Unsupervised Learning of Boosted Tree Classifier using Graph Cuts for Hand Pose Recognition

Toufiq Parag
Computer Science
Rutgers University
tparag@cs.rutgers.edu

Ahmed Elgammal
Computer Science
Rutgers University
elgammal@cs.rutgers.edu

Abstract

This study proposes an unsupervised learning approach for the task of hand pose recognition. Considering the large variation in hand poses, classification using a decision tree seems highly suitable for this purpose. Various research works have used boosted decision trees and have shown encouraging results for pose recognition. This work also employs a boosted classifier tree learned in an unsupervised manner for hand pose recognition. We use a recursive two way spectral clustering method, namely the Normalized Cut method (NCut), to generate the decision tree. A binary boosting classifier is then learned at each node of the tree generated by the clustering algorithm. Since the output of the clustering algorithm may contain outliers in practice, the variant of boosting algorithm applied at each node is the Soft Margin version of AdaBoost, which was developed to maximize the classifier margin in a noisy environment. We propose a novel approach to learn the weak classifiers of the boosting process using the partitioning vector given by the NCut algorithm. The algorithm applies a linear regression of feature responses with the partitioning vector and utilizes the sample weights used in boosting to learn the weak hypotheses. Initial result shows satisfactory performances in recognizing complex hand poses with large variations in background and illumination. This framework of tree classifier can also be applied to general multi-class object recognition.

1 Introduction

Hand pose recognition is a non-trivial task with potential application in many areas such as human computer interaction, sign language interpretation etc. Hand pose recognition can be formulated as an instance of the general multi class object recognition problem which is a very important goal in computer vision. Since human hand is an articulated object with many degrees of freedom and since there are no obvious discriminative features such as colors, appearance landmarks, etc., recognizing hand pose appears to be very challenging. For efficient recognition, it is essential to find features that can discriminate different hand poses and to share these features in a hierarchical structure . Feature selection and feature sharing is a fundamental issue in multi class object recognition which is typically addressed in a supervised manner, e.g. [16]. However, in the case of hand pose recognition an unsupervised setting is favorable.

This paper proposes an unsupervised learning framework for learning a boosted classifier tree for multi class object recognition (for this particular study, hand pose). The framework facilitates feature selection and feature sharing among large sets of features through learning boosted classifiers directly from soft labels obtained through spectral clustering. We use spectral clustering, namely Normalized Cut [14] to obtain a top-down two-way hierarchical partitioning of the data. At each level of the tree generated by the hierarchical partitioning, we can train a classifier to discriminate the two sub-classes. However, we show that as a result of the spectral clustering, we can directly obtain weak classifiers which fit the obtained spectral partition. Such weak learner can then be boosted to form a classifier. As we can speculate, either or both of

the subsets generated by the clustering algorithm may contain impurities. Therefore, we need to apply a boosting algorithm that has been shown to work well in the presence of noise, e.g. [12]. Ideally, the leaves of this classifier tree should comprise only the images of the same object class (in this study, same hand pose). At recognition time, an appropriate distance measure is used to find the representative image for the corresponding object class (the hand pose) at the leaves of this tree. The approach has been tried for hand pose recognition on a training set of images containing large variability in background, illumination and for some cases, the exact shape of the hand. Our algorithm were learned on a dataset consisting of five hand poses (three of them were used in [15]).

An overview of the whole approach is described in section 2. The process of building the tree classifier is illustrated in section 3 where sections 3.1, 3.2, and 3.3 describe the clustering method, weak classifier learning and the soft margin AdaBoost respectively. Section 4 explains the experimental setup and results obtained. Finally, section 5 discusses the findings and future works on this study.

1.1 Related Work

Researchers have recently grown strongly motivated to estimate hand gestures present in an image. There have been extensive research on recovering hand pose using a 3D hand model through model fitting. However, we focus our review of related work to learning-based approaches which treats the problem as a classification problem. Hand pose recognition task was formulated as a database indexing problem for a large database of articulated hand images in [1, 2]. In [3], volumetric representation of hands learned from their silhouettes were considered. Kölsch et.al. [9] and it's extension [8], investigated the performance of Viola-Jones detector [7] for recognizing hand gestures with a frequency spectrum based estimator to predict which gestures are more likely to yield a higher classification rate than that of others. Both [9, 8] used labeled hand pose images to learn the classifiers.

