorithm, one can write,

$$\boldsymbol{\pi}_{\boldsymbol{k}}^{(t+1)} = \sum_{i=1}^{t+1} \frac{1}{t+1} z_{ik}^{(t)} = \frac{t}{t+1} \sum_{i=1}^{t} \frac{1}{t} z_{ik}^{(t)} + \frac{1}{t+1} z_{(t+1)k}^{(t)} \tag{17}$$

Then, define the interim estimate of π_k by:

$$\boldsymbol{\pi}_{\boldsymbol{k}}^{(t+1)} = \frac{t}{t+1} \boldsymbol{\pi}_{\boldsymbol{k}}^{(t)} + \frac{1}{t+1} z_{(t+1)k}^{(t)} \tag{18}$$

Hence the updates given by Eqs. (14), (15), and (18) provide the incremental procedure for computing the SFMD component parameters. Their practical use, however, requires strongly mixing condition (data randomization) and a decaying annealing procedure (learning rate decay). These two steps are currently controlled by user-defined parameters, which may not be optimized for a specific case. Therefore, algorithm initialization must be chosen carefully and appropriately. In addition, the data distribution for each class can also be modeled by a finite generalized Gaussian mixture (FGGM) given by (17):

$$f_r(\mathrm{x}(i)) = \sum_{k=1}^{K_r} \boldsymbol{\pi}_{\boldsymbol{k}} g_k(\boldsymbol{x}(i)) \tag{19}$$

where $g_k(\mathbf{x}(i))$ is the generalized Gaussian kernel, representing the kth local cluster's pdf defined by

$$g_k(\mathbf{x}(i)) = \frac{\alpha\beta_k}{2\Gamma(1/\alpha)} \exp[-|\beta_k(\mathbf{x}(i) - \boldsymbol{\mu}_{\boldsymbol{k}})|^\alpha], \qquad \alpha > 0 \tag{20}$$

where $\boldsymbol{\mu}_k$ is the mean, $\Gamma(\cdot)$ is the Gamma function, and β_k is a parameter related to the variance $\boldsymbol{\sigma}_k$ by

$$\beta_k = \frac{1}{\sigma_k}\left[\frac{\Gamma(3/\alpha)}{\Gamma(1/\alpha)}\right]^{1/2} \tag{21}$$

It has been shown that, when $\alpha = 2.0$, one has the Gaussian pdf; when $\alpha = 1.0$, one has the Laplacian pdf. When $\alpha \gg 1$, the distribution tends to a uniform pdf; when $\alpha < 1$, the pdf becomes sharp. Therefore, the generalized Gaussian model is a suitable model for those data whose statistical properties are unknown,and the kernel shape can be controlled by selecting different α values.

PARAMETER MODIFICATION FOR MINIMIZING CLASSIFICATION ERRORS

From the introduction it is known that the Bayesian classifier is theoretically the "optimal" classifier, and methods to achieve it have been discussed in the previous two sections. However, in many practical situations, the achieved classifiers may perform worse than expected. Two reasons may cause such disappointment: (1) the final statistical model chosen is not the same as the true object probability model, and (2) the number of training samples is not large enough to form sufficient statistics. In order to solve this problem, Lin et al. (5) propose a modular network called *Probabilistic Decision Based Neural Network* (PDBNN). PDBNN uses the logarithm of the likelihood density function $p(\mathbf{x}|\omega)$ as the discriminant function for object class ω:

$$\phi(\mathbf{x}, \mathbf{w}) = \log\, p(\mathbf{x}|\omega) = \log\left[\sum_k \boldsymbol{\pi}_{\boldsymbol{k}} g(\mathbf{x}|\theta_k)\right] \tag{22}$$

where

$$\mathbf{w} \equiv \{\boldsymbol{\mu}_{\boldsymbol{k}}, \boldsymbol{\sigma}_{\boldsymbol{k}}, \boldsymbol{\pi}_{\boldsymbol{k}}, T\} \tag{23}$$

and T is the threshold of the subnet.

Decision-based learning algorithm fine-tunes the decision boundaries formed by those Bayesian posterior probabilities for different object classes.

