An Incremental Learning Method for Face Recognition under Continuous Video Stream

Juyang Weng, Colin H. Evans, Wey-Shiuan Hwang
Department of Computer Science and Engineering
Michigan State University
East Lansing, MI 48824

Abstract

The current technology in computer vision requires humans to collect images, store images, segment images for computers and train computer recognition systems using these images. It is unlikely that such a manual labor process can meet the demands of many challenging recognition tasks. Our goal is to enable machines to learn directly from sensory input streams while interacting with the environment including human teachers. We propose a new technique which incrementally derives discriminating features in the input space. Virtual labels are formed by clustering in the output space to extract discriminating features in the input space. We organize the resulting discriminating subspace in a coarse-to-fine fashion and store the information in a decision tree. Such an incremental hierarchical discriminating regression (IHDR) decision tree can be modeled by a hierarchical probability distribution model. We demonstrate the performance of the algorithm on the problem of face recognition using video sequences of 33,889 frames in length from 143 different subjects. A correct recognition rate of 95.1% *has been achieved.*

1 Introduction

Traditionally, classification and regression trees use univariate split at each internal node, such as in CART [1], C5.0 [8] and many others. This means that the partition by each node uses a hyper-plane that are orthogonal to an axis in the input space X. Multivariate linear splits correspond to partition hyper-planes that are typically not orthogonal to any axis of the input space. Trees that use multivariate splits are called oblique tree. As early as in the mid 70's, Freidman [2] proposed a discriminating node-split for building a tree which would result in an oblique tree. OC1 by Murthy et al. [5] and SHOSLIF tree by Weng [9] are two methods for constructing oblique trees. For an extensive survey of decision trees, see a recent survey by Murthy [7]. OC1 uses iterative search to find a split. SHOSLIF uses principal component analysis (PCA) and linear discriminant analysis (LDA) to directly compute splits. Multivariate nonlinear splits correspond to curved partition surfaces with any orientation. The incremental hierarchical discriminant regression (HDR) tree proposed in this paper uses multivariate nonlinear splits, with multivariate linear splits as a special case.

As far as we know, there is no published incremental statistical method for constructing a regression tree for high dimensional input space based on discriminant analysis. By high dimensional space we mean that the dimensionality is above a few thousands and the number of samples can be smaller than the dimensionality [1]. This high dimensional issue becomes increasingly important with increased use of high dimensional digital multimedia data such as images and video where each pixel value is a component of the input vector [2]. These applications present a high degree of correlation among components of high-dimensional input, not matched by typical lower-dimensional human-prepared feature data. None of CART, C5.0, OC1 or other published tree classifiers that we know was designed for this high-dimensional, highly correlated input situation. SHOSLIF tree is for high input dimensionality and it has an incremental version [11], but it uses PCA only for splits. It is technically challenging to incremen-

[1] When the number of samples is smaller than the input dimensionality (i.e., the number of features), Brieman et al. [1] and Murthy [5] called the situation data underfits the concept and thus disregard the situation.

[2] This corresponds to a now well-accepted and highly successful approach called appearance-based approach, with which the human system designer does not define features at all but rather applies statistical methods directly to high-dimensional, preprocessed image frame [10] [6].

tally construct a classification or regression tree that uses discriminant analysis, due to the complex nature of the problem involved. Why is discriminant analysis important? Discriminant analysis uses information of the output space in addition to the information in the input space to compute the splits. PCA uses only information in the input space and cannot use information in the output space. Consequently, variations in the input space that is totally useless for output (e.g., pure noise components) will also be captured by PCA. On the other hand, various neural networks are incremental in training. They tend to build a network with some discriminating power. However, since neural networks typically do not use a statistical model, they suffer from local minima problem and thus give poorer performance as we will show later in this paper.

