An Incremental Learning Algorithm with Automatically Derived Discriminating Features*

Juyang Weng and Wey-Shiuan Hwang
Department of Computer Science
Michigan State University
East Lansing, MI 48823, USA

ABSTRACT

We propose a new technique which incrementally derive discriminating features in the input space. This technique casts both classification problems (class labels as outputs) and regression problems (numerical values as outputs) into a unified regression problem. The virtual labels are formed by clustering in the output space. We use these virtual labels to extract discriminating features in the input space. This procedure is performed recursively. We organize the resulting discriminating subspace in a coarse-to-fine fashion and store the information in a decision tree. Such an incrementally hierarchical discriminating regression (IHDR) decision tree can be realized as a hierarchical probability distribution model. We also introduce a sample size dependent negative-log-likelihood (NLL) metric to deal with large-sample size cases, small-sample size cases, and unbalanced-sample size cases. This is very essential since the number of training samples per class are different at each internal node of the IHDR tree. We report experimental results for two types of data: face image data along with comparison with some major appearance-based method and decision trees, hall way images with driving directions as outputs for the automatic navigation problem – a regression application.

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, fast and correctly retrieve, and easily manage images in the database [10].

An essential issue for image database is the representation of the image. We can categorize the image representation into two types: the model-based system and the appearance-based system. The model-based approach uses manually defined features to represent objects in the images. A lot of efforts has been made on this paradigm [2] [5] [6]. The focus of this approach is to design an efficient algorithm from a set of manually selected features. The strength of the method 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 examples, given a face image database, the designer needs to manually find the features for faces. The face features become be useless when a car image database is presented. The designer then needs to find another set of features for car images. Such kind 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 [13] [7]. 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, NN classifier is very time and space consuming for high-dimensional image space and 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 [13]. 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) can well distinguish different classes and thus are better for the purpose of classification, provided that the samples contain sufficient information [12].

The second issue for image database is how to organize the represented features so that the retrieval is both fast and accurate. Linear search is very time-consuming and makes it impractical. One way to solve this problem is to use a decision tree. A well designed decision tree can retrieve a matched sample with a logarithmic time complexity. This is a very useful characteristic for large image databases. There is a very rich literature about decision trees, see surveys [8] [3]. The applications of decision tree 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. Appearance-based approach drastically changed this situation. Traditional decision trees for low-dimensional input space have been found not suited for input dimensionality of a few thousands and up, even after data-dimensional reduction using techniques such as PCA, as we report in this paper. The major reason is the high complexity of sample distribution that cannot be captured by a single-level PCA. As demonstrated by [11], if a different subspace is computed at each internal node of the tree, a better generalization power results.

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 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 an 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 application, where c_{ij} is the cost of confusing classes i and j, one can embed n class labels into an $(n-1)$-dimensional Euclidean outputs pace by assigning vector y_i to class i, $i = 1, 2, \ldots, n$, so that $||y_i - y_i||$ is as close to as c_{ij}, as much as possible. This process is not always possible since a pre-defined cost matrix $[c_{ij}]$ is not always easy to provide. (2) Canonical mapping. Map n class labels into an n dimensional output space so that the i-th class label corresponds to a vector in which the i-th component is 1 and all other components are zeros. After this mapping, the distance between any two different class labels is the same: 1. 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 explained below. (3) Mapping labels into 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 as that between different classes. In many application, it is a desirable property. 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.

One the other hand, 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 general regression problem is more general than a classification problem.

2.2. 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 since the within-class scatter and between-class scatter matrices are all defined. However, if y_i is a numerical output, which can take any value for each component, it is challenge to perform discriminant analysis effectively.