Unlike most ML estimation techniques, which estimate parameters for class ω_j by using the training samples *belonging to ω_j only,* decision-based learning algorithm utilizes "useful" samples from all the object classes to do reinforced and antireinforced learning. Given a set of training patterns $\mathbf{X} = \{\mathbf{x}(t);\ t = 1, 2, \ldots, M\}$. The set $\mathbf{X}$ is further divided into the "positive training set" $\mathbf{X}^+ = \{\mathbf{x}(t);\ \mathbf{x}(t) \in \omega,\ t = 1, 2, \ldots, N\}$ and the "negative training set" $\mathbf{X}^- = \{\mathbf{x}(t);\ \mathbf{x}(t) \notin \omega,\ t = N + 1, N + 2, \ldots, M\}$. Define an energy function

$$E = \sum_{t=1}^{M} l(d(t)) \tag{24}$$

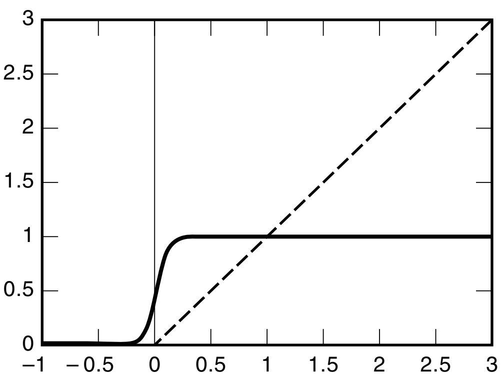

Figure 3. The difference between the penalty functions of a hard-decision DBNN (solid line) and a fuzzy-decision neural network (dashed line).

where

$$d(t) = \begin{cases} T - \phi(\mathbf{x}(t), \mathbf{w}) & \text{if } \mathbf{x}(t) \in \mathbf{X}^+ \\ \phi(\mathbf{x}(t), \mathbf{w}) - T & \text{if } \mathbf{x}(t) \in \mathbf{X}^- \end{cases} \tag{25}$$

The discriminant function $\phi(\mathbf{x}(t), \mathbf{w})$ is defined in Eq. 22. T is the threshold value. The *penalty function* l can be either a piecewise linear function

$$l(d) = \begin{cases} \zeta d & \text{if } d \geq 0 \\ 0 & \text{if } d < 0 \end{cases} \tag{26}$$

where ζ is a positive constant, or a sigmoidal function

$$l(d) = \frac{1}{1 + e^{-d/\xi}} \tag{27}$$

Figure 3 depicts these two possible penalty functions. The reinforced and anti-reinforced learning rules for the network are the following:

$$\begin{aligned} &\text{Reinforced Learning:} && \mathbf{w}^{(j+1)} = \mathbf{w}^{(j)} + \eta l'(d(t)) \nabla \phi(\mathbf{x}(t), \mathbf{w}) \\ &\text{Antireinforced Learning:} && \mathbf{w}^{(j+1)} = \mathbf{w}^{(j)} - \eta l'(d(t)) \nabla \phi(\mathbf{x}(t), \mathbf{w}) \end{aligned} \tag{28}$$

The gradient vectors in Eq. (28) can be computed in the similar fashion as what was done in PSOM. If the misclassified training pattern is from the positive training set, reinforced learning will be applied. If the training pattern belongs to the so-called negative training set, then only the anti-reinforced learning rule will be executed—since there is no "correct" class to be reinforced.

Note that, since the linear penalty function imposes too excessive a penalty for patterns with large margins of error, the network learning may be deteriorated by outlier patterns. In contrast, the sigmoidal function treats the errors with equal penalty, once the magnitude of error exceeds a certain threshold. This *soft* decision-making leads asymptotically to a minimum error classification (18). However, the proper threshold value ξ is different from application to application, so it must be carefully selected. Also note that, although the fine-tuning of the decision boundaries may cause the probability estimation of an individual object class to be less than optimal, it is believed to lead to better classification results, in general. For example, significant improvement in classification result (e.g., recognition rate from 70% to 90%) contributed by the fine-tuning process is observed in the face-recognition experiment in (5).

APPLICATION EXAMPLE: FACE RECOGNITION

In the final section of this article, a face-recognition system is used as an example, showing how a hypothesis-testing scheme can be implemented in real applications. A PDBNN-based face-recognition system (5) is developed under a collaboration between Siemens Corporate Research, Princeton, and Princeton University. The total system diagram is depicted in Fig. 4. All four main modules—face detector, eye localizer, feature extractor, and face recognizer, are implemented on a Sun Sparc10 workstation. An RS-170 format camera, with a 16 mm, F1.6 lens is used to acquire image sequences. The S1V digitizer board digitizes the incoming image stream into 640 × 480 8-bit gray-scale images, and stores them into the frame buffer. The image acquisition rate is on the order of 4 to 6 frames per second. The acquired images are then down-sized to 320 × 240 for the following processing.

As shown in Fig. 4, the processing modules are executed sequentially. A module will be activated only when the incoming pattern passes the preceding module (with an agreeable confidence). After a scene is obtained by the image-acquisition system, a quick detection algorithm based on binary template matching is applied, to detect the presence of a proper sized moving object. A PDBNN face detector is then activated to determine whether there is a human face. If positive, a PDBNN eye localizer is activated to locate both eyes. A sub-image (approx. 140 × 100) corresponding to the face region will then be extracted. Finally, the feature vector is fed into a PDBNN face recognizer for recognition and subsequent verification.