Due to the large variability of the gestures in real images, it may not always be possible to label the training images perfectly. Wu et.al. [17] uses a dataset of both unlabeled and labeled data for view independent gesture estimation. A template (generated by both edge and color features) matching technique has also been employed for hand pose detection in [15] that uses a dataset containing large variation in background and illumination. A linear classifier learned with oriented edges and marginalized templates has been shown to produce very good results for three hand poses. A detection tree classifier were used to prune out a large portion of the image, which is less probable to contain a hand, from the search. Since hand appearances in the real world images can vary largely in shape and color for the same hand pose class, a template based approach is not sufficient for pose estimation. In [11], a two-layer boosted classifier tree has been used to detect the hand in the image and recognize its pose. The top layer of the tree detects hand in an image window and the cascades of classifiers below recognize the hand pose. High success rates have been reported by the authors on hand poses with the same skin color of hands, background, and illumination which seem to underestimate the actual scenario.

Our work, however, does not deal with finding out a hand in any given query image. We are interested in discovering the pose provided an image of only the hand itself. We used an *extended* dataset of that was used in [15] for its large variability in background, illumination, and the exact shape of the hand for the same hand pose because we are interested in this study to recognize pose in realistic cluttered environment under varying conditions.

2 Overview of the approach

Let us start by considering a set of images of different hand poses, $\mathscr{I} = \{\mathbf{x}_k \in \Re^L | \, k = 1, 2, \cdots, n\}$ If we recursively divide this set $\mathscr{I}$ into two disjoint subsets $\mathscr{I}_A$ and $\mathscr{I}_B$ by a top-down hierarchal

clustering algorithm, we can build a tree where leaf contains a set of images which is anticipated to contain images from only one object class. This tree can be employed as a tree classifier for classifying multiple objects if we apply a binary classifier at each node containing image set $\mathscr{I}$ of this tree to realize the separating hyperplane between two subsets $\mathscr{I}_A$ and $\mathscr{I}_B$ of images. In the following paragraphs we will describe the clustering process and the training of the tree classifier in this work.

Suppose we have N images $\mathbf{x}_k,\ k = 1,2,\cdots,N$ for training and they are stacked one on top of the other in the matrix $X \in \Re^{N\times L}$. We first apply a suitable feature ϕ on the images to transform the data matrix from input space to the feature space; i.e., from $X \in \Re^{N\times L}$ to $X_\phi \in \Re^{N\times L_\phi}$. Since the dimensionality L_ϕ of the feature space is usually too large for the clustering algorithm to generate meaningful clusters, we need to further apply a dimensionality reduction tool. Principal Component Analysis (PCA) is used to reduce the dimensionality of the transformed data matrix. Let us denote the data matrix in the space spanned by the principal components of X_ϕ as $Z_{PCA} \in \Re^{N\times L_{PCA}}$.

It seems more intuitive to use a partitioning algorithm that makes no assumption about the number of clusters and utilizes the association between the subsets of samples for partitioning. The Normalized Cut [14] method (more generally any spectral clustering algorithm) is one such algorithm which apparently is more suitable for the problem of partitioning an image set. Unlike other clustering algorithms, e.g. K-Means, EM etc., NCut calculates a vector that contains useful information about the correspondence between the potential clusters and the images in the training set.

We, therefore, execute Normalized Cut (NCut) method to cluster the data $Z_{PCA} = [\mathbf{z}_1^T\ \mathbf{z}_2^T\ \cdots\ \mathbf{z}_N^T]^T$ in the PCA space. Given the distance matrix $W \in \Re^{N\times N}$ of the PCA space representation of images $\mathbf{z}_k,\ k = 1,2,\cdots N$, the Normalized Cut algorithm calculates a vector $\mathbf{b} \in \{-1,1\}^N$ (after thresholding) such that $\mathscr{I}_A = \{\mathbf{x}_{k_+} | b_{k_+} = +1\}$ and $\mathscr{I}_B = \{\mathbf{x}_{k_-} | b_{k_-} = -1\}$ defines two subsets of images that have the least association between each other (in the PCA space). The clustering algorithm is repeatedly applied on the new nodes in the same manner to generate the tree structure to be used by the classifier.

