Hierarchical Discriminant Regression

Wey-Shiuan Hwang, *Member*, *IEEE*, and Juyang Weng, *Member*, *IEEE*

Abstract—The main motivation of this paper is to propose a new classification and regression method for challenging high-dimensional data. The proposed new technique casts classification problems (class labels as output) and regression problems (numeric values as output) into a unified regression problem. This unified view enables classification problems to use numeric information in the output space that is available for regression problems but are traditionally not readily available for classification problems—distance metric among clustered class labels for coarse and fine classifications. A doubly clustered subspace-based hierarchical discriminating regression (HDR) method is proposed in this work. The major characteristics include: 1) Clustering is performed in both output space and input space at each internal node, termed "doubly clustered." Clustering in the output space provides virtual labels for computing clusters in the input space. 2) Discriminants in the input space are automatically derived from the clusters in the input space. These discriminants span the discriminating subspace at each internal node of the tree. 3) A hierarchical probability distribution model is applied to the resulting discriminating subspace at each internal node. This realizes a coarse-to-fine approximation of probability distribution of the input samples, in the hierarchical discriminating subspaces. No global distribution models are assumed. 4) To relax the per class sample requirement of traditional discriminant analysis techniques, a sample-size dependent negative-log-likelihood (NLL) is introduced. This new technique is designed for automatically dealing with small-sample applications, large-sample applications, and unbalanced-sample applications. 5) The execution of HDR method is fast, due to the empirical logarithmic time complexity of the HDR algorithm. Although the method is applicable to any data, we report the experimental results for three types of data: synthetic data for examining the near-optimal performance, large raw face-image data bases, and traditional databases with manually selected features along with a comparison with some major existing methods, such as CART, C5.0, and OC1.

Index Terms—Discriminant analysis, classification and regression, decision trees, high-dimensional data, image retrieval.

1 Introduction

The capability of computers to efficiently and effectively retrieve information from image databases gives a significant impact on the progress of digital library technology. A central task of a multimedia information system is to efficiently store, quickly and correctly retrieve, and easily manage images in the database.

An essential issue for an image database is the representation of the image. We can categorize the content-based image retrieval into two types: the model-based and the appearance-based. The model-based approach uses manually defined features to represent objects in the images. A lot of efforts has been made in this approach [1], [2], [3], [4], [5]. Most of them have been focusing on designing an efficient algorithm from a set of manually selected features. The strength of the model-based approach is the efficiency in representing images. With a proper design and a restricted domain of images, only a very small number of parameters is sufficient to represent the objects in the image and to distinguish among different objects. However, the model-based approach is difficult to generalize. For example, given a face image database, the designer needs to manually find the features for faces. The face features become useless when a car image database is presented. The designer has to find another set of features for car images. Such a process of manually designing features cannot scale up to large and complex domains since there are countless models to be built.

The appearance approach has recently drawn much attention in machine vision [6], [7], [8]. Instead of relying on human designer to define features, the appearance-based approach enables machines to automatically derive features from image samples. To do so, it considers a two-dimensional image as a long vector. Statistical classification tools are applied directly to the sample vectors. One example is the nearest neighbor (NN) classifier. As is well-known, an NN classifier is very time and space consuming for high-dimensional image space or a large image database. To use fewer features to represent a set of images, the principal component analysis (PCA) has been used for face recognition [6], [9]. PCA can optimistically reconstruct the images represented with the least mean square errors. However, the features which can well represent the original data set are not necessarily good for the purpose of classification. The features derived from the linear discriminant analysis (LDA) are meant for well distinguishing different classes and, thus, are relatively better for the purpose of classification, provided that the samples contain sufficient information [10].

The second issue for image databases is how to organize the represented features so that the retrieval is both fast and successful. Linear search is very time-consuming which makes it not practical for very large image databases. One way to solve this problem is to use a decision tree. A well designed decision tree can retrieve a matched sample with a

• *The authors are with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824. E-Mail: {hwangwey, weng}@cse.msu.edu.*

Manuscript received 8 Mar. 2000; revised 30 Aug. 2000; accepted 31 Aug. 2000.
Recommended for acceptance by R. Sharma.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 111665.

logarithmic time complexity. This is a very useful characteristic for large image databases. There is a very rich literature about decision trees, see surveys [11], [12], [13], [14]. However, the applications of decision trees have been traditionally for a low-dimensional feature space with manually selected features. This is true, largely because humans cannot define a large number of useful features. The appearance-based approach drastically changed this situation. Traditional decision trees for a low-dimensional input space have been found not suitable for input dimensionality of a few thousand and up, even after data-dimensional reduction using techniques such as PCA, as we report in this paper. A major reason is the high complexity of sample distribution that cannot be adequately captured by a single-level PCA. As demonstrated by [15], if a different subspace is computed at each internal node of the tree, a better generalization power results. In [16], they have demonstrated that using an appropriate tree the classification can be better and faster. A series of limitations of existing tree classifiers has motivated the work presented here with advances and advantages summarized in the abstract.

2 A New Subspace Regression Tree for High-Dimensional Learning

2.1 Hierarchical Discriminant Regression

Discriminant analysis can be categorized into two types according to their outputs: class-label output and numerical output. If the output is a class label, the task is called classification. Otherwise, the task is called regression. We cast both classification and regression tasks into a regression one in this study.

A classification task can be stated as follows: Given a training sample set $L = \{(x_i, l_k) \mid i = 1, 2, \ldots, n,\ k = 1, 2, \ldots, c\}$, where $x_i \in \mathcal{X}$ is an input (feature) vector and l_k is the symbolic label of x_i, the task is to determine the class label of any unknown input $x \in \mathcal{X}$.

How can one cast a classification task into a regression one? We consider three cases:

1. If a cost matrix $[c_{ij}]$ is readily available from applications, where c_{ij} is the cost of confusing classes i and j, one can embed c class labels into an $(c-1)$-dimensional Euclidean output space by assigning vector y_i and y_j to class i and j, respectively, where $i, j = 1, 2, \ldots, c$, so that $||y_i - y_j||$ is as close to c_{ij}, as much as possible. This process is not always possible since a predefined cost matrix $[c_{ij}]$ is not always easy to provide.
2. Canonical mapping. Map c class labels into a c-dimensional output space so that the ith class label corresponds to a vector in which the ith component is one and all other components are zeros. After this mapping, the distance between any two different class labels is the same: one. This label mapping does not assign different distances to different output vectors but will do so for coarse classes in a coarse-to-fine classification as we explain below.
3. Mapping labels into the input space. Each sample (x_i, l_k) belonging to class k is converted to (x_i, y_k) where y_k, the vector class label, is the mean of all x_i that belong to the same class. This label mapping scheme considers the distance in input space among different classes. In many applications, this is a desirable way. In each leaf node of the regression tree, each training sample (x_i, y_k) has a link to label l_k so that when (x_i, y_k) is found as a good match for unknown input x, l_k is directly output as the class label. There is no need to search for the nearest neighbor in the output space for the corresponding class label.

As we know, one cannot map a numeric output space into a set of class labels without losing the numeric properties among an infinite number of possible numerical vectors. Therefore, a regression problem is more general than the corresponding classification problem.

2.1.1 Discriminant Analysis for Numerical Output

Now, we consider a general regression problem: approximating a mapping $h : \mathcal{X} \mapsto \mathcal{Y}$ from a set of training samples $\{(x_i, y_i) \mid x_i \in \mathcal{X},\ y_i \in \mathcal{Y},\ i = 1, 2, \ldots, n\}$. If y_i was a class label, we could use linear discriminant analysis (LDA) [17] since the within-class scatter and between-class scatter matrices are all defined. Unfortunately, if each class has only very few samples, the within-class scatter matrix is poorly estimated and the LDA is not very effective. If the classification problem is cast into a regression one, it is possible to form coarse classes so that each class has more samples, which enable better estimation of the within class scatter matrix. However, if y_i is a numerical output, which can take any value for each component, it is a challenge to figure out an effective discriminant analysis procedure.

To attack this challenge, we introduce a new hierarchical statistical modeling method. Consider the mapping $h : \mathcal{X} \mapsto \mathcal{Y}$, which is to be approximated by a regression tree,[1] called a hierarchical discriminant regression (HDR) tree, for the high-dimensional space $\mathcal{X}$. Our goal is to automatically derive discriminant features, although no class label is available (other than the numerical vectors in space $\mathcal{Y}$).

Two types of clusters are formed at each node of the HDR tree—y-clusters and x-clusters, as shown in Fig. 1. The y-clusters are clusters in the output space $\mathcal{Y}$ and x-clusters are those in the input space $\mathcal{X}$. There are a maximum of q clusters of each type at each node. The q y-clusters determine the virtual class label of each training sample (x, y) based on its y part. The virtual class label is used to determine which x-cluster the input sample (x, y) should update using its x part. Each x-cluster approximates the sample population in $\mathcal{X}$ space for the samples that belong to it. It may spawn a child node from the current node if a finer approximation is required. At each node, y in (x, y) finds the nearest y-cluster in, e.g., Euclidean distance (more general distance metrics can also be used). This y-cluster indicates to which corresponding x-cluster the input (x, y)

1. A regression tree is, by definition, a decision tree whose output is a numeric vector while a classification tree is a decision tree whose output is a class label [14].

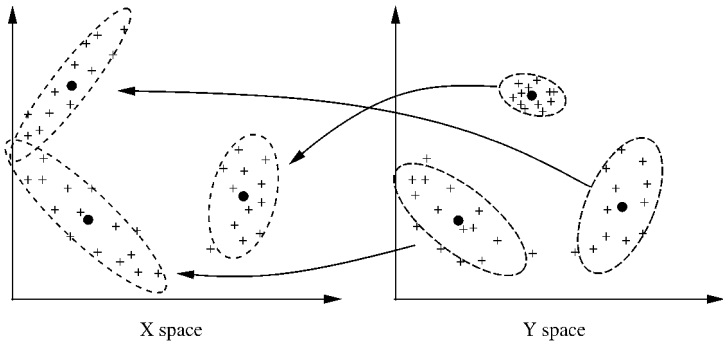

Fig. 1. Y-clusters in space $\mathcal{Y}$ and the corresponding x-clusters in space $\mathcal{X}$. The first and the second order statistics are computed for each cluster. By default, the normalized Mahalanobis distance is used for x-cluster and the Euclidean distance is used for y-cluster.

belongs. Then, the x part of (x, y) is used to compute the statistics of the x-cluster (the mean vector and the covariance matrix). These statistics of every x-cluster are used to estimate the probability for the sample (x, y) to belong to which x-cluster, whose probability distribution is modeled as a multidimensional Gaussian at this level. A total of q centers of the q x-clusters gives $q-1$ discriminant features which span $(q-1)$-dimensional discriminant space. A probability-based distance (to be discussed in Section 2.2) from x to each of the q x-clusters is computed to determine which x-cluster should be further searched. If the probability is high enough, the sample (x, y) should further search the corresponding child (maybe more than one but with an upper bound k) recursively, until the corresponding terminal nodes are found.