In this paper, we present an incremental way of constructing a regression tree that uses discriminant analysis. Further, we deal with the unbalanced sample problem in that some regions of input space may have a very large number of samples while other regions have only very sparse samples. A sample-size dependent likelihood measure is proposed to make suboptimal decision for different sample sizes, which is also very critical for an incremental algorithm which receive training data incrementally. We also require real-time speed of the regression system which is essential for many interactive applications and thus a hierarchical data pruning structure such as a tree is a must. We present experimental result to demonstrate the performance of the new technique and compare it with some major published methods.

2 The Method

2.1 Discriminating features from doubly clustering

Let us 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) [3] 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 input component, it is a challenge to figure out an effective discriminant analysis procedure that can disregard input components that are either irrelevant to output or contribute little to the output. 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, called incremental hierarchical discriminating regression (IHDR) tree,

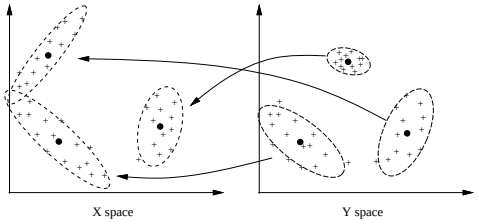

Figure 1. Y-clusters in space $\mathcal{Y}$ and the corresponding x-clusters in space $\mathcal{X}$. The first and the second order statistics are updated for each cluster.

for the high dimensional space $\mathcal{X}$. Our goal is to automatically derive discriminating features although no class label is available (other than the numerical vectors in space $\mathcal{Y}$). In addition, for real-time requirement, we must process each sample (x_i, y_i) to update the IHDR tree using only a minimal amount of computation.

Two types of clusters are incrementally updated at each node of the IHDR 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 arriving sample (x, y) based on its y 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 Euclidean distance and updates (pulling) the center of the y-cluster. This y-cluster indicates which corresponding x-cluster the input (x, y) belongs to. Then, the x part of (x, y) is used to update 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 current sample (x, y) to belong to the x-cluster, whose probability distribution is modeled as a multidimensional Gaussian at this level. In other words, each node models a region of the input space $\mathcal{X}$ using q Gaussians. Each Gaussian will be modeled by more small Gaussians in the next tree level if the current node is not a leaf node.

Moreover, the center of these x-clusters provide essential information for discriminating subspace, since these x-clusters are formed according to virtual labels in $\mathcal{Y}$ space. We define a discriminating subspace as the linear space that passes through the centers of these x-clusters. A total of q centers of the q x-clusters give $q - 1$ discriminating features which span $(q - 1)$-dimensional discriminating space. A probability-based distance called size-dependent negative-log-likelihood

(SNLL) [4] is computed from x to each of the q x-clusters 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.

The algorithm incrementally builds an IHDR tree from a sequence of training samples. The deeper a node is in the tree, the smaller the variances of its x-clusters are. When the number of samples in a node is too small to give a good estimate of the statistics of q x-clusters, this node is a leaf node.

2.2 Algorithm for the incremental hierarchical discriminating regression tree

The algorithm incrementally builds an IHDR tree from a sequence of training samples. The deeper a node is in the tree, the smaller the variances of its x-clusters are. When the number of samples in a node is too small to give a good estimate of the statistics of q x-clusters, this node is a leaf node. The following is the outline of the incremental algorithm for tree building (also tree retrieval when y is not given).

Procedure 1 `Update-node`*: Given a node N and (x, y) where y is either given or not given, update the node N using (x, y) recursively. Output: top matched terminal nodes. The parameters include: k which specifies the upper bound in the width of parallel tree search; δ_x the sensitivity of the IHDR tree in $\mathcal{X}$ space as a threshold to further explore a branch; and c representing if a node is on the central search path. Each returned node has a flag c. If $c = 1$, the node is a central cluster and $c = 0$ otherwise.*