In the tree classifier, we will train a binary classifier at each node so that it will be able to classify any query image $\mathbf{x}^q$ as a member of either of the two subsets of images residing at the two children of the node. Then the query image $\mathbf{x}^q$ will again be discriminated by a another classifier at the next level of the tree and this process will continue until $\mathbf{x}^q$ reaches a leaf. Since the clustering algorithm may not be able to separate the groups of images of same pose perfecty, there may be some impurities in either or both $\mathscr{I}_A$ and $\mathscr{I}_B$. Therefore, we chose a variant of theAdaBoost [4] algorithm, namely the Soft margin AdaBoost [12] which maximizes the lower bound of the margin of the strong classifier in presence of noise, to be employed at each node. Our choice of the learning algorithm will be justified both theoretically and practically in latter sections. The classifier was trained using the feature responses $\tilde{\psi}_k = \psi(\mathbf{x}_k)$ of a different set of features ψ. We have used a novel method for learning of the weak hypotheses from these responses $\tilde{\psi}_k$, using the clustering information produced by the NCut method and the probability distribution imposed on the training examples at each round of the boosting process.

When the query image $\mathbf{x}^q$ reaches the leaf, we need a way to find out which of the object classes it belongs to. Since we are not using any labels for the learning process, we determine the image closest to $\mathbf{x}^q$ within the image set at the leaf according to some similarity measure $d()$. The class of the closest image is then declared as the class of the query image $\mathbf{x}^q$.

3 Unsupervised Learning of Tree Classifier

3.1 Normalized Cut Clustering

Consider each point $\mathbf{z}_k,\ k = 1,2,\cdots,N$ in the PCA space as nodes of some graph $G = (V,E)$ where there is an edge from each node to all other nodes in V. Let W be the weighted adjacency

matrix for the edges in E. We calculated the distance between any two points $\mathbf{z}_{k_1}$ and $\mathbf{z}_{k_2}$, or equivalently, the weight for the edge between $\mathbf{z}_{k_1}$ and $\mathbf{z}_{k_2}$, using multivariate gaussian distance in the PCA space.

$$[W]_{k_1,k_2} \;=\; w(k_1,k_2) \;=\; \exp\big[-\frac{1}{2}(\mathbf{z}_{k_1}-\mathbf{z}_{k_2})^T\Sigma^{-1}(\mathbf{z}_{k_1}-\mathbf{z}_{k_2})\big]. \tag{1}$$

The bandwidth (or covariance) matrix Σ for the gaussian measure was determined by the local scaling approach proposed in [18].

Given the weight matrix W, the Normalized Cut algorithm [14] will generate the optimum partitioning vector $\mathbf{r}$ to divide V ($\mathscr{I}$ in our case) into two disjoint subsets A and B ($\mathscr{I}_A$ and $\mathscr{I}_B$ respectively). The partitioning vector $\mathbf{r}$ is almost always real valued whereas we need a binary indicator vector $\mathbf{b}$ that assigns $\mathbf{z}_k$ to the the appropriate cluster. We follow the approach described in [14] for searching the entry δ in $\mathbf{r}$ that produces the minimum association between two clusters, i.e. the minimum NCut value.

Returning to the notations we have been using so far, the two subsets $\mathscr{I}_A$ and $\mathscr{I}_B$ of $\mathscr{I}$ are obtained by checking the values of the eigenvector entries against the threshold δ.

$$\mathscr{I}_A = \{\mathbf{x}_{k_+}|b_{k_+}=1\} \quad \text{and} \quad \mathscr{I}_B = \{\mathbf{x}_{k_-}|b_{k_-}=-1\} \tag{2}$$

where $b_k = sign(r_k-\delta)$ and the symbols b_k and r_k denotes the entries in the vectors $\mathbf{b}$ and $\mathbf{r}$ respectively for $k=1,2,\cdots,N$. The resulting subsets are recursively partitioned into smaller sets in the same way until the algorithm satisfies some terminating condition. The quantitative measure of association between two clusters, namely the NCut value, and the cluster size are used as a stopping criterion. One of the reasons to favor Normalized Cut method over other clustering methods is the fact that it makes no assumptions about the number of clusters and the distribution of points pertaining to it. The only external parameter we need to set,namely the NC_{th} value is intuitive and less strict for an assumption. Furthermore, as we will see in the next section, the partitioning vector $\mathbf{r}$ can be used to learn the binary classifier at each node of the classifier tree.