For computational efficiency, none of the x-clusters and y-clusters keep actual input samples, unlike the traditional clustering methods. Only the first order statistics are used to represent the clusters. For example, each y-cluster keeps the mean vector and the covariance matrix, depending on the distance metric used in the $\mathcal{Y}$ space, while each x-cluster keeps the mean vector and the full covariance matrix in an efficient form.

In summary, the algorithm recursively builds an HDR tree from a set of training samples. The deeper a node is in the tree, the smaller the variances of its x-clusters are. The way we use decision trees is very different from the traditional ones like CART [14] and OC1 [13]. We do not compute the features of all the data set and then build the decision tree only on these features. Instead, the method automatically derives (linear) features for each internal node using all the samples assigned to that node. Thus, the tree is also a feature deriver, not just a feature selector. Our tree does not select an input component at a time as other trees do. The proposed HDR tree in fact automatically derives features, which are linear combination of all the input components using class labels. The following is the outline of the algorithm for tree building and retrieval.

Procedure 1. `BuildSubtree`: Given a node N and a subset S' of the training samples that belong to N, $S' = \{(x_i, y_i) \mid x_i \in \mathcal{X},\ y_i \in \mathcal{Y},\ i = 1, 2, \ldots, n\}$, build the subtree which roots from the node N using S recursively. At most q, clusters are allowed in one node.

1. Let p be the number of the clusters in node N.
   - Call `Clustering-Y` (procedure 2) to obtain p y-clusters.
   - If y_i belongs to jth y-cluster, then x_i belongs to jth x-cluster.
2. Compute the mean and covariance matrices of each x-cluster.
3. For every x_i of (x_i, y_i) in S':
   - Find the nearest x-cluster j according to the probability-based distances.
   - Assign the sample (x_i, y_i) to cluster j.
4. For each cluster j, now we have a portion of samples S_j assigned to it. If the largest Euclidean distance among y_i's in the x-cluster is larger than a number δ_y, a child node N_j of N is created from the x-cluster and this procedure is called recursively with input samples S_j and node N_j. The number δ_y represents the sensitivity of the HDR tree in the space $\mathcal{Y}$.

Procedure 2. `Clustering-Y`: Given a set of output vectors $Y = (y_1, y_2, \ldots, y_n)$, return p y-clusters. $p \leq q$, where q represents the maximum number of clusters allowed in one node.

1. Let the mean Y_1 of y-cluster 1 be y_1. Set $p = 1$ and $i = 2$.
2. For i from 2 to n do
   a. Find the nearest y-cluster j for y_i.
   b. Compute the Euclidean distance $d = dist(y_i, Y_j)$, where Y_j is the jth y-cluster mean, $j \leq p$.
   c. If $d \geq \delta_y$ and $p < q$, let the mean Y_{p+1} of y-cluster $p+1$ be y_i. Set $p = p + 1$.
   d. Otherwise, update y-cluster j using the new member y_i.

The procedure to create a HDR tree just calls procedure `BuildSubtree` with root R and all the training samples $S = \{(x_i, y_i) \mid x_i \in \mathcal{X},\ y_i \in \mathcal{Y},\ i = 1, 2, \ldots, n\}$. The procedure to query the HDR tree for an unknown sample x is described below.

Procedure 3. `Retrieval`: Given an HDR tree T and a sample x, estimate the corresponding output vector y. A

parameter k specifies the upper bound in the width of parallel tree search.

1. From the root of the tree, compute the probability-based distance to every cluster in the node. Select at most top k x-clusters which have the smallest probability-based distances to x. These x-clusters are called active x-clusters.
2. For every active cluster received, check if it points to a child node. If it does, mark it inactive and explore its child node by computing the probability-based distances of x-clusters in the child node. At most, k^2 active x-clusters can be returned.
3. Mark at most k active x-clusters according to the smallest probability-based distances.
4. Do the above steps 2 through 3 recursively until all the resulting active x-clusters are all terminal.
5. Let the cluster c have the shortest distance among all terminal active x-clusters. Output the corresponding mean of its y-cluster as the estimated output y for x.

It is worth noting that the boundary partition uses the samples according to the pseudo y labels. The resulting partition does not have to correctly separate every sample according to its label, because finer levels can do that. The samples that fall into partition cells that are not ideally for them should not be used to compute partition since they will deteriorate the partition (without using correct y-labels.) This is because we want to use discriminant analysis instead of unsupervised clustering methods.

If we use Gaussian distribution to model each x-cluster, this is a *hierarchical version* of the well-known mixture-of-Gaussian distribution models: The deeper the tree is, the more Gaussians are used and the finer are these Gaussians. At shallow levels, the sample distribution is approximated by a mixture of large Gaussians (with large variances). At deep levels, the sample distribution is approximated by a mixture of many small Gaussians (with small variances). The multiple search paths guided by probability allow a sample x that falls in-between two or more Gaussians at each shallow level to explore the tree branches that contain its neighboring x-clusters. Those x-clusters to which the sample (x, y) has little chance to belong are excluded for further exploration.

2.2 Distance in Discriminating Space

2.2.1 Discriminating Subspace

In the above section, we need to estimate the distance for an input vector x to belong to an x-cluster. For a real-time system, it is typically the case that the system cannot afford to keep all the samples in each cluster. Thus, each cluster will be represented by some statistical measures with an assumed distribution.

We first consider x-clusters. Each x-cluster is represented by its mean as its center and the covariance matrix as its size. However, since the dimensionality of the space $\mathcal{X}$ is typically very high, it is not practical to directly keep the covariance matrix. If the dimensionality of the input space $\mathcal{X}$ is 3,000, for example, each covariance matrix requires $3,000 \times 3,000 = 9,000,000$ numbers! We adopt a more efficient method.

As explained in Section 2.1.1, each internal node keeps up to q x-clusters. The centers of these q x-clusters are denoted by:

$$C = \{c_1, c_2, ..., c_q \mid c_i \in \mathcal{X}, i = 1, 2, ..., q\}. \quad (1)$$

The locations of these q centers tell us the subspace $\mathcal{D}$ in which these q centers lie. $\mathcal{D}$ is a discriminant space, since the clusters are formed based on the clusters in the output space $\mathcal{Y}$. We can compute the between-cluster scatter and within-cluster scatter in the subspace $\mathcal{D}$. Suppose that the number of samples in cluster i is n_i and, thus, the grand total of samples is $n = \sum_{i=1}^{q} n_i$. The mean of the q cluster centers, denoted by c is computed as:

$$c = \frac{1}{n}\sum_{i=1}^{q} n_i c_i.$$

The covariance matrix of cluster i is denoted by Γ_i, $i = 1, 2, ..., q$. The within-cluster scatter matrix is the weighted average of the q covariance matrices:

$$S_w = \frac{1}{n}\sum_{i=1}^{q} n_i \Gamma_i. \quad (2)$$

The between-cluster scatter matrix is the sample covariance matrix for the cluster centers:

$$S_b = \frac{1}{n}\sum_{i=1}^{q} n_i (c_i - c)(c_i - c)^T. \quad (3)$$

The sample mixture matrix is the covariance matrix of all the samples regardless of their cluster assignments and it is also equal to

$$S_m = S_w + S_b.$$

The Fisher's linear discriminant analysis [17], [18] finds a subspace that maximizes the ratio of between-cluster scatter and within-cluster scatter: $|S_b|/|S_w|$. Since we decide to use the entire discriminant space $\mathcal{D}$, we do not need to consider the within-cluster scatter here in finding $\mathcal{D}$ and, thus, simplifies the computation. Once we find this discriminating space $\mathcal{D}$, we will use the size-dependent negative-log-likelihood (SDNLL) distance as discussed in Section 2.2.2 to take care of the reliability of each dimension in $\mathcal{D}$ by using information that is richer than the matrix S_w.

2.2.2 Size-Dependent Negative-Log-Likelihood

We need a system that fully uses the information available no matter how many samples have been observed. This is an issue that has received little attention, since it is typically assumed that sufficient samples are available. However, it is often not the case in practice. In terms of belongingness, we have three measures: likelihood, Mahalanobis distance, and Euclidean distance. The likelihood uses the covariance matrix of each individual cluster. It requires that each x-cluster has enough samples to estimate the $(q-1) \times (q-1)$ covariance matrix. It is the most demanding in richness of observations. The Mahalanobis distance uses the average of the covariance matrices. It is less demanding since it just

requires that the average of covariance matrix has reasonably rich observations but not necessarily every x-cluster. The Euclidean distance is estimated by $\rho^2 I$ and, thus, has only one parameter ρ, the standard deviation of all samples. Thus, it is the least demanding. When very few samples are available for all the clusters, the Euclidean distance is a suitable measurement.

Consider the negative-log-likelihood (NLL) defined from Gaussian density:

$$L(x, c_i) = \frac{1}{2}(x - c_i)^T \Gamma_i^{-1}(x - c_i) + \frac{1}{2}\ln(|\Gamma_i|). \quad (4)$$

We call it Gaussian NLL for x to belong to the ith cluster. We call it Mahalanobis NLL if Γ_i is replaced by the within-class scatter matrix—the average of the covariance matrices. We call it Euclidean NLL if Γ_i is replaced by a scalar matrix $\rho^2 I$. We would like to use the Euclidean NLL when the number of samples in the node is small. Gradually, as the number of samples increases, the within-class scatter matrix of q x-clusters is better estimated. Then, we would like to use the Mahalanobis NLL. When a cluster has very rich observations, we would like to use the full Gaussian NLL for it. However, since we approximate the global distribution using coarse-to-fine and global-to-local techniques, no assumption about global distribution is made.