The face detector, the eye localizer, and the face recognizer adopt the hypothesis-testing scheme. Face detection and eye localization are basically two-state classification problems. If the input pattern is a face or eye, it will be classified as the face or eye class (ω_1), otherwise it is a non-face or non-eye pattern (ω_0). Face recognition is M-state or $M+1$-state classification problem. It is an M-state problem if the task is to recognize one person in an M-people database. It is an $M+1$-state problem if the task is not only to recognize one out of M people, but also to reject persons who are not in the database (the "unknown" class). PDBNN is observed to have special advantage in the $M+1$-state problem, because it adopts log-likelihood as its discriminant function. Interested readers should consult (5).

The system built upon the proposed has been demonstrated to be applicable under reasonable variations of orientation and/or lighting, and with possibility of eyeglasses. This method has been shown to be very robust against large variation of face features, eye shapes, and cluttered background (5). The algorithm takes only 200 ms to find human faces in an image with 320 × 240 pixels on a Sun Sparc10 workstation. For a facial image with 320 × 240 pixels, the algorithm takes 500 ms to locate two eyes. In the face-recognition stage, the computation time is linearly proportional to the number of persons in the database. For a 200-person database, it

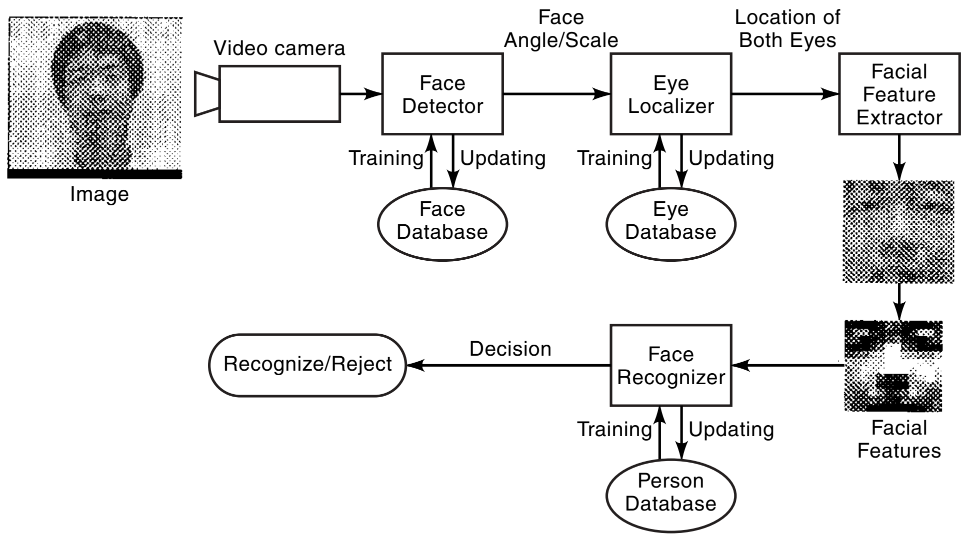

Figure 4. System configuration of the face-recognition system. The face-recognition system acquires images from a video camera. The face detector determines if there are faces inside images. The eye localizer indicates the exact positions of both eyes. It then passes their coordinates to the facial feature extractor, to extract low-resolution facial features as input by face recognizer.

takes less than 100 ms to recognize a face. Furthermore, because of the inherent parallel and distributed processing nature of PDBNN, the technique can be easily implemented via specialized hardware for real-time performance.

An experiment on the face database is conducted by using the Olivetti Research Laboratory in Cambridge, UK (the ORL database). There are 10 different images of 40 different persons. There are variations in facial expression (open/close eyes, smiling/non-smiling), facial details (glasses/no glasses), scale (up to 10%), and orientation (up to 20 degrees). A HMM-based approach is applied to this database and achieves a 13% error rate (19). The popular eigenface algorithm (20) reports the error rate around 10% (19,21). In (22), a pseudo 2-D HMM method is used and achieves 5% at the expense of long computation time (4 min/pattern on Sun Sparc II). In (21) Lawrence et al. use the same training and test set size as Samaria did, and a combined neural network (self-organizing map and convolutional neural network) to do the recognition. This scheme spent four hours to train the network and less than one second for recognizing one facial image. The error rate for ORL database is 3.8%. The PDBNN-based system reaches similar performance (4%), but has much faster training and recognition speed (20 min for training and less than 0.1 s for recognition). Both approaches run on SGI Indy. Table 1 summarizes the performance numbers on ORL database.