3.2 One-pass Learning of weak classifiers

We will learn a binary boosting classifier [4] at each node of the tree generated by the NCut method. We propose a fast and elegant method for learning the weak classifiers at each stage of boosting by utilizing the information provided by the clustering algorithm. Using our notation, $\mathbf{r}$ (the second eigenvector of the standard eigensystem in section 3.1) stores the partitioning information for subsets $\mathscr{I}_A$ and $\mathscr{I}_B$. As we know [10], in ideal cases, $\mathbf{r}$ is a piecewise linear vector characterizing the correspondence between the points to the clusters to be discovered. Therefore, when classifying between the two subsets produced by the NCut algorithm, the vector $\mathbf{r}$ can be considered to contain the 'soft labels'(we will be using this term for $\mathbf{r}$ in rest of the paper) for the classifying task. The binary vector $\mathbf{b}$ (as defined in equation 2) generated by thresholding $\mathbf{r}$ against δ works as the actual labels for classification.

For the purpose of learning the binary classifiers, a different set of features $\psi^j, j=1,2,\cdots,L_\psi$ are applied on the images. Let us denote the the N dimensional vector for j-th feature responses of all the images $\mathbf{x}_k,\ k=1,2,\cdots,N$ as $\tilde{\psi}^j=[\tilde{\psi}^j_1,\tilde{\psi}^j_2,\cdots,\tilde{\psi}^j_N]^T$ where $1\le j\le L_\psi$. At each round t of boosting process, we need to find a weak classifier making the least classification error in terms of the labels $\mathbf{b}$ weighted by the probability distribution $\pi^t_k,\ k=1,2,\cdots,N$ imposed on the training samples at t-th round. We learn a binary weak classifier $h^j\in\{+1,-1\}^N$ for each feature ψ^j. The weak hypothesis h^j assigns an image $\mathbf{x}_k$ to subset $\mathscr{I}_A$ (class +1) if the corresponding feature response $\tilde{\psi}^j_k$ is greater than some threshold θ^j and assigns $\mathbf{x}_k$ to subset $\mathscr{I}_B$ (class -1) otherwise.

$$h^j(\mathbf{x}_k)=\begin{cases}+1 & \text{if } \tilde{\psi}^j_k>\theta^j\\ -1 & \text{if } \tilde{\psi}^j_k\le\theta^j\end{cases} \tag{3}$$

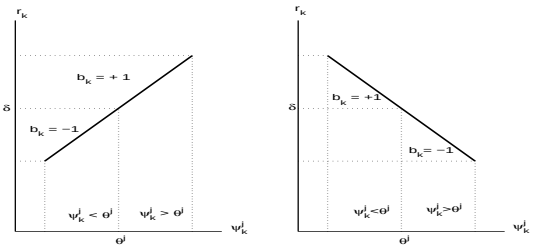

Figure 1: Linear relationship between feature response and soft labels

The most discriminative feature among all the $\psi^j, j = 1,2,\cdots,L_\psi$ will have distinct densities of responses for samples of two classes and will have a high correspondence between its entries and that of the label $\mathbf{b}$. Since both soft labels $\mathbf{r}$ and feature responses $\tilde{\psi}^j$ are real numbers, we expect the relationship reflected be more strongly between them. We model this correspondence by a linear relationship between $\mathbf{r}$ and $\tilde{\psi}^j$ as follows:

$$\mathbf{r} = \alpha^j \tilde{\psi}^j + \beta^j \tag{4}$$

where α^j and β^j are two scalar coefficients. This assumption of linear dependence gives us a straightforward way to determine the threshold θ^j by $\theta^j = \frac{\delta - \beta^j}{\alpha^j}$.

The linear model between the soft labels and the feature responses thus eliminates the necessity for an exhaustive search (i.e. to check sufficiently large number of values in the range of the feature values) to compute θ^j. The direction of the inequality signs in the definition of h^j in equation 3 may be reversed with the sign of α^j as shown in figure 1. The soft labels $r_k, k = 1,2,\cdots,N$ of the N images are plotted (in the y-axis) against the features responses $\tilde{\psi}^j_k$ of particular feature ψ^j in figure 1. As we can see, if the slope of the linear relation is positive, the weak classifier assigns a class label +1 to $\mathbf{x}_k$ if the feature response $\tilde{\psi}^j_k$ is greater than the corresponding threshold θ^j. But if the slope is negative, $\mathbf{x}_k$ is assigned a label -1 when $\tilde{\psi}^j_k > \theta^j$. Finally, the weak hypothesis h^{j_t} to be selected at round t is the one making the least weighted error against the actual labels $\mathbf{b}$.

$$h^{j_t} = \arg\min_j \ \varepsilon^j_t \quad \text{where} \quad \varepsilon^j_t = \sum_{k=1}^{N} \pi^t_k \, I[b_k \neq h^j(\mathbf{x}_k)]. \tag{5}$$

Here I is the indicator function that outputs 1 if the expression within the square braces is true and outputs 0 otherwise.