Before we discuss how to realize this transition among the three NLLs, we discuss different characteristics of them. Suppose that the input space is $\mathcal{X}$ and the discriminating subspace for an internal node is $\mathcal{D}$. The Euclidean NLL treats all the dimensions in the discriminating subspace $\mathcal{D}$ the same way, although some dimensionalities are more important than others. It has only one parameter ρ to estimate. The Mahalanobis NLL uses within-class scatter matrix S_w computed from all the samples in all the q x-clusters. It uses the inverse matrix S_w^{-1} as the weight in computing NLL. The meaning of this Mahalanobis matrix weight S_w^{-1} is as follows: The matrix S_w^{-1} properly rotates the basis b_1, $b_2, ..., b_{q-1}$ of the subspace $\mathcal{D}$ so that the correlation about the new basis vectors b'_1, $b'_2, ..., b'_{q-1}$ all vanishes. Then S_w^{-1} applies to each rotated basis vector b'_i, $i = 1, ..., q-1$, a weight which is the inverse of the sample variance along b'_i. It can be noticed that, using Mahalanobis NLL as the weight is equivalent to using Euclidean NLL in the basis computed from Fisher's LDA procedure [17], [10]. Thus, the Mahalanobis NLL takes care of the reliability of different input components but the Euclidean NLL does not. The former not only decorrelates the input components, but also weighs all decorrelated new components. The number of parameters in S_w is $q(q-1)/2$ and, thus, the Mahalanobis NLL requires more samples than the Euclidean NLL. Next, consider the Gaussian NLL. As we can see, the Mahalanobis NLL does not treat different x-clusters differently because it uses a single within-class scatter matrix S_w for all the q x-clusters in each internal node. This is not the case with Gaussian NLL. Using Gaussian NLL, $L(x, c_i)$ in (4) uses the covariance matrix Γ_i of x-cluster i. Note that the decision boundary of the Euclidean NLL and the Mahalanobis NLL is linear and that of the Gaussian NLL is quadratic.

How do we realize such an automatic transition from the three NLLs? We also like to make this transition smooth when the number of samples increases. We have two measurements of maturity for each cluster i: the number of samples n_i and the elapsed time since its creation t_i. The former is appropriate for off-line applications where the elapsed time is not applicable. The latter is more suitable for real-time applications where a very large number of similar samples may be observed during a short time period, causing n_i to increase significantly without having observed enough variation in the data. In the following, we assume that n_i is used as the maturity measurement. For each node, we compute the within-class scatter S_w, which has a total of $n = \sum_{i=1}^{q} n_i$ samples. For each x-cluster in the node, we estimate its sample covariance matrix Γ_i. For each x-cluster, we start with a scalar covariance matrix $\rho^2 I$. For the three types of NLLs, we have three matrices, $\rho^2 I$, S_w, and Γ_i. Consider the number of samples received for each scalar, called the number of samples per scalar (NSPS), of the element of the matrices. The NSPS for $\rho^2 I$ is $n-1$, since the first sample does not give any estimate of the variance. A bounded NSPS is defined to limit the growth of NSPS so that other matrices that contain more scalars can take over when there are a sufficient number of samples for them. Thus, the bounded NSPS for $\rho^2 I$ is

$$b_e = \min\{n - 1, n_s\},$$

where n_s denotes the switch point for the next more complete matrix to take over. At this point, a weight of about 50 percent will be used for $\rho^2 I$ and another weight of 50 percent for the next matrix. How large should n_s be? Consider a series of random variables drawn independently from a distribution with a variance σ^2, the expected sample mean of n random variables has a covariance $\sigma^2/(n-1)$. We can choose a switch confidence value α for $1/(n-1)$. When $1/(n-1) = \alpha$, we consider that the estimate can take about a 50 percent weight. Thus, $n = 1/\alpha + 1$. As an example, let $\alpha = 0.1$ meaning that we trust the estimate with 50 percent weight when the expected variance of the estimate is reduced to about 10 percent of that of a single random variable. We get then $n = 11$, which leads to $n_s = 11$.

Now, consider the NSPS for S_w matrix for the Mahalanobis NLL. The number of independent vectors received is $n - q$ because each of the q x-cluster requires a vector to form its mean vector. Thus, there are $(n-q)(q-1)$ independent scalars. There are $(q-1)q/2$ estimated scalars in the (symmetric) matrix S_w. Thus, the NSPS for S_w is

$$\frac{(n-q)(q-1)}{(q-1)q/2} = \frac{2(n-q)}{q}.$$

To prevent the value from being negative when $n < q$, we take NSPS for S_w to be

$$\max\left\{\frac{2(n-q)}{q}, 0\right\}.$$

TABLE 1
Characteristics of Three Types of Scatter Martices

Type	Euclidean $\rho^2 I$	Mahalanobis S_w	Gaussian Γ_i
NSPS	$n-1$	$\frac{2(n-q)}{q}$	$\frac{2(n-q)}{q^2}$

The bounded NSPS for S_w is

$$b_m = \min\left\{\max\left\{\frac{2(n-q)}{q}, 0\right\}, n_s\right\}.$$

Since the Gaussian NLL cannot be trusted until every matrix Γ_i has received enough samples, we define the NSPS for Γ_i estimates for the Gaussian NLL as the minimum among all the q x-clusters:

$$b_g = \min_{1\le i\le q}\left\{\frac{2(n_i-1)}{q}\right\}. \tag{5}$$

This is somewhat conservative, since it may be the cases where x-cluster that has the least samples is not among the nearest x-clusters. The above NSPS is meant to contain the worst error (when the x-cluster with the fewest samples is the nearest x-cluster). Alternatively, if we are interested in containing the mean error, we may choose NSPS to be the average number of samples per x-cluster:

$$b_g = \frac{1}{q}\sum_{i=1}^{q}\left\{\frac{2(n_i-1)}{q}\right\} = \frac{2(n-q)}{q^2}. \tag{6}$$

In the above computation, we only consider the x-clusters that have at least one sample. It is worth noting that the NSPS for the Gaussian NLL does not need to be bounded, since among our models it is the best estimate with a large number of samples.

Let us consider the three NSPS: b_e, b_m, and b_g. From the definitions, we know that they grow roughly at rates n, $2n/q$ and $2n/q^2$, respectively. b_e gets saturated by n_s the earliest. Then, b_m does. b_g never saturates. Table 1 summarizes the result of the NSPS values of the above derivation.

We define a *size-dependent scatter matrix* (SDSM) W_i as a weighted sum of three matrices:

$$W_i = w_e\rho^2 I + w_m S_w + w_g\Gamma_i, \tag{7}$$

where $w_e = b_e/b$, $w_m = b_m/b$, $w_g = b_g/b$, and b is a normalization factor so that these three weights sum to one: $b = b_e + b_m + b_g$. Using this size-dependent scatter matrix W_i, the *size-dependent negative log likelihood* (SDNLL) for x to belong to the ith x-cluster is defined as:

$$L(x,c_i) = \frac{1}{2}(x-c_i)^T W_i^{-1}(x-c_i) + \frac{q-1}{2}\ln(2\pi) + \frac{1}{2}\ln(|W_i|). \tag{8}$$

It is worth noting the relation between LDA and SDNLL metric. Fisher's LDA in space $\mathcal{D}$ gives a basis for a subspace $\mathcal{D}' \subseteq \mathcal{D}$. This basis is a properly oriented and scaled version for $\mathcal{D}$ so that the within-cluster scatter in $\mathcal{D}'$ is a unit matrix [17, Sections 2.3 and 10.2]. In other words, all the basis vectors in D' are already weighted according to the within-cluster scatter matrix S_w in D. If $\mathcal{D}'$ has the same dimensionality as $\mathcal{D}$, the Euclidean distance in $\mathcal{D}'$ is equivalent to the Mahalanobis distance in $\mathcal{D}$, up to a scale factor. However, if the covariance matrices are very different across different x-clusters and each of them has enough samples to allow a good estimate of individual covariance matrix, Fisher's LDA in space $\mathcal{D}$ is not as good as Gaussian likelihood. The SDNLL in (8) allows automatic and smooth transition between three different types of likelihood, Euclidean, Mahalanobis, and Gaussian, according to the predicted effectiveness of each likelihood.

2.2.3 Computational Considerations

We are now ready to discuss the computational steps for the previous procedures.

The first issue is how to represent the space $\mathcal{D}$ which is spanned by the centers of x-clusters in (1). These centers are vectors in $\mathcal{X}$, which typically has a very high dimensionality. The matrix weighted squared distance from a vector $x \in \mathcal{X}$ to the cluster i is defined by:

$$d^2(x,c_i) = (x-c_i)^T W_i^{-1}(x-c_i), \tag{9}$$

which is two times of the first term of (8).

We have two major issues to deal with. First, the dimensionality of the SDSM W_i is very large if we represent it in $\mathcal{X}$ directly. Second, the sample covariance matrix, which we will use to estimate matrix W_i, is not invertible before the number of samples has reached the high dimensionality of $\mathcal{X}$. We must find an efficient way of computing the weighted matrix square distance from each x-cluster.

A way to address the first issue is to represent the discriminant space $\mathcal{D}$ by orthonormal basis vectors. Using the method explained in Appendix A, we keep an orthonormal basis of the linear manifold $\mathcal{D}$. To address the second issue, we represent the covariance matrix in the orthonormal basis for subspace $\mathcal{D}$ instead of $\mathcal{X}$. Since the dimensionality of D is at most $q-1$, the matrix W_i in the orthonormal basis is much smaller than that in the original space $\mathcal{X}$.

The computational steps are described as follows: Suppose that the dimensionality of the input space $\mathcal{X}$ is d. From q x-cluster centers in (1) in $\mathcal{X}$, use the GSO procedure in Appendix A to compute the $q-1$ orthonormal basis vectors $M = [\epsilon_1, \epsilon_2, ..., \epsilon_{q-1}]$, where each column ϵ_i is a unit basis vector, $i = 1, 2, ...q-1$, and M is a $d \times (q-1)$ matrix. For each x-cluster center c_i, its projection vector in the orthonormal basis M is given by:

$$e_i = M^T c_i.$$

Thus, each x-cluster c_i is represented by only a $(q-1)$-dimensional vector e_i. Given an unknown vector $x \in \mathcal{X}$, project it onto the basis $v = M^T x$. Then, the

matrix-weighted squared distance in (9) is computed only in $(q-1)$-dimensional space using the basis M. The SDSM W_i for each x-cluster is then only a $(q-1)\times(q-1)$ square symmetric matrix, of which only $q(q-1)/2$ parameters need to be estimated. When $q=6$, for example, this number is 15.

Given a column vector v represented in the discriminating subspace with an orthonormal basis whose vectors are the columns of matrix M, the representation of v in the original space $\mathcal{X}$ is $x=Mv$.