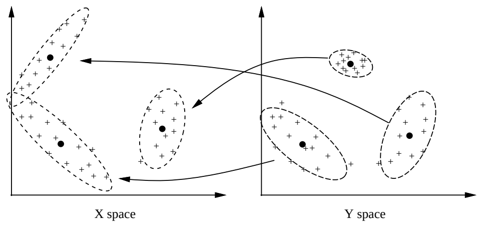

Figure 1. Y-clusters in space $\mathcal{Y}$ and the corresponding x-clusters in space $\mathcal{X}$. The first and the second order statistics are updated for each cluster. By default, the normalized Mahalanobis distance is used for x-cluster and the Euclidean distance is used for distance to y-cluster.

We introduce a new hierarchical statistical modeling method to address this challenge. Consider the mapping $h : \mathcal{X} \mapsto \mathcal{Y}$, which is to be approximated by a regression tree[1], called hierarchical discriminant regression (HDR) tree, for the high dimensional space $\mathcal{X}$. Our goal is to automatically generate discriminant features although no class label is available (other than the numerical vectors in space $\mathcal{Y}$). We must process each sample (x_i, y_i) to update the HDR tree using only a minimal amount of computation.

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 (e.g., $q = 6$) clusters of each type at each node. The q y-clusters determine the virtual class label of each 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 finer approximation is required. At each node, y in (x, y) finds the nearest y-cluster in Euclidean distance. This y-cluster indicates that which corresponding x-cluster to which the input (x, y) belongs. Then, the x part of (x, y) is used to compute the statistics of the x-cluster (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 the x-cluster, whose probability distribution is modeled as a multidimensional Gaussian at this level. A total of q centers of the q x-clusters give $q-1$ discriminant features which span $(q-1)$-dimensional discriminant space. A probability-based distance (to be discussed in Section 2.3) from x to each of the q x-clusters is used to determine which x-cluster should be further searched. If the probability is high enough, the

[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 [3].

sample (x, y) should further search the corresponding child (may be 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 orders of statistics are used to represent the clusters. For example, each y-cluster keeps the mean vector and the diagonal covariance matrix depending on the distance metric 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 a 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 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, among the samples in $S = \{(x_i, y_i) \mid x_i \in \mathcal{X},\ y_i \in \mathcal{Y},\ i = 1, 2, \ldots, n\}$, 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 i-th y-cluster, then x_i belongs to i-th x-cluster.*
2. *Compute the mean and covariance matrix 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.*
   - *Let the sample (x_i, y_i) belong to cluster j.*
4. *For each cluster j, now we have a portion of samples S_j that belong to the x-cluster. 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 δ_y represents the sensitivity of the HDR tree in $\mathcal{Y}$ space.*

Procedure 2 `Clustering-Y`*: Given a set of output vectors $Y = (y_1, y_2, \ldots, y_n)$, return p y-clusters. $p \le q$, where q represents the maximum 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 $d = dist(y_i, Y_j)$.*
   - (c) *If $d \ge \delta_y$ 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 by using 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 for query the HDR tree for an unknown sample x is described in below.

Procedure 3 `Retrieval`*: Given a HDR tree T and sample x, estimate the corresponding output vector y. 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 reached leaf nodes. Output the corresponding mean of its y-cluster as the estimated output y for x.*

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 Gaussian (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.

2.3. Distance in discriminating space

In the above SubSection, 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. 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 $\mathcal{X}$ is 3000, for example, each covariance matrix requires $3000 \times 3000 = 9,000,000$ numbers! We adopt a more efficient method.

As explained in Section 2.2, each internal node keeps up to q x-clusters. The centers of these q x-clusters are denoted by

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

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 space $\mathcal{Y}$. In this space, we can compute the between-cluster scatter and within-cluster scatter. Suppose that the number of samples in cluster i is n_i and the grand total of samples is $n = \sum_{i=1}^{q} n_i$. The mean of the cluster center, denoted by c is computed as

$$c = \frac{1}{n} n_i c_i.$$

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 within-cluster scatter matrices:

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

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

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

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 [4] 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 discriminant space $\mathcal{D}$, we will use size-dependent negative-log-likelihood (SDNLL) distance as discussed in Section 2.4 to take care of the reliability of each dimension in $\mathcal{D}$.