We can use Least Square method to compute the linear coefficients α^j and β^j in equation 4. Let us define matrix $\tilde{\Psi}^j = [\tilde{\psi}^j \ \mathbf{1}]$ and vector $\mathbf{a}^j = [\alpha^j \ \beta^j]^T$ to write the linear dependence in equation 4 in the matrix form as follows:

$$\mathbf{r} = \tilde{\Psi}^j \, \mathbf{a}^j. \tag{6}$$

Recall that at each round t of boosting, we emphasize on learning a different subset of the training examples according to the probability measure π^t_k. Therefore, we need to use a weighted least square approximation to calculate the coefficients of the features that are biased towards the samples by higher π^t_k than that of others. We need to find out the coefficients $\mathbf{a}^j$ minimizes the weighted squared error

$$\varepsilon_{rgs} = (\tilde{\Psi}^j \, \mathbf{a}^j - \mathbf{r})^T \Pi^t \, (\tilde{\Psi}^j \, \mathbf{a}^j - \mathbf{r}) \tag{7}$$

where $\Pi^t = diag(\pi^t_1, \pi^t_2, \ \ldots \ , \pi^t_N)$. The coefficients $\mathbf{a}^j$ that minimizes ε_{rgs} can be derived as (see [6]for reference)

$$\mathbf{a}^j = ((\tilde{\Psi}^j)^T \Pi^t \, \tilde{\Psi}^j)^{-1} (\tilde{\Psi}^j)^T \Pi^t \, \mathbf{r}. \tag{8}$$

3.3 Soft Margin AdaBoost

The weak hypotheses learned from the feature responses are combined to form a strong classifier. A large margin classifier should be preferred to ensure the robustness of the tree classifier. The authors of [13] have demonstrated that, the margin of the final classifier increases with the number of weak classifiers added to the final classifier of a boosting process and results in a continuous reduction in generalization error. There are two issues worth discussing at this point. First, if at any stage of the regular AdaBoost [4], the weighted error made by the weak classifier becomes zero, we need to abort the iterative process of adding new classifiers. The second concern is, in noisy environment, adding more weak hypothesis will concentrate on learning the noisy samples rather than increasing the margin [12].

Rätsch et.al. [12], proposed a modification to the objective functional that AdaBoost-type algorithms minmize. Let us denote $H(\mathbf{x}_k) = \sum_{t=1}^{T} c_t h_t(\mathbf{x}_k)$ as the linear combination of T weak hypotheses $h_t(\mathbf{x}_k) \in \{-1,+1\},\ t = 1,2,\cdots,T$ according to the (non-normalized) coefficients $\mathbf{c} = [c_1, c_2, \cdots, c_T]^T$. If we define the margin for each sample $\mathbf{x}_k$ as $\rho_k(\mathbf{c}) = b_k \sum_{t=1}^{T} c_t h_t(\mathbf{x}_k)$, where b_k is the label for $\mathbf{x}_k$, it was argued that AdaBoost minimizes the following expression at each step of boosting:

$$G(\mathbf{c}) = \sum_{k=1}^{N} \exp\{-\frac{1}{2}\rho_k(\mathbf{c})\}. \tag{9}$$

It was assumed that each hypothesis makes a weighted error less than $\frac{1}{2}$. The authors of [12] introduce a *soft margin* $\tilde{\rho}_k(\mathbf{c})$ for each sample $\mathbf{x}_k$ as the summation of $\rho_k(\mathbf{c})$ and a term quantifying the how much we 'mistrust' $\mathbf{x}_k$ to be an accurately labeled example.

$$\tilde{\rho}_k(\mathbf{c}) = \rho_k(\mathbf{c}) + B(\mu_k)^p \ \text{ where } \ \mu_k = \sum_{r=1}^{t} c_r \pi_k^r \tag{10}$$

The quantity μ_k, which is a weighted summation of the importance values (or weights) π_k^t of $\mathbf{x}_k$ at boosting round t, reduces the attention we are giving to any individual sample for maximizing its margin. The two external parameters B and p are used to further tune the contribution of μ_k to $\tilde{\rho}_k(\mathbf{c})$. The intuitive explanation for μ_k is, usually the outliers are largely different from the training samples that actually belong to the object class. Consequently, they are difficult to learn with the hypotheses trained for this class. Therefore, their π_k^t remain generally larger than that of the true samples of the class.