To compute the matrix weighted squared distance in (9), we should use a numerically efficient method. For example, we can use Cholesky factorization [19, Section 4.2] which is for a positive definite matrix (which is symmetric). The Cholesky decomposition algorithm computes a lower triangular matrix L from W so that W is represented by $W=LL^T$. The procedure is relegated to Appendix B.

With the lower triangular matrix L, we first compute the difference vector from the input vector x and each x-cluster center c_i: $d_i=x-c_i$. The matrix weighted squared distance is given by:

$$d^2(x,c_i)=v^TW_i^{-1}v=v^T(LL^T)^{-1}v=(L^{-1}v)^T(L^{-1}v). \quad (10)$$

We solve the linear equation $Ly=v$ and then $y=L^{-1}v$ and $d^2(x,c_i)=(L^{-1}v)^T(L^{-1}v)=\|y\|^2$. Since L is a lower triangular matrix, the solution for y in $Ly=v$ is trivial since we simply use the backsubtitution method as described in [20, p. 42].

Another issue is the shape of the automatically generated tree. A major advantage of clustering at each node is that each node tends to spawn an approximately balanced subtree. However, there is no guarantee that the resulting tree is balanced. If we allow certain amount of unbalance but limit its degree, it is called bounded unbalanced tree [16]. In Appendix C, we prove that the height of such a tree and thus the time complexity of updating it per sample is logarithmic in the number of leaf nodes.

3 The Experimental Results

Several experiments were conducted using the proposed HDR algorithm. First, we present the experimental results using synthetic data. Then, we show the experimental results for real face images. In addition to these, the proposed algorithm was also applied to the data with manually extracted features from images.

3.1 Experiments Using Synthetic Data

The motivation of using synthetic data for testing is to investigate the behavior of different distance matrices and to examine the near optimality potential of our new algorithm with known distributions as a ground truth (but our algorithm does not know the distribution).

The first experiment used a data set that has three clusters. The number of dimension is two. Each cluster was modeled by a Gaussian distribution. The centers of the clusters are at $(0,0)$, $(5,0)$, and $(0,5)$, respectively. The covariance matrix of the first cluster is an identity matrix. Those for the second and the third are,

$$\begin{bmatrix}4&0\\0&1\end{bmatrix},\begin{bmatrix}4&0\\0&2.25\end{bmatrix},$$

respectively. We first show the effects of the number of samples. In Fig. 2a, only three samples per class are used to estimate the decision boundaries. Since the number of samples is very small, the SDNLL is very much like that of the Euclidean distance, which results in a reasonable boundary as shown. There are twenty samples per class in Fig. 2b. The decision boundary of SDNLL is between those of Euclidean distance and Mahalanobis distance. Fig. 2c used 500 samples per class. The decision boundary of SDNLL then is very close to that of Gaussian NLL, which is an appropriate distance metric here because of the large number of available samples. Fig. 2d shows the behaviors under the unbalanced sample situation where the third cluster receives much fewer samples than the first while the number of samples for the second cluster is in-between. Fig. 2 indicated that the SDNLL distance metric behaves in the way we wanted.

For the large-sample case of Fig. 2c, we would like to examine how close the error rates are to the best possible Bayesian error rates which use the ground truth about the distribution instead of samples. In the experiment, we used 500 samples per class to train. Table 2 shows the classifications from 1) ground truth of distribution using Bayesian optimality (Bayesian-GT), 2) the parameters estimated from the training data based on Bayesian optimality (Bayesian-Sample), and 3) the proposed new algorithm (SDNLL). It shows that the classification errors are very close among all the measurements. Of course, our method would not be able to be close to the Bayesian error rates if there are not enough samples per class.

The second experiment presented here is for a 3D data set with a 2D discriminating space. There were three clusters, each being modeled by a Gaussian distribution with means, $(0,0,0)$, $(5,0,0)$, and $(0,5,0)$, respectively. Their covariance matrices are:

$$\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix},\begin{bmatrix}4&0&0\\0&1&0\\0&0&1\end{bmatrix},\begin{bmatrix}1&0&0\\0&4&0\\0&0&2.25\end{bmatrix}.$$

There were 500 samples per class for training. Since it is not easy to estimate the error probability in high-dimensional space, we use the random-generated samples to obtain the error probability instead of using analytical probability to compute it. A total of 10,000 trials with 1,500 samples per trial is tested.

We know that the basis is on x-y plane for the ground truth and we expect the deriving discriminating space $\mathcal{D}$ to be roughly so. The basis vectors derived from the proposed algorithm are: $(0.89,-0.45,0.001)$, $(-0.45,-0.89,-0.025)$, which are very close to what we expect. The error rate is estimated on the basis we derived. Table 3 shows the classification results from Bayesian-GT, Bayesian-Sample rule, and our new algorithm SDNLL, like the case of 2D. Of course, the Bayesian-Sample algorithm cannot deal cases with small sample and unbalanced samples.

The third experiment used a high-dimensionality and six clusters. Each class was modeled by a Gaussian distribution in 100 dimensions with means $(0,\ldots,0)$, $(0,5,0,\ldots,0)$,

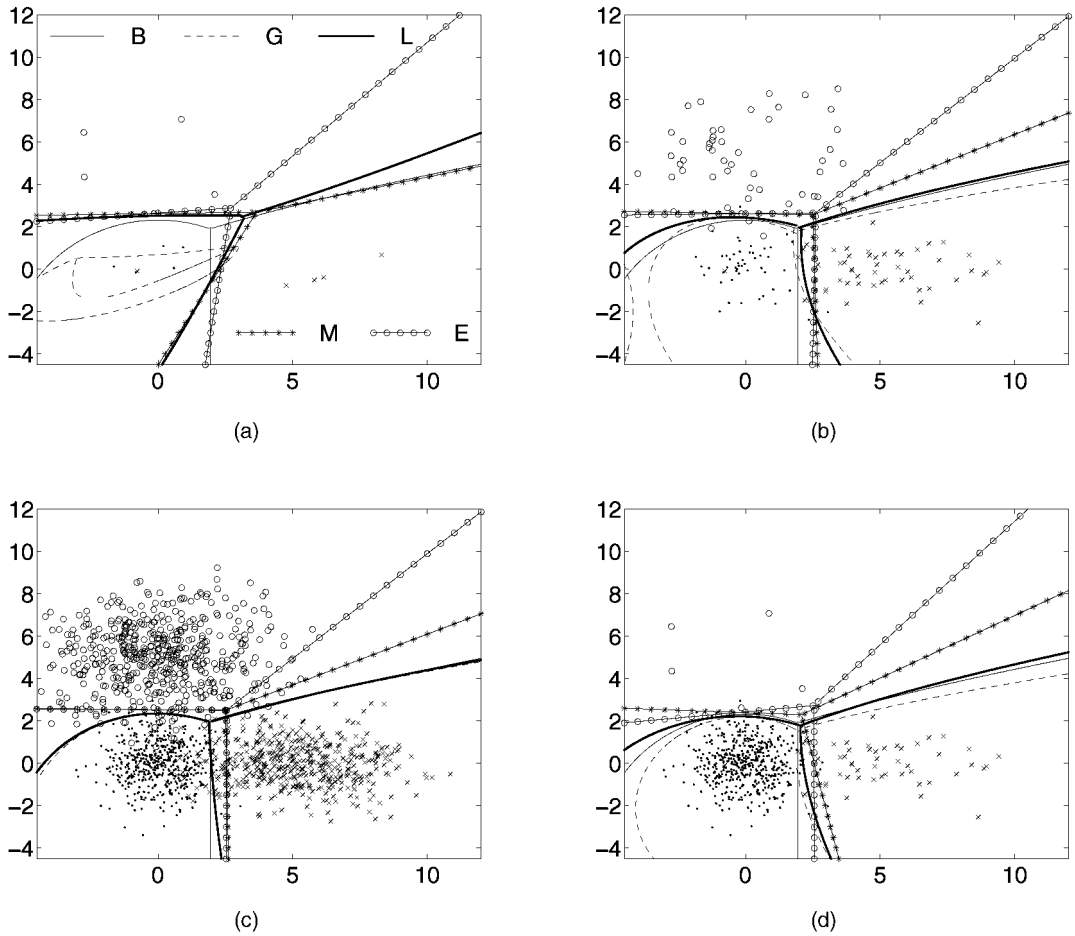

Fig. 2. Decision boundaries estimated by different number of samples for different metrics. Lines "B" mean decision boundaries for Bayesian decision rule. This method uses the ground truth for distribution and thus is independent of samples. Lines "E" are for Euclidean distance measured from a scalar covariance matrix $\rho^2 I$. Lines "G" are measured by Gaussian NLL using estimated full sample covariance matrices for all clusters. Lines "M" are for Mahalanobis distance using a single estimated covariance matrix S_w. Lines "L" use our SDNLL. (a) Small-sample case. (b) Mid-sample case. (c) Large-sample case. (d) Unbalanced-sample case.

$(0,0,5,0,\ldots,0)$, $(0,0,0,5,0,\ldots,0)$, $(0,0,0,0,5,0,\ldots,0)$, and $(0,0,0,0,0,5,0,\ldots,0)$, respectively. The covariance matrix for class 0 is an identity matrix. The covariance matrix for the class $i, i > 0$ is an identity matrix except that the (i,i) element is equal to 2.25. There were 500 samples per class for training and the other 500 samples per class for testing. We expect the basis for discriminating subspace $\mathcal{D}$ is very much in the first six dimensions. The resulting basis is indeed very close to what we expect. Table 4 shows the error rates for the three types of classification rules under comparison. Since the first cluster overlaps with the other clusters, the errors listed in Table 4 in the first row and column are larger than the other rows and columns. The error rates from the proposed algorithm are comparable with those that use direct Bayesian optimality. This is because our framework at each node is based on Bayesian optimality. Due to speed requirements, we do not require global tree optimality since it otherwise requires prohibitive iterations in high-dimensional space.