2.4. Size-dependent negative-log-likelihood

Let us consider the negative-log-likelihood (NLL) defined from Gaussian density of dimensionality $q - 1$:

$$\begin{aligned} L(x, c_i) \;\; = \;\; & \tfrac{1}{2}(x - c_i)^T \Gamma_i^{-1} (x - c_i) + \\ & \tfrac{q-1}{2} \ln(2\pi) + \tfrac{1}{2} \ln(|\Gamma_i|). \end{aligned} \qquad (4)$$

We call it Gaussian NLL for x to belong to the cluster of c_i. We call it Mahalanobis NLL if Γ_i is replaced by the within-class scatter matrix of each node — the average of covariance matrices of q clusters. We call it Euclidean NLL if Γ_i is replaced by a scale matrix $\rho^2 I$.

Suppose that the input space is $\mathcal{X}$ and the discriminating subspace for an internal node is $\mathcal{D}$. The Euclidean NLL has only one parameter ρ to estimate. Thus it is the least demanding among the three NLL in the richness of observation. When very few samples are available for all the clusters, the Euclidean distance is the suited distance.

The Mahalanobis NLL uses within-class scatter matrix Γ computed from all the samples in all the q x-clusters. The number of parameters in Γ is $q(q-1)/2$, and thus,the Mahalanobis NLL requires more samples than the Euclidean NLL.

For Gaussian NLL, $L(x, c_i)$ in Eq.(4) uses the covariance matrix Γ_i of x-cluster i. It requires that each x-cluster has enough samples to estimate the $(q-1) \times (q-1)$ covariance matrix. It thus is the most demanding on the number of observations. Note that the decision boundary of the Euclidean NLL and the Mahalanobis NLL is linear and that by the Gaussian NLL is quadratic.

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 are 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. We would like to make an automatic transition when the number of samples increases. We define the number of samples n_i as the measurement of maturity for each cluster i. $n = \sum_{i=1}^{q} n_i$ is the total number of samples in a node.

For the three types of NLLs, we have three matrices, $\rho^2 I$, Γ, and Γ_i. Consider the number of scales received to estimate each parameters, called number of scales per parameter (NSPP), of the element of the matrices. The NSPP for $\rho^2 I$ is $(n-1)(q-1)$, since the first sample does not give any estimate of the variance and each independent vector contains $q-1$ scales. For the Mahalanobis NLL, there are $(q-1)q/2$ parameters to be estimated in the (symmetric) matrix Γ. 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. The NSPP for the matrix Γ is $\frac{(n-q)(q-1)}{(q-1)q/2} = \frac{2(n-q)}{q}$. To avoid the value to be negative when $n < q$, we take NSPP for Γ to be $\max\left\{\frac{2(n-q)}{q}, 0\right\}$. Similarly, the NSPP for Γ_i for the Gaussian NLL is $\frac{1}{q}\sum_{i=1}^{q} \frac{2(n_i - 1)}{q} = \frac{2(n-q)}{q^2}$.

A bounded NSPP is defined to limit the growth of NSPP so that other matrices that contain more scalars can take over when there are a sufficient number of samples for them. Thus, the bounded NSPP for $\rho^2 I$ is $b_e = \min\{(n-1)(q-1), n_s\}$, where n_s denotes the switch point for the next more complete matrix to take over. Similarly, the bounded NSPP for for Γ is $b_m = \min\left\{\max\left\{\frac{2(n-q)}{q}, 0\right\}, n_s\right\}$. It is worth noting that the NSPP 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. Thus the bounded NSPP for Gaussian NLL is $b_g = \frac{2(n-q)}{q^2}$.