Replacing $\rho_k(\mathbf{c})$ by $\tilde{\rho}_k(\mathbf{c})$ in the equation, the iterative update formulae for $\mathbf{c}$ and π_k^t become:

$$\begin{aligned} \mathbf{c}^{(t)} &= [\mathbf{c}^{(t-1)}\ c^t]^T \ \text{ with } \ c^t = \arg\min_{c^t \geq 0} \sum_{k=1}^{N} \exp\Big\{-\frac{1}{2}\ [\ \rho_k(\mathbf{c}^{(t)}) \ + \ B\ |\mathbf{c}^{(t)}|\mu_k^t\ ]\Big\}; \\ \pi_k^t &= \frac{1}{Z_t}\ \exp\Big\{-\frac{1}{2}[\ \rho_k(\mathbf{c}^{(t)}) \ + \ B\ |\mathbf{c}^{(t)}|\mu_k^t\ ]\Big\}. \end{aligned} \tag{11}$$

The superscript t for each quantity denotes the value of that specific variable at boosting stage t. The denominator Z_t in the definition of π_k^t is a normalizing constant to convert $\pi_k^t, k = 1,2,\cdots,N$ into a probability distribution. As can be observed, these update rules allow to add as many weak hypothesis as we want even if the corresponding weighted error vanishes. The final form of the strong classifier is a sign function of the value of H with normalized linear coefficients : $f(\mathbf{x}_k) = sign\left[\frac{H(\mathbf{x}_k)}{\sum_{t=1}^{T} c^t}\right]$.

4 Experiment and Results

For our experiment, we used the datasets of [15] with two more types of hand pose images. These images were scaled to the size 64×64 pixels and converted to grayscale. The image set contains 5 hand poses, namely hand open to side, pointing to side, fist to the side, pointing towards the

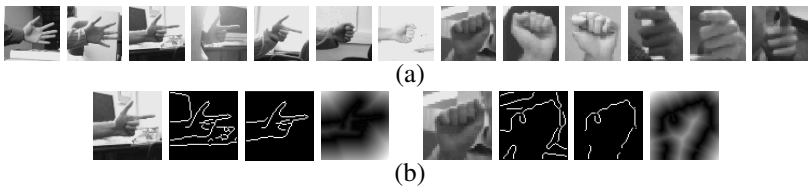

Figure 2: Images from training set and their feature transforms. (a) Sample images from training set. (b) Feature transform for clustering. From left to right, the training image, original edge image, cleared edge image, distance transform w.r.t. the cleared edge image

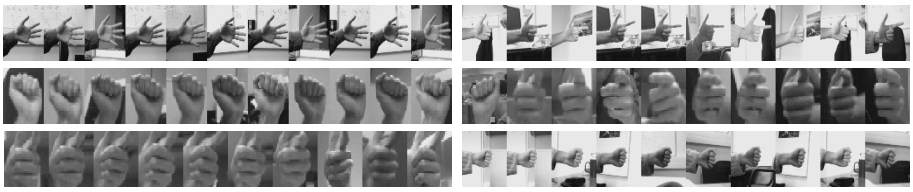

Figure 3: Sample clusters generated by the NCut algorithm

camera, and fist upwards. The background, illumination, exact hand shape and clothing of the images vary largely in the data, as can be seen from some example images in figure 2 (a). We have randomly picked 343 images for training and 375 images (75 per pose) for testing purposes. The sizes of image sets for different hand poses were not the same in the training set, but each set contains at least 50 images. Each pose of the test set contains 75 images. The task is to find out the hand pose of each of these 375 test images when presented to the learning algorithm.

Feature used for Clustering: For clustering purposes, we have used the Distance transform (DT) [5] images as the feature ϕ (section 2). To eliminate the interference of the background, the edge images that were used to calculate the DT of an image were manually cleaned to remove the edges in the background. Some of the images $\mathbf{x}_k, k = 61, 156$ and their feature transforms $\phi(\mathbf{x}_k)$ have been shown in figure 2 (b).