3.2 Experiments Using Real Face Data

We applied the new algorithm to appearance-based face image retrieval tasks. The first experiment used face images from the Weizmann Institute at Israel. The image database

TABLE 2
Optimality for 2D Synthetic Data

Error rate	Bayesian-GT	Bayesian-Sample	SDNLL
class 1	0.0402	0.0443	0.0498
class 2	0.0660	0.0673	0.0682
class 3	0.0289	0.0298	0.0317

TABLE 3
Error Rates for 3D Synthetic Data

	Bayesian-GT	Bayesian-Sample	SDNLL
class 1	7%	7.2%	7.2%
class 2	6.8%	8.2%	7.6%
class 3	1.6%	2.0%	4.8%

TABLE 4
Error Rates for 100D Synthetic Data

	Bayesian-GT	Bayesian-Sample	SDNLL
class 1	9.6%	11.2%	18%
class 2	2.6%	2%	2%
class 3	2.8%	3.2%	4%
class 4	4.4%	3.8%	5%
class 5	2.8%	2.6%	2.8%
class 6	2.8%	3%	3.8%

Fig. 3. Face images from Weizmann Institute, all the combinations of three lightings, two expressions, and five orientations.

was constructed from 28 human subjects, each having 30 images with all possible combinations of two different expressions under three different lighting conditions with five different orientations. An example of the face images from one human subject is shown in Fig. 3.

As mentioned in Section 2.1, there are three mapping methods for a classification problem, we used the third mapping method (class mean in X space as the label) for the face recognition problem. It is worth noting that we chose δy very small in this experiment. This makes the first q prototypes to be the cluster centers and the leaf nodes have only samples with the same class label (pure node).

We applied the leave-one-out cross validation method to test this image data set. We first used the first view of each subject as testing samples and the rest of them as training samples. So there are 28 images as the testing samples for the first run. The second run used the second view of each subject as the testing samples. This results in the other 28 images count in the testing samples. We followed the same fashion for 30 runs. The total number of testing images is $28 \times 30 = 840$. Table 5 compares different appearance-based methods. For the principal component analysis (PCA), the number of eigenvectors used is determined by keeping 95 percent of the total sample variance. This gives 127 eigenvectors for PCA. A PCA tree is a binary classification tree where each node uses PCA.

Further, we compared the error rate of the proposed HDR algorithm with some major tree algorithms. CART[2] and C5.0 are among the best known classification trees. However, like most other decision trees, they are univariate trees in that each internal node used only one input component to partition the samples. This means that the partition of samples is done using hyperplanes that are

2. We have experimented with the same data set using CART implemented by OC1. The performance is much worse than those reported in Table 5. See CART for FERET set in Table 7.

TABLE 5
The Performance for Weizmann Face Data Set

Method	Error rate	Avg. testing time (msec)
PCA	0.95%	290
PCA tree	1.79%	76
LDA	0.00%	110
NN	1.31%	370
C5.0 with PCA	27.98%	197
OC1 with PCA	37.62%	203
HDR	0.00%	82

TABLE 6
The Performance for Weizmann Face Data Set

Method	Error rate	Avg. testing time (msec)
PCA	12.8%	115
PCA tree	14.58%	34
LDA	2.68%	105
NN	12.8%	164
SVM with PCA	12.5%	90
HDR	1.19%	78

orthogonal to one axis. We do not expect this type of tree can work well in a high-dimensional space for highly correlated multimedia data like images. Thus, we also tested a more recent multivariate tree OC1. We realize that these trees were not designed for high-dimensional spaces like those from images. We also tested the corresponding versions by performing PCA before using CART, C5.0, and OC1 and call them CART with PCA, C5.0 with PCA, and OC1 with PCA, respectively.

As shown in Table 5, LDA shares the best performance with our new HDR method in this test. However, the new HDR method is faster than LDA and has a more compact representation. The speed difference will be more significant when the data set is much larger.

The same data set was divided into training set and testing set for the other comparison. The training set contains 504 face images. Each subject contributed 18 face images in the training set which includes three different poses, three different lightings, and two different expressions. The remaining 336 images were used for the testing set. Each subject had 12 images for testing, which include two different poses, three different lightings, and two expressions.

Table 6 compares several methods. PCA is faster than nearest neighbor (NN) and shares a similar accuracy. However, the 95 percent of variance used by PCA results in about 98 eigenvectors which are much less than that of NN (5,632D!). PCA organized with a binary tree was faster than straight NN, as shown in Table 6. It is the fastest algorithm among all the methods we tested but the performance is worse than those of PCA and NN. The accuracy of LDA is the second best. The proposed HDR method is faster than LDA and resulted in the lowest error rate.

We also applied support vector machines (SVM) [21], [22] to this image set to compare the performance. We used the SVM software obtained from Royal Holloway, University of London [23] for this experiment. We used the PCA of the face images as the input features for the SVM.[3] The best result we obtained by tuning the parameters of the software is reported in Table 6. The recognition rate of the SVM with PCA is similar to that of PCA alone. However, SVM with PCA is faster than PCA. This is because SVM has more compact representation and PCA alone needs to conduct linear search for every training sample.

To give an intuitive display about what are inside the HDR tree, we show in Fig. 4 the mean face images and the discriminating features with $q = 5$. Then, the dimensionality of the discriminating subspace is four.

We also performed two experiments using the FERET face data set [24]. We used the frontal views from the data set. There are 457 persons with frontal views, 33 with four

3. The software failed when we used the original image input with dimensionality 5,632.

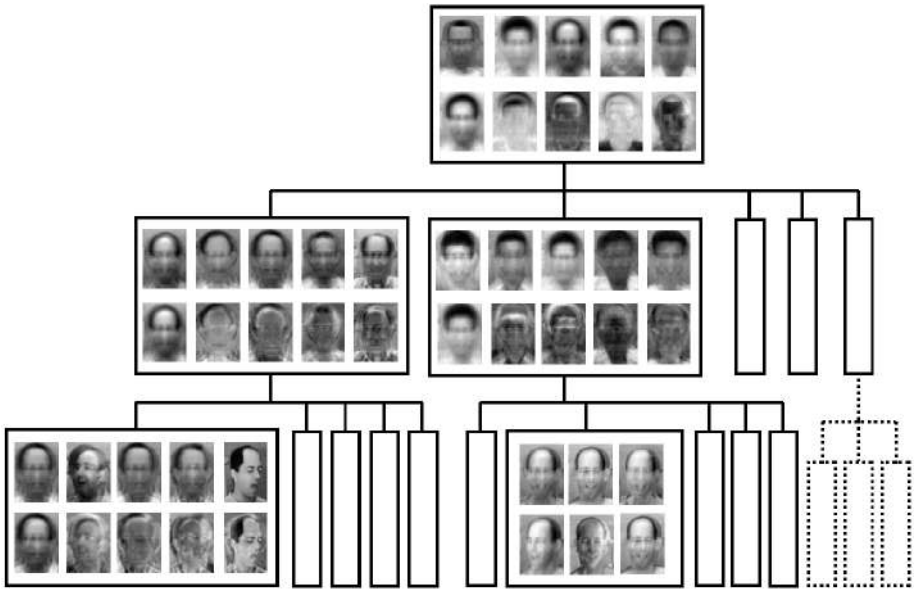

Fig. 4. An illustration of the HDR algorithm. The training images come from the Weizmann face data set. Each block indicates a tree node. The first row of each node shows the x-cluster centers presented as images. The first image of the second row is the grand mean of all the x-clusters. The remaining images of the second row are the discriminating features represented as images.

frontal images, one with six images, and the remaining 423 people having only two images each.

A face normalization program was used to translate, scale, and rotate each face image into a canonical image of 88 rows and 64 columns so that eyes are located at the prespecified positions, as shown in Fig. 5. To reduce the effect of background and nonfacial areas, image pixels are weighted by a function of the radical distance from the image center. Further, the image intensity is masked by a linear function so that the minimum and maximum values of each image are zero and 255, respectively. Fig. 5 shows the effect of such a series of transformations.

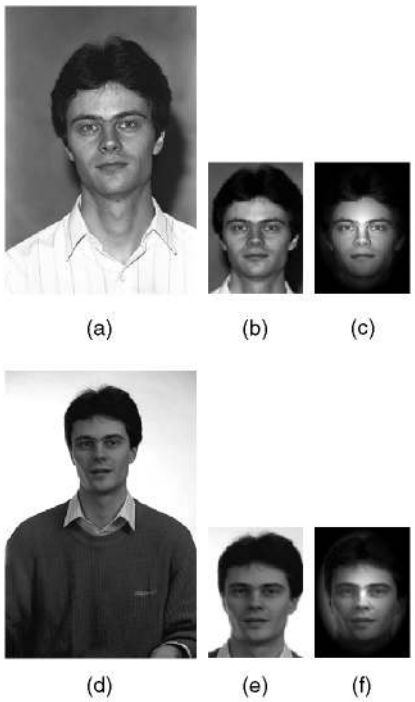

Fig. 5. The demonstration of the image normalization process. (a) and (d): The original images from the FERET data set. (b) and (e): The normalized images. (c) and (f): The masked images.

In the first experiment for the FERET data set 34 human subjects were involved. Each person had three face images for the purpose of training. The other face image was used for testing. We compare different options of the proposed algorithms. First, we used Euclidean distance in the discriminating subspace instead of SDNLL distance. With different choices of the number of x-clusters (q), we found that the performance does not significantly increase with the increase of q. Then we used SDNLL distance and the result is shown in Fig. 6. We found that the best q ($q = 18$ and beyond) resulted in 100 percent recognition rate.

Fig. 7a shows the depth of the HDR trees with different qs and distance metrics. All the options resulted in a similar tree height. The tree constructed using Euclidean distance

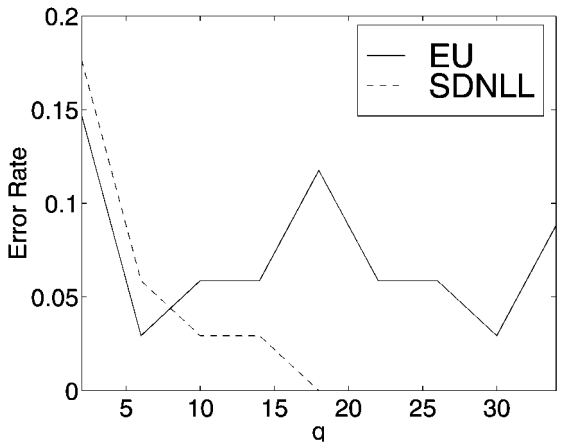

Fig. 6. The plot of error rate vs. number of x-clusters for FERET face test one using Euclidean distance and the new SDNLL distance.