1. *Find the top matched x-cluster in the following way. If $c = 0$ skip to step (2). If y is given, do (a) and (b); otherwise do (b).*
   (a) *Update the mean of the y-cluster nearest y in Euclidean distance by using amnesic averages. Update the mean and the covariance matrix of the x-cluster corresponding to the y-cluster by using amnesic average.*
   (b) *Find the x-cluster nearest x according to the probability-based distances. The central x-cluster is this x-cluster. Update the central x-cluster if it has not been updated in (a). Mark this central x-cluster as active.*
2. *For all the x-clusters of the node N, compute the probability-based distances for x to belong to each x-cluster.*
3. *Rank the distances in increasing order.*
4. *In addition to the central x-cluster, choose peripheral x-clusters according to increasing distances until the distance is larger than δ_x or a total of k x-clusters have been chosen.*
5. *Return the chosen x-clusters as active clusters.*

From the above procedure, we can observe the following points. (a) When y is given, the corresponding x-cluster is updated, although this x-cluster is not necessarily the one on the central path from which the tree is explored. Thus, we may update two x-clusters, one corresponding to the given y, the other being the one used for tree exploration. The update for the former is an attempt to pull it to the right location. The update for the latter is an attempt to record the fact that the central x-cluster has hit this x-cluster once. (b) No matter y is given or not, the x-cluster along the central path is always updated. (c) Only the x-clusters along the central path are updated, other peripheral x-clusters are not. We would like to avoid, as much as possible, storing the same sample in different brother nodes.

Procedure 2 `Update-tree`*: Given the root of the tree and sample (x, y), update the tree using (x, y). If y is not given, estimate y and the corresponding confidence. The parameters include: k which specifies the upper bound in the width of parallel tree search.*

1. *From the root of the tree, update the node by calling* `Update-node` *using (x,y).*
2. *For every active cluster received, check if it points to a child node. If it does, mark it inactive and explore the child node by calling* `Update-node`*. At most q^2 active x-clusters can be returned this way if each node has at most q children.*
3. *The new central x-cluster is marked as active.*
4. *Mark additional active x-clusters according to the smallest probability-based distance d, up to k total if there are that many x-clusters with $d \leq \delta_x$.*
5. *Do the above steps 2 through 4 recursively until all the resulting active x-clusters are all terminal.*
6. *Each leaf node keeps samples (x_i, y_i) that belong to it. The output is y_i if x_i is the nearest neighbor among these samples.*
7. *If the current situation satisfies the spawn rule, i.e. the number of samples exceeds the number required for estimating statistics in new child, the*

top-matched x-cluster in the leaf node along the central path spawns a child which has q new x-clusters. All the internal nodes are fixed in that their clusters do not further update using future samples so that their children do not get temporally inconsistent assignment of samples.

The above incrementally constructed tree gives a coarse-to-fine probability model. 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. This results in the well-known logarithmic time complex for tree retrieval: $O(\log m)$ where m is the number of leaf nodes in the tree, assuming that the number of samples in each leaf node is bounded above by a constant.

2.3 Amnesic average

The average of n input data $x_1, x_2, ..., x_n$ is given by

$$\bar{x}^{(n)} = \frac{1}{n}\sum_{i=1}^{n} x_i = \sum_{i=1}^{n}\frac{1}{n}x_i. \quad (1)$$

Therefore, each x_i receives the same weight $1/n$. This is called an equally weighted average. If x_i arrives incrementally and we need to compute the average for all the inputs received so far, it is more efficient to recursively compute the current average based on the previous average:

$$\bar{x}^{(n+1)} = \frac{n\bar{x}^{(n)} + x_{n+1}}{n+1} = \frac{n}{n+1}\bar{x}^{(n)} + \frac{1}{n+1}x_{n+1}. \quad (2)$$

The recursive equation Eq. (2) gives an equally weighted average. In amnesic average, the new input gets more weight than old inputs as given in the following expression:

$$\bar{x}^{(n+1)} = \frac{n-l}{n+1}\bar{x}^{(n)} + \frac{1+l}{n+1}x_{n+1}, \quad (3)$$

where l is a parameter.