Table 1. Performance of Different Face Recognizers on the ORL Database

System	Error Rate	Classification Time	Training Time
PDBNN	4%	<0.1 s	20 min
SOM + CN	3.8%	<0.5 s	4 h
Pseudo 2-D-HMM	5%	240 s	n/a
Eigenface	10%	n/a	n/a
HMM	13%	n/a	n/a

Part of this table is adapted from (21).

BIBLIOGRAPHY

1. D. M. Titterington, A. F. M. Smith, and U. E. Markov, *Statistical Analysis of Finite Mixture Distributions,* New York: Wiley, 1985.
2. R. Duda and P. Hart, *Pattern Classification and Scene Analysis,* New York: Wiley, 1973.
3. L. Perlovsky and M. McManus, Maximum likelihood neural networks for sensor fusion and adaptive classifiation, *Neural Netw.,* **4**: 89–102, 1991.
4. Y. Wang, Image quantification and the minimum conditional bias/variance criterion. *Proc. 30th Conf. Inf. Sci. Syst.,* Princeton, March 1996, pp. 1061–1064.
5. Shang-Hung Lin, S. Y. Kung, and L. J. Lin, Face recognition/detection by probabilistic decision-based neural network, *IEEE Trans. Neural Netw.,* **8**: 114–132, 1997.
6. L. Xu and M. I. Jordan, *On convergence properties of the EM algorithm for Gaussian mixture,* Technical Report, MIT Artificial Intelligence Laboratory, January 1995.
7. S. Haykin, *Neural Networks: A Comprehensive Foundation,* New York: Macmillan, 1994.
8. D. M. Titterington, Comments on 'application of the conditional population-mixture model to image segmentation', *IEEE Trans. Pattern Anal. Mach. Intell.,* **6**: 656–658, 1984.
9. C. E. Priebe, Adaptive mixtures, *J. Amer. Stat. Assoc.,* **89** (427): 1–11, 1994.
10. H. Akaike, A new look at the statistical model identification, *IEEE Trans. Autom. Control,* **19**: 716–723, 1974.
11. J. Rissanen, Minimax entropy estimation of models for vector processes, *Syst. Identification,* 97–119, 1987.
12. Y. Wang et al., Data mapping by probabilistic modular networks and information theoretic criteria, to appear in *IEEE Trans. Signal Process.*
13. S. German, E. Bienenstock, and R. Doursat, Neural networks and the bias/variance dilemma, *Neural Computat.,* **4**: 1–52, 1992.
14. J. L. Marroquin and F. Girosi, *Some extensions of the K-means algorithm for image segmentation and pattern classification,* Technical Report, MIT Artificial Intelligence Laboratory, January 1993.
15. R. A. Jacobs, Increased rates of convergence through learning rate adaptation, *Neural Netw.,* **1**: 295–307, 1988.

16. T. M. Cover and J. A. Thomas, *Elements of Information Theory,* New York: Wiley, 1991.
17. J. Zhang and J. M. Modestino, A model-fitting approach to cluster validation with application to stochastic model-based image segmentation, *IEEE Trans. Pattern Anal. Mach. Intell.,* **12**: 1009–1017, 1990.
18. B. H. Juang and S. Katagiri, Discriminative learning for minimum error classification, *IEEE Trans. Signal Process.,* **40** (12): 3043–3054, 1992.
19. F. S. Samaria and A. C. Harter, Parameterization of a stochastic model for human face identification, *Proc. IEEE Workshop Applications Comput. Vision,* Sarasota, FL, 1994.
20. M. Turk and A. Pentland, Eigenfaces for recognition, *J. Cognitive Neurosci.,* **3**: 71–86, 1991.
21. S. Lawrence et al., *Face recognition: A convolutional neural network approach,* Technical Report, NEC Research Institute, 1995.
22. F. S. Saramia, *Face recognition using hidden Markov model,* Ph.D. Thesis, University of Cambridge, Cambridge, UK 1994.

SHANG-HUNG LIN
EPSON Research and Development Inc.
S. Y. KUNG
Princeton University

TESTING INSULATION. See INSULATION TESTING.
TEST-SET. See STANDING WAVE MEASUREMENT AND NETWORK ANALYZER CALIBRATION.
TEST STRUCTURES FOR SEMICONDUCTOR MANUFACTURING. See SEMICONDUCTOR MANUFACTURING TEST STRUCTURES.
TEXT RECOGNITION. See DOCUMENT IMAGE PROCESSING.
TEXT RETRIEVAL. See DOCUMENT HANDLING; INFORMATION RETRIEVAL AND ACCESS.
TEXTURE, IMAGE. See IMAGE SEGMENTATION.
TEXTURE MAPPING. See VISUAL REALISM.
TEXTURE OF IMAGES. See IMAGE TEXTURE.