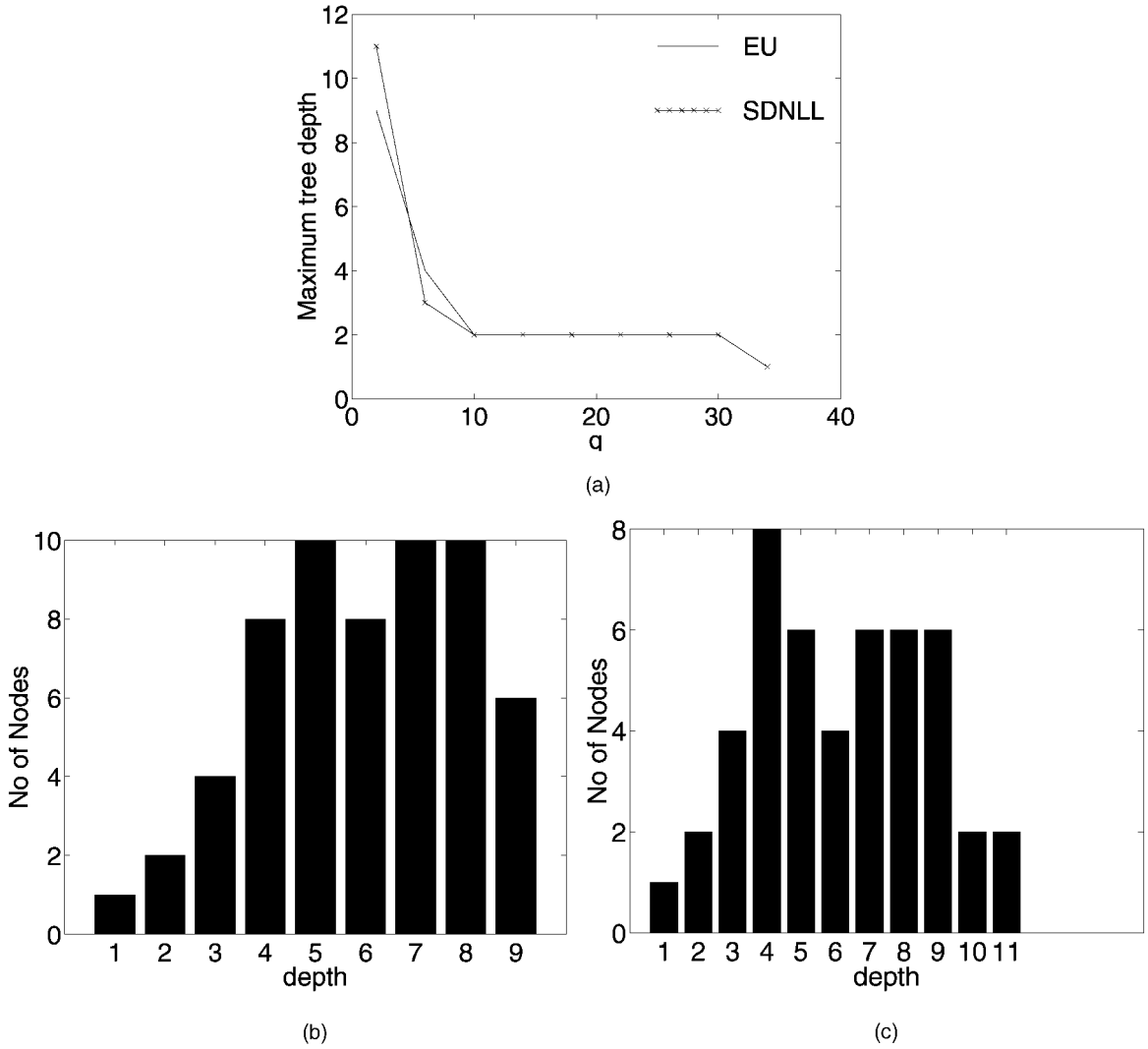

Fig. 7. The tree structures of FERET face test one. (a) The plot of depth of the tree vs. q for different distance options. (b) and (c): The plots of tree structures for different options with $q = 2$ for EU and SDNLL, respectively. EU: Euclidean Distance.

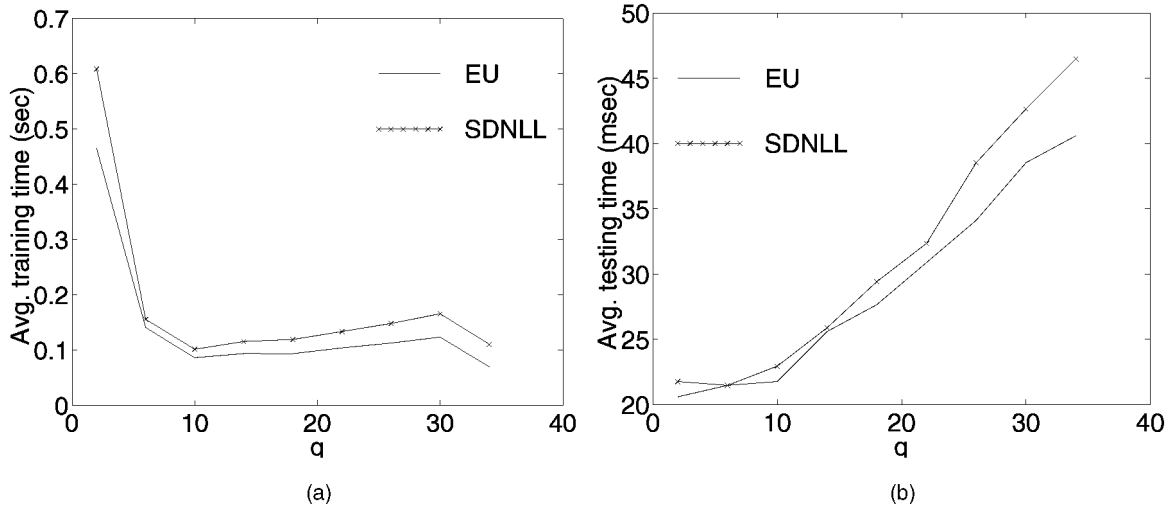

Fig. 8. The timing data of FERET face test one. (a) The plot of the average training time vs. q. (b) The plot of the average testing time vs. q. EU: Euclidean Distance.

has the most shallow depth. Fig. 7b and c give the nodes counts at every level of the trees for $q = 2$. It is worth noting that the structure of the trees affects the speed of the algorithm. As shown in Fig. 8b, a deeper tree results in a

TABLE 7
The Performance Comparison of Decision Trees for the FERET Test One

Method	Error rate		Time (sec)	
	Training	Testing	Training	Testing
CART	10%	53%	2108.00	0.029
C5.0	1%	41%	21.00	0.030
OC1	6%	56%	2206.00	0.047
CART with PCA	11%	53%	10.89	0.047
C5.0 with PCA	6%	41%	9.29	0.047
OC1 with PCA	5%	41%	8.89	0.046
HDR	0%	0%	12.25	0.027

faster tree retrieval because it works on a lower-dimensional space at each level. Fig. 8 indicates that the SDNLL distance metric does not require significantly more time to compute.

A summary of the performance comparison with some existing major tree classifiers is listed in Table 7. Notice that the training time is measured for the total time to train the corresponding system. The testing time is the average time per query. To make a fair comparison, the computation time for PCA is included in C5.0 with PCA, OC1 with PCA, and CART with PCA. As shown, none of the existing decision trees can deal with the FERET set acceptably well, not even the versions that use PCA as a preprocessing step.

The second experiment for the FERET data set used all the available data. Some decision tree programs used in Table 7 failed on this large data set, which prevented us from doing an extensive comparison as in Table 7. That was why we used a smaller data set in experiment one for comparison purpose in the first place. As described before, most of subjects have only two views in this large set. We used leave-one-out cross validation method. For each trial, one image was selected for each person for testing and the remaining images were used for training. Thus, the number of samples for each cluster is not equal. The error rates of the HDR method are plotted in Fig. 9. A characterization of the tree generated by the new HDR method is shown in Fig. 10. The speed of updating the tree on a SPARC 20 station is given in Fig. 11.

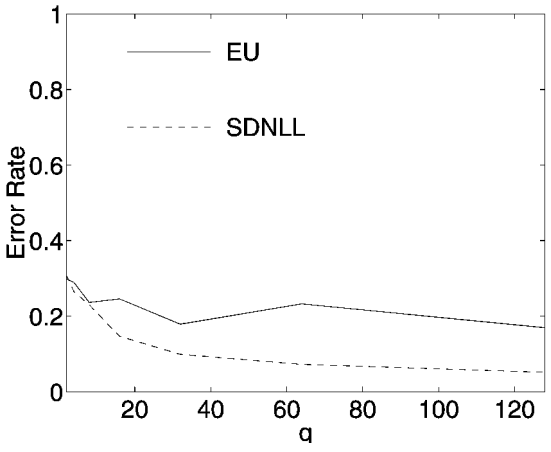

Fig. 9. The performance plots of FERET test two. The plots of error rate vs. number of x-clusters. EU: Euclidean Distance.

3.3 Experiments Using Data with Manually Extracted Features

We further investigated how our HDR algorithm performs on lower-dimensional real data, such as those publically available data sets that use human defined features. We reported the comparison results for two data sets from the StatLog project [25].

1. Letter image recognition data. There are 26 classes which corresponding to 26 capital letters. Each sample has 16 numeric features. 15,000 samples were used for training and 5,000 samples were used for testing.
2. Satellite image dataset. There are six decision classes representing different types of soils from satellite image. Each sample has 36 attributes. The training set includes 4,435 samples and the testing set includes 2,000 samples.

We inserted the performance of the new HDR algorithm to the results which were published in the StatLog project, as shown in Tables 8 and 9. For these lower-dimensional data sets, the performance of HDR algorithm is comparable with other best existing ones.