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

$$W_i = w_e \rho^2 I + w_m \Gamma + w_g \Gamma_i \qquad (5)$$

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 1: $b = b_e + b_m + b_g$. Using this size-dependent scatter matrix W_i, the *size-dependent negative log likelihood* (SDNLL) for x to belong to the x-cluster with center c_i is defined as

$$\begin{aligned} L(x, c_i) \;\; = \;\; & \tfrac{1}{2}(x - c_i)^T W_i^{-1} (x - c_i) + \\ & \tfrac{q-1}{2} \ln(2\pi) + \tfrac{1}{2} \ln(|W_i|). \end{aligned} \qquad (6)$$

3. The experimental results

We show the experimental results for face recognition problem as an example of classification application. For regression problem, we demonstrated the performance of our algorithm in autonomous navigation problem.

3.1. Experiments using real face data

Since our primary interest is in images which have a high dimensionality, we applied the new algorithm to appearance-based face image retrieval tasks. The experimental results on FERET face data set [9] are reported here. We used the frontal views from the data set. Thirty four human subjects were involved in this experiment. Each person had three face images for the purpose of training. The other face image was used for testing.

A face normalization program was used to translate, scale, and rotate each face image into a canonical image of 88 rows and 64 columns [1] so that eyes are located at pre-specified positions as shown in Fig 2.

To reduce the effect of background and non-facial areas, image pixels are weighted by a number that is 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 the images are 0 and 255, respectively. Fig 2 shows the effect of such a normalization procedure.

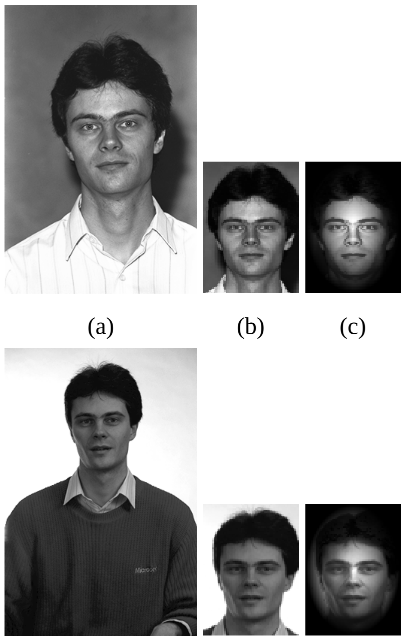

Figure 2. The demonstration of the image normalization process. (a), (e) The original image from the FERET data set. (b), (f) The normalized image. (c), (g) The masked image.

We compared the error rate of the proposed HDR algorithm with some major tree algorithms. CART 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 orthogonal to one axis. We do not expect this type of tree can work well in a high dimensional space. 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. A summary of comparison is listed in Table 1. 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 FERET set, not even the versions that use PCA as a preprocessing step.

Table 1. The performance of decision tree for FERET test

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 PCA	11%	53%	10.89	0.047
C5.0 PCA	6%	41%	9.29	0.047
OC1 PCA	5%	41%	8.89	0.046
HDR	0%	0%	23.41	0.041

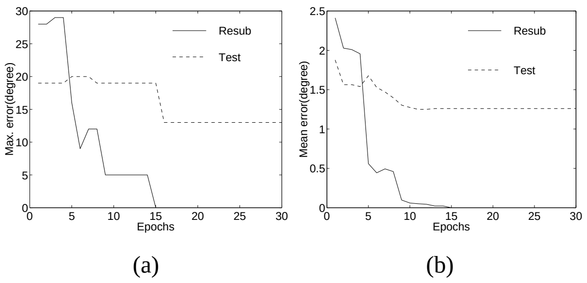

Figure 3. The performance of the autonomous navigation. (a) The plot for maximum error rates vs. epochs. (b) The plot for mean error rates vs. epochs. The solid line represents the error rates for resubstitution test. The dash line represents the error rates for the testing set.

3.2. Experiments with autonomous navigation problem

A vision-based navigation system accepts an input image X and outputs the control signal C to control the vehicle. The navigator can be denoted by an function f that maps the input image space $\mathcal{X}$ to control signal space $\mathcal{C}$. The learning process of the autonomous navigation problem then can be realized as a function approximation. This is a very challenging task since the function to be approximated is with very high dimensional input space and the real application requires the navigator to perform in real time.