The amnesic average can also be applied to the recursive computation of a covariance matrix Γ_x from incrementally arriving samples: $x_1, x_2, ..., x_n, ...$ where x_i is a column vector for $i = 1, 2, ...$ The unbiased estimate of the covariance matrix from these n samples $x_1, x_2, ..., x_n$ is given in a batch form as

$$\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})^T \quad (4)$$

with $n > 1$, where $\bar{x}$ is the mean vector of the n samples. Using the amnesic average, $\bar{x}^{(n+1)}$, up to the $(n+1)$-th sample, we can compute the amnesic covariance matrix up to the $(n+1)$-th sample as

$$\Gamma_x^{(n+1)} = \frac{n-1-l}{n}\Gamma_x^{(n)} + \frac{1+l}{n}(x_{n+1} - \bar{x}^{(n+1)})(x_{n+1} - \bar{x}^{(n+1)})^T \quad (5)$$

for $n > l + 1$. When $n \leq l + 1$, we may use the batch version as in expression (4). Even with a single sample x_1, the corresponding covariance matrix should not be estimated as a zero vector, since x_1 is never exact if it is measured from a physical event. For example, the initial variance matrix $\Gamma_x^{(1)}$ can be estimated as $\sigma^2 I$, where σ^2 is the expected digitization noise in each component and I is the identity matrix of the appropriate dimensionality.

3 The Experimental Results

We performed the online face recognition and gender recognition using image sequences. No existing learning algorithm can learn directly from unsegmented, unedited real video streams as we demonstrated in the following experiments. We used basic method in [12] to accomplish this. The new IHDR tree enables the method to handle a large number of classes. In terms of the number of image frames used, the experiments reported here were among the largest in the reported studies.

It is convenient to introduce simulated sensors and effectors. We call them numerical sensor and effector. A numerical sensor gives a vector of a predefined dimensionality at each time t. A numerical sensor can be realized by a graphic user interface, a digital dial, a joystick, a key board, a file, or anything that can convert a physical measurement into one of a set of predefined codes. A numerical effector is similar to the numerical sensor, except that it is for output.

The inputs to the IHDR algorithm contain two sensors: a video camera as the visual sensor and a numerical sensor as the simulated auditory sensor. Each vector at time t from the auditory sensor is considered the

feature vector of a sound segment from a microphone, although such feature vectors are manually defined for simplicity. The output of the IHDR depends on the tasks we trained. If the system is learning the identities from image sequence, the output is the name of the person. If it is learning gender, the output will be either "male" or "female."

To proceed, training and testing sessions are interleaved appropriately through the time, according to what the teacher wants. Each person presented himself or herself to the camera. During the presentation, we input the person's name, the gender. Questions about name ("who?") and gender ("gender?") were asked and the correct responses were imposed. The trainer has a simple language in mind. Each question and answer is represented by a pre-defined code in the numerical sensor and effector, respectively.

In the experiment, the system is introduced to 143 people one after another. During the teaching sessions and testing sessions, each person enters the scene, stays for a short time, and then exits the scene. The camera is fixated closely on the faces of each person so that face recognition is based on individual variations between faces and not on clothing, background, or position and occlusion of the face image. Fig. 2 illustrates a segment of the video stream used for training. To teach the system how to act after a question is asked, the appropriate action is imposed to the effector at the right context.

Figure 2. A temporally subsampled segment of the real-time video stream.

Each video sequence was around 50-60 frames in length, with a resolution of 88×64. In total, 33,889 images were collected across all of these sequences. The preprocessor of the visual camera includes normalization of each image frame so that the mean over all the pixels in each frame is zero and the variance of the image pixel intensities is 1. Each pixel in each image frame is weighted before being fed through as a part of the sensory input $x(t)$. The center pixel has more weight than those in the periphery to take into account the fact that human fovea has a higher visual acuity than the periphery. Zhang and his colleagues 1993 [13] explicitly investigated experimentally how the degree of "softness" of an aperture affects the segregation of figure and ground in human subjects.