4 Conclusions

We cast both classification and regression problems into a unified regression framework. This allows us to design the new doubly-clustered method. The clusters in the output space provide coarse-to-fine virtual class labels for the clusters in the input space. To deal with high-dimensional input space, a different discriminating subspace is

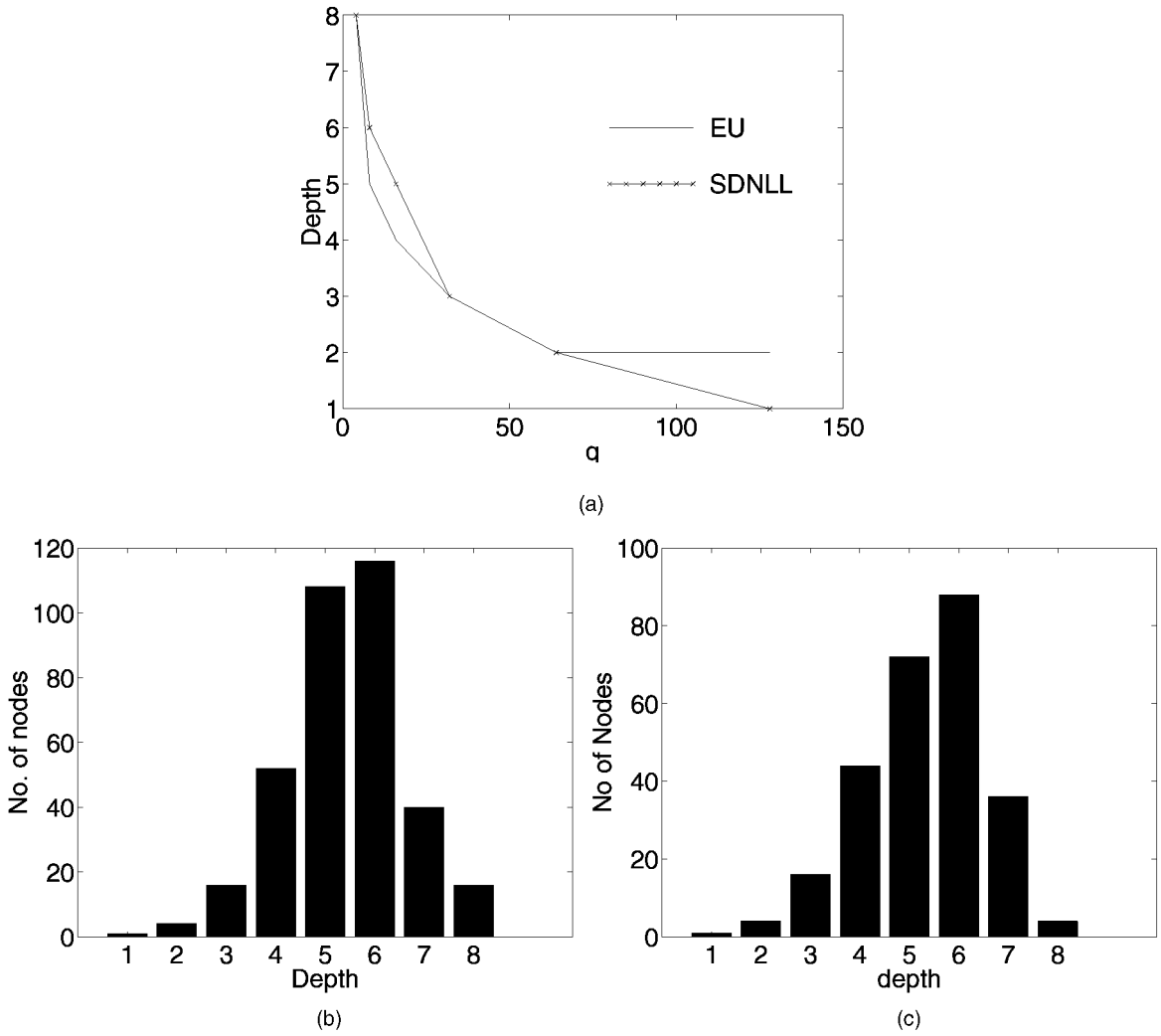

Fig. 10. The tree structures of FERET face test two. (a) the plot of depth of the tree vs q for different options. (b) and (c) are the plots of tree structures for Euclidean Distance and SDNLL distance with $q = 2$, respectively.

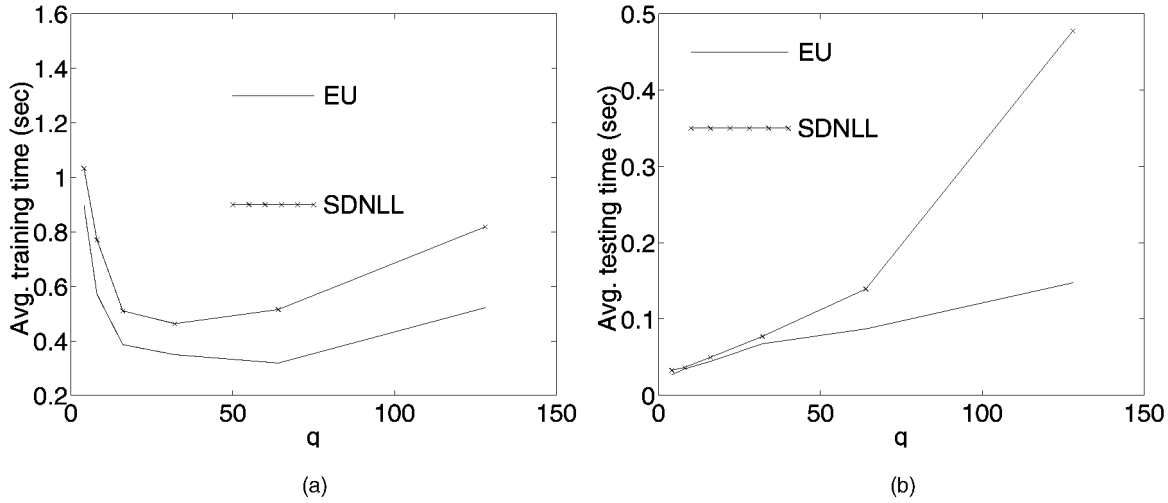

Fig. 11. The timing data of FERET face test two. (a) The plot of the average training time vs. q. (b) The plot of the average testing time vs. q. EU: Euclidean Distance.

automatically derived at each internal node of the tree. A size-dependent probability-based distance metric SDNLL is proposed to deal with large sample cases, small sample cases, and unbalanced sample cases according to Bayesian framework under local coarse Gaussian models. The global model, however, does not assume Gaussian distribution.

TABLE 8
Test Results on Letter Image Recognition Data

Algorithm	Error rate		Time (sec)	
	training	testing	training	testing
Alloc80	0.065	0.064	39575	?
KNN	0	0.068	15	2135
** HDR	0	0.070	212.7	30
LVQ	0.057	0.079	1487	48
QuaDisc	0.101	0.113	3736	1223
Cn2	0.021	0.115	40458	52
BayTree	0.015	0.124	276	7
NewId	0	0.128	1056	2
IndCart	0.010	0.130	1098	1020
C4.5	0.042	0.132	309	292
Dipol92	0.167	0.176	1303	80
Radial	0.220	0.233	?	?
LogDisc	0.234	0.234	5062	39
Ac2	0	0.245	2529	92
Castle	0.237	0.245	9455	2933
Kohonen	0.218	0.252	?	?
Cal5	0.158	0.253	1033	8
Smart	0.287	0.295	400919	184
Discrim	0.297	0.302	326	84
BackProp	0.323	0.327	277445	22
Bayes	0.516	0.529	75	18
Itrule	0.585	0.594	22325	69
Default	0.955	0.960	?	?
Cascade	1.0			
Cart	1.000			

TABLE 9
Test Results on Satellite Image Dataset

Algorithm	Error rate		Time (sec)	
	training	testing	training	testing
KNN	0.089	0.094	2105	944
LVQ	0.048	0.105	1273	44
** HDR	0	0.108	2.36	0.41
Dipol92	0.051	0.111	746	111
Radial	0.111	0.121	564	74
Alloc80	0.036	0.132	63840	28757
IndCart	0.023	0.138	2109	9
Cart	0.079	0.138	330	14
BackProp	0.112	0.139	72495	53
BayTree	0.020	0.147	248	10
NewId	0.067	0.150	226	53
Cn2	0.010	0.150	1664	36
C4.5	0.040	0.150	434	1
Cal5	0.125	0.151	764	7
QuaDisc	0.106	0.155	157	53
Ac2	?	0.157	8244	17403
Smart	0.123	0.159	27376	11
LogDisc	0.119	0.163	4414	41
Cascade	0.112	0.163	7180	1
Discrim	0.149	0.171	68	12
Kohonen	0.101	0.179	12627	129
Castle	0.186	0.194	75	80
Bayes	0.308	0.287	75	17
Default	0.758	0.769		
Itrule	?	100.00		

Our experimental study with synthetic data showed that the method can achieve near-Bayesian optimality for both low-dimensional data and high-dimensional data with low-dimensional data manifolds. With the help of the new decision tree, the retrieval time for each sample is of a logarithmic complexity for a bounded unbalanced HDR tree. The output of the system can be both class label or numerical vectors, depending on how the system trainer gives the training data. The experimental results have demonstrated that the algorithm can deal with a wide variety of sample sizes with a wide-variety of dimensionality.

APPENDIX A
LINEAR MANIFOLD

Given a set of vectors $V = \{v_1, v_2, ..., v_n\}$, which is a subset of a vector space $\mathcal{X}$. We want to express the subspace that passes the head tips of the vectors in V.

For numerical stability, we use the center of the vectors in V,

$$\bar{v} = \frac{1}{n}\sum_{i=1}^{n} v_i$$

and define the set of scatter vectors from their center: $s_i = v_i - \bar{v},\ i = 1, 2, ..., n$. These n scatter vectors are not linearly independent because their sum is equal to a zero

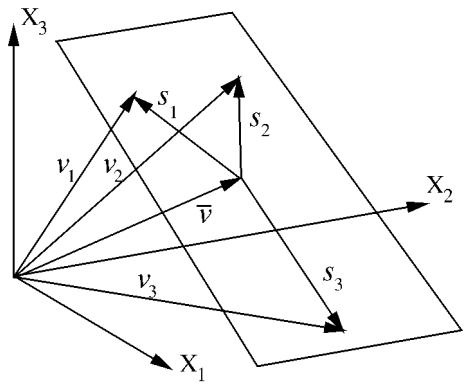

Fig. 12. The linear variety (hyperplane) that passes through the head points of the vectors. It can be represented by $\bar{v} + \text{span}(S)$, the spanned space from scatter vectors translated by the center vector $\bar{v}$.

vector. Let S be the set that contains these scatter vectors: $S = \{s_i \mid i = 1, 2, ..., n\}$. The subspace spanned by S, denoted by $\text{span}(S)$, consists of all the possible linear combinations from the vectors in S.

A translation of a subspace is called a linear manifold [26] (also called linear variety [27]). The subspace M translated to vector v_0 is denoted by $v_0 + M$: $v_0 + M = \{v_0 + m \mid m \in M\}$. Thus, the subspace that passes the head tips of the vectors in S can be represented by the linear manifold $\mathcal{D} = \bar{v} + \text{span}(S)$, as shown in Fig. 12.