We applied our algorithm to this challenging problem. Totally 318 images with the corresponding heading directions were used for training. The resolution of each image is 30 by 40. We used the other 204 images to test. Fig 3 shows the maximum error rates and mean error rates versus the number of epochs. Both maximum error and mean error converge around the 15th epoch. Fig 4 gives plots of the histograms of the error rates at different epochs. With the increase of the epochs, we observed the improvement of the maximum error and mean error.

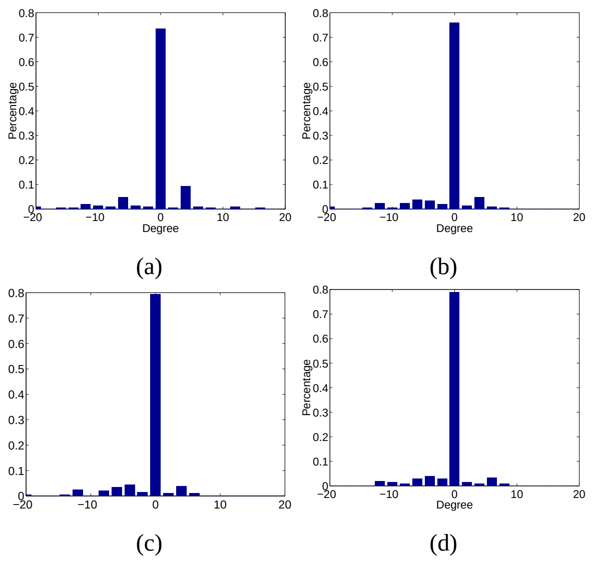

Figure 4. The histograms of the error rates. Plot (a), (b), (c), and (d) correspond to the histograms at epoch 1, 6, 11, 20, respectively.

4. Conclusion

We cast both classification and regression problems into a unified regression framework. This allows us to design the new doubly-clustered method. Clusters in output space provide coarse-to-fine virtual class labels from clusters in the input space. To deal with high-dimensional input space, a discriminating subspace is 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. 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 decision tree, the retrieval time for each sample is of logarithmic complexity. 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.

References

[1] Hamid Alavi. State self-organization for continuous video stream input, a master's project report. Technical report, Department of Computer Science, Michigan State University, 1997.

[2] Martin Bichsel. *Strategies of Robust Object Recognition for the Automatic Identification of Human Faces*. PhD thesis, Eidgenössischen Technischen Hochschule Zürich, 1991. Diss. ETH Number 9467.

[3] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Chapman & Hall, New York, 1993.

[4] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, New York, NY, second edition, 1990.

[5] K. Ikeuche and T. Kanade. Automatic generation of object recognition programs. *Proceedings of the IEEE*, 76(8):1016–1035, 1988.

[6] D. J. Kriegman and J. Ponce. On recognizing and positioning curved 3-D objects from image contours. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12(12):1127–1137, 1990.

[7] H. Murase and S. K. Nayar. Visual learning and recognition of 3-D objects from appearance. *Int'l Journal of Computer Vision*, 14(1):5–24, January 1995.

[8] S. K. Murthy. Automatic construction of decision trees from data: A multidisciplinary survey. *Data Mining and Knowledge Discovery*, 1998.

[9] P. J. Phillips, H. Moon, P. Rauss, and S. A. Rizvi. The FERET evaluation methodology for face-recognition algorithms. In *Proc. IEEE Conf. Comp. Vision Pattern Recognition*, pages 137–143, San Juan, Puerto Rico, June 1997.

[10] J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufman, San Mateo, CA, 1993.

[11] D. Swets and J. Weng. Discriminant analysis and eigenspace partition tree for face and object recognition from views. In *Proc. Int'l Conference on Automatic Face- and Gesture-Recognition*, pages 192–197, Killington, Vermont, Oct. 14-16 1996.

[12] D. L. Swets and J. Weng. Using discriminant eigenfeatures for image retrieval. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18(8):831–836, 1996.

[13] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.