Figure 3. These are the seven face identification errors made by the system. The first row shows the query image and and seconde row shows the corresponding retrieved image.

The experimental result for face recognition was 7 errors out of 143. That for gender identification was 4 errors out of 143. Table 3 gives the false identification images. The face identification and gender identification tasks were tested against a nearest neighbor classifier for comparison. The nearest neighbor classifier was provided with the individual training samples used only during the question-and-answer phases of training, and the "no action" images were not used. This made for a training set of 858 images. The nearest neighbor classifier had 4 errors out of 143 for face identification, and 2 errors out of 143 for gender recognition. This shows that the performance of the proposed architecture is comparable to a nearest neighbor classifier for this specific sequence of face recognition. Other more complicated data sets showed [9] that nearest neighbor classifier is not as good as the classifiers that use discriminant features.

Table 1. The recognition rate for the face sequence

Method	Face Results	Gender Results
IHDR algorithm	95.1% (Best)	97.2% (Best)
Nearest Neighbor	97.2%	98.6%

A major advantage of the proposed method over a nearest neighbor classifier is that it is fast, can be trained online, and is scalable to large data sets. The average training and testing time is illustrated in Table 2. The speed differences are impressive because the proposed algorithm was generated online, while the nearest neighbor classifier was generated in batch

with extra total knowledge about which of the 50,833 training samples to exclude. Additionally, the IHDR algorithm maintains a tree structure, which on average gives it retrieval times that are logarithmical to the number of samples stored in the tree. For real-time applications, the proposed algorithm is definitely a feasible method – fast, online, and scalable.

Table 2. The training and testing time for the face sequence

Method	Search Time	Training Time
IHDR	0.04s (Worst)	0.15s
Nearest Neighbor	0.2s	0s

4 Conclusion

We cast both classification and regression problems into a unified regression framework. This allows us to design the new doubly-clustered method, clustering in both output and input spaces. Clusters in the output space provide coarse-to-fine virtual class labels for samples in the input space. To deal with high-dimensional input space in which some components are not very useful and some can be very noisy, a discriminating subspace is incrementally 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. Our experimental study (not reported here) with the synthetic data has 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 decision tree, the retrieval time for each sample is of logarithmic complexity making real time performance a reality even for high-dimensional inputs. The output of the system can be both class label or numerical vectors, depending on how the system trainer gives the training data. The incremental building of the tree opens up the possibility of real-time interactive training where the number of training samples can be extremely large but the resulting tree does not need to store all the training samples.

References

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

[2] J. H. Friedman. A recursive partition decision rule for nonparametric classification. *IEEE Trans. on Computers*, 26:404–408, April 1977.

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

[4] W. Hwang, J. Weng, M. Fang, and J. Qian. A fast image retrieval algorithm with automatically extracted discriminant features. In *Proc. IEEE Workshop on Content-based Access of Image and Video Libraries*, pages 8–15, Fort Collins, Colorado, June 1999.

[5] S. K. M. S. Kasit and S. Salzberg. A system for induction of oblique decision trees. *Journal of Artificial Intelligence*, 2:1–33, Aug. 1994.

[6] 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.

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

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

[9] D. L. Swets and J. Weng. Hierarchical discriminant analysis for image retrieval. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 21(5):386–401, 1999.

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

[11] J. Weng and S. Chen. Incremental learning for vision-based navigation. In *Proc. Int'l Conference on Pattern Recognition*, Vienna, Austria, Aug. 1996.

[12] J. Weng, C. H. Evans, and W. S. Hwang. Automated animal-like learning for developing a face recognition system. In *Proc. 2nd Int'l Conf. on Audio and Visual-Based Person Authentication*, pages 49–54, Washington, DC, March 1999.

[13] J. Zhang, S. L. Yeh, and K. K. D. Valois. Motion contrast and motion integration. *Vision Research*, 33:2721–2732, 1993.