The orthonormal basis $a_1, a_2, ..., a_{n-1}$ of the subspace $\text{span}(S)$ can be constructed from the radial vectors $s_1, s_2, ..., s_n$ using the *Gram-Schmidt Orthogonalization* (GSO) procedure:

Procedure 4. GSO Procedure: Given vectors $s_1, s_2, ..., s_{n-1}$, compute the orthonormal basis vectors $s_1, s_2, ..., s_{n-1}$.

1. $a_1 = s_1/\|s_1\|$.
2. For $i = 2, 3, ..., n-1$, do the following
   a. $a_i' = s_i - \sum_{j=1}^{i-1}(s_i^T a_j)a_j$.
   b. $a_i = a_i'/\|a_i'\|$.

In the above procedure, a degeneracy occurs if the denominator is zero. In the first step, the generacy means s_1 is a zero vector. In the remaining steps, it means that the corresponding vector s_i is a linear combination of the previous radial vectors. If a degeneracy occurs, the corresponding s_i should be discarded in the basis computation. The number of basis vectors that can be computed by the GSO procedure is the number of linearly independent radial vectors in S.

Given a vector $x \in \mathcal{X}$, we can compute its scatter part $s = x - \bar{v}$. Then compute the projection of x onto the linear manifold. It's ith coordinates in the orthonormal basis is given by $\beta_i = s^T a_i$, $i = 1, 2, ..., n-1$. We call the vector $f = (\beta_1, \beta_2, ..., \beta_{n-1})^T$ the feature vector of x in the linear manifold S.

Appendix B

Cholesky Decomposition

Procedure 5. Cholesky factorization: Given an $n \times n$ positive definite matrix $A = [a_{ij}]$, compute lower triangular matrix $L = [l_{ij}]$ so that $A = LL^T$.

For $i = 1, 2, ..., n$ do

1. For $j = 1, 2, ..., i-1$ do
$$l_{ij} = (a_{ij} - \sum_{k=1}^{j-1} l_{ik}l_{jk})/l_{jj}.$$
2. $l_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2}$.

Appendix C

Time Complexity of the HDRT Retrieval Algorithm

Theorem 1. *Given n training samples, the time complexity for a retrieval from a Bounded Unbalanced HDR tree is $O(2qd \log n / \log(1/\alpha))$, where α is the Unbalanced Bound of the tree and d is the dimensionality of the input space $\mathcal{X}$. The time complexity is $O(\log(n))$ if d is considered as a constant.*

Proof. Suppose a node N of the tree is assigned with $n_1 + n_2 + \cdots + n_q$ samples, where n_i is the number of samples assigned to the ith child of N. Rank these n_i's so that $n_1 \geq n_2 \geq \cdots \geq n_q$. Because the tree is a Bounded Unbalanced Tree, by definition we know that $n_1 \leq \alpha(n_1 + n_2 + \cdots + n_q)$, and is true for all the nodes of the tree.

In other words, each deeper level of the tree reduces the number of samples by a factor of at least α. The lth level down the tree will receive at most $n\alpha^l$ samples. In the worst case, we have just a single sample at tree height h (thus, the largest height possible.) Then $n\alpha^h \leq 1$, and $\alpha^h \leq (1/n)$. Then the height of the tree

$$h \leq \log_{(1/\alpha)} n = (\log n / \log(1/\alpha)).$$

For the time complexity of the operation in a node N, the HDR retrieval algorithm first subtracts the grand cluster mean from the test sample x and then projects the scatter vector to the $q-1$ dimensional feature space. The computational cost for these operations is $d + 2d(q-1)$. The computational cost of the SDNLL to each X-cluster is about $(q-1) + (q-1)^2$, which involves subtracting the cluster mean and a backsubstitution operation. The total computational time for a given test sample in a node thus is about $d + 2d(q-1) + q((q-1) + (q-1)^2)$. Since typically we have $d \gg q$, the computational cost in a node can be approximated by $2qd$. The retrieval time can be estimated by

$$T = h \times (2qd) = (2qd / \log(1/\alpha)) \log n = O(\log(n)).$$

. □

ACKNOWLEDGMENTS

The authors would like to thank Hamid Alavi for his efforts in programming the image transformation procedures. They would like to acknowledge the contribution of the SVM softwares from Royal Holloway, University of London [23]. The work is supported in part by the US National Science Foundation under grant IIS 9815191, DARPA ETO under contract DAAN02-98-C-4025, DARPA ITO under grant DABT63-99-1-0014, and research gifts from Siemens Corporate Research and Zyvex.

REFERENCES

[1] Y. Lamdan and H.J. Wolfson, "Geometric Hashing: A General and Efficient Model-Based Recognition Scheme," *Proc. Second Int'l Conf. Computer Vision,* pp. 238–249, 1988.
[2] M. Bichsel, "Strategies of Robust Object Recognition for the Automatic Identification of Human Faces," doctoral thesis, Eidgenössischen Technischen Hochschule Zürich, no. 9,467, 1991.
[3] K. Ikeuche and T. Kanade, "Automatic Generation of Object Recognition Programs," *Proc. IEEE,* vol. 76, no. 8, pp. 1,016–1,035, 1988.
[4] D.J. Kriegman and J. Ponce, "On Recognizing and Positioning Curved 3-D Objects from Image Contours," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 12, no. 12, pp. 1,127–1,137, 1990.
[5] W. Eric and L. Grimson, *Object Recognition by Computer: The Role of Geometric Constraints.* MIT Press, 1990.
[6] M. Turk and A. Pentland, "Eigenfaces for Recognition," *J. Cognitive Neuroscience,* vol. 3, no. 1, pp. 71–86, 1991.
[7] H. Murase and S.K. Nayar, "Visual Learning and Recognition of 3-D Objects from Appearance," *Int'l J. Computer Vision,* vol. 14, no. 1, pp. 5–24, Jan. 1995.
[8] J. Weng, "Cresceptron and SHOSLIF: Toward Comprehensive Visual Learning," *Early Visual Learning,* S.K. Nayar and T. Poggio, eds., New York: Oxford Univ. Press, 1996.
[9] A. Pentland, B. Moghaddam, and T. Starner, "View-Based and Modular Eigenspaces for Face Recognition," *Proc. IEEE Conf. Comp. Vision Pattern Recognition,* pp. 84–91, June 1994.
[10] D.L. Swets and J. Weng, "Using Discriminant Eigenfeatures for Image Retrieval," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 18, no. 8, pp. 831–836, 1996.
[11] G.R. Dattatreya and L.N. Kanal, "Decision Tress in Pattern Recognition," *Progress in Pattern Recognition,* L. Kanal and A. Rosenfeld, eds., pp. 189–239, 1985.
[12] S.R. Safavin and D. Landgrebe, "A Survey of Decision Tree Classifier Methodology," *IEEE Trans. Systems, Man and Cybernetics,* vol. 21, no. 3, pp. 660–674, May/June 1991.
[13] S.K. Murthy, "Automatic Construction of Decision Trees from Data: A Multidisciplinary Survey," *Data Mining and Knowledge Discovery,* vol. 2, no. 4, pp. 345–389, 1998.
[14] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees.* New York: Chapman & Hall, 1993.
[15] D. Swets and J. Weng, "Discriminant Analysis and Eigenspace Partition Tree for Face and Object Recognition from Views," *Proc. Int'l Conf. Automatic Face- and Gesture-Recognition,* pp. 192–197, Oct. 1996.
[16] D.L. Swets and J. Weng, "Hierarchical Discriminant Analysis for Image Retrieval," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 21, no. 5, pp. 386–401, 1999.
[17] K. Fukunaga, *Introduction to Statistical Pattern Recognition,* second ed. New York: Academic Press, 1990.
[18] S.S. Wilks, *Mathematical Statistics.* New York: Wiley, 1963.
[19] G.H. Golub and C.F. van Loan, *Matrix Computations.* Baltimore, Ma.: Johns Hopkins Univ. Press, 1989.
[20] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, *Numerical Recipes.* New York: Cambridge Univ. Press, 1986.
[21] V. Vapnik, *The Nature of Statistical Learning Theory.* New York: Springer-Verlag, 1995.
[22] V. Cherkassky and F. Mulier, *Learning from Data: Concepts, Theory, and Methods.* New York: Wiley, 1998.
[23] C. Saunders, M.O. Stitson, J. Weston, L. Bottou, B. Schlkopf, and A. Smola, "Support Vector Machine Reference Manual," Technical Report, CSD-TR-98-03, Royal Holloway, Univ. of London, Egham, UK, Mar. 1998, http://svm. cs. rhbnc. ac. uk/.
[24] P.J. Phillips, H. Moon, P. Rauss, and S.A. Rizvi, "The FERET September 1996 Database and Evaluation Procedure," *Proc. Int'l Conf. Audio and Video-Based Biometric Person Authentication,* Mar. 1997.
[25] *Machine Learning, Neural and Statistical Classification.* D. Michie, D.J. Spiegelhalter, and C.C. Taylor, eds., Ellis Horwood, 1994.
[26] E. Oja, *Subspace Methods of Pattern Recognition.* Letchworth, UK: Research Studies Press, 1983.
[27] D.G. Luenberger, *Optimization by Vector Space Methods.* New York: Wiley, 1969.

Wey-Shiuan Hwang received the BS degree from National Taiwan University, Taiwan, in 1990, and the MS and PhD degrees from Michigan State University, in 1995 and 1999, respectively, all in computer science. He is currently a research associate in the Department of Computer Science, Michigan State University, East Lansing. His research interests include computer vision, autonomous learning robots, and pattern recognition.

Juyang Weng received the BS degree from Fudan University, Shanghai, China, in 1982, and the MS and PhD degrees from University of Illinois, Urbana-Champaign, in 1985 and 1989, respectively, all in computer science.

From Jan. 1989 to Sept. 1990, he was a researcher at Centre de Recherche Informatique de Montréal, Montreal, Quebec, Canada, while adjunctively with Ecole Polytechnique de Montréal. From Oct. 1990 to August 1992, he held a visiting research assistant professor position at University of Illinois, Urbana-Champaign. In August, 1992, he joined the Department of Computer Science, Michigan State University, East Lansing, where he is now an associate professor. He coauthored the book (with T.S. Huang and N. Ahuja) *Motion and Structure from Image Sequences* (Springer-Verlag, 1993). His current research interests include mental development, computer vision, autonomous navigation, human-machine interface using vision, speech, gesture, and actions. Recently, he has been pursuing a new research direction called developmental robots—robots that can autonomously develop their own cognitive and behavioral capabilities through online, real-time interactions with their environments, including human teachers, using their sensors (e.g., vision, audition, touch) and effectors (e.g., speech, locomotion, manipulatory effectors).