Building decision tree by clustering: The PCA transform of the DT images in X_ϕ generates the $L_{PCA} < L_\phi$ dimensional data matrix Z_{PCA} retaining 99% of the variance. To apply NCut clustering to Z_{PCA}, we need to calculate the distance matrix W. We consider distances of η nearest neighbors of $\mathbf{z}_k$ (according to the Euclidean distance in the PCA space) while computing the bandwidth matrix Σ used to calculate W in equation 1 (see [18] for details). As we can infer, the height of the tree can be minimized if the clustering algorithm divides the image set $\mathscr{I}$ into almost equal sized subsets $\mathscr{I}_A$ and $\mathscr{I}_B$. Therefore, we examined multiple clustering results for the weight matrix W computed in l_{PCA} (where $4 \leq l_{PCA} \leq L_{PCA}$) dimensional subspaces of the PCA space and with $\eta \in \{5, 10, 20\}$. We calculated the ratio $\frac{|\mathscr{I}_A|}{|\mathscr{I}_B|}$ of sizes of $\mathscr{I}_A$ and $\mathscr{I}_B$ discovered by NCut algorithm for all these l_{PCA} and η values. The combination of the values of l_{PCA} and η that produced the largest ratio $\frac{|\mathscr{I}_A|}{|\mathscr{I}_B|}$ was used for final clustering. The terminating NCut value was $NC_{th} = 0.375$ and the minimum size of cluster to be partitioned further was taken as 10. Some of the subsets at the leaves of the tree generated by recursively calling NCut algorithm are shown in figure 3.

We have found in the experiment that, NCut is actually separating the training set very well at the internal nodes. For example, at the root, the two subsets created by the clustering contain the poses {open side, fist side, point side} (the first three poses in figure 2 (a) from left) and {point toward camera, fist up} respectively with two noise images. The former subset is further divided into sets containing hand poses {open side} with one noise image and {point side, fist side} (the second and third pose from left in figure 2 (a)).

Features for classification: After generating the tree, we need to apply a binary classifier at each node. We can not use DT features for classification since the edges from background clutter (which were not present in training images) will distract the classifier. Therefore, the features

we used for classification are oriented gaussians and their first and second partial derivatives at two bandwidths, namely $\{8,5\}$ pixels. They were generated for eight orientations ranging from 0 to 2π with an interval of $\frac{\pi}{4}$. For the elliptical gaussian filters, the bandwidths in y-direction are half of that used in x-direction. The soft margin AdaBoost [12] classifier is then learned at each internal node of the tree. The external parameters in equation 11 were set to $B = 100$ and $p = 2$.

Pose recognition at leaf: When a query image $\mathbf{x}^q$ reaches the leaf, we searched over the subset $\{\mathbf{x}_k\}_{k=1}^{N_{leaf}}$ residing at the leaf based on a variant of Chamfer distance to find out the image $\mathbf{x}_{closest}$ closest to $\mathbf{x}^q$. The hand pose of $\mathbf{x}_{closest}$ is assigned as the class of $\mathbf{x}^q$. The cleaned edge images (as shown in figure 2) of these N_{leaf} images are utilized to calculate the distance. The DT image of $\mathbf{x}^q$, calculated with respect to its own edge features, is compared to the clean edge images of the leaf. We then build N_{leaf} histograms, each comprising 10 bins, of the pixels $x^q(i), i = 1,2,\cdots,L$ according to the the distance $x^q(i)$ has to its closest edge feature in the cleaned edge image of $\mathbf{x}_k$ at the leaf. Out of these N_{leaf} histograms, we consider only frequencies of pixels in the first bin of the histogram. The leaf image having the largest number of pixels in the first bin is considered to be the best match for $\mathbf{x}^q$. This measure is more strict than Chamfer distance since it only considers the number of pixels which are nearest to the edge features (in the cleaned image) instead of taking into account the distance of all the pixels in $\mathbf{x}^q$.

Table 1: Recognition accuracies of the decision tree with different classifiers

Pose	Fist side	Fist up	Point forward	Point side	Open side	Classifier
Fist side	0.96	0.01	0	0	0.03	softAB_w
	0.93	0	0	0	0.67	softAB_exstv
	0.97	0	0	0	0.03	AB_w
Fist up	0.02	0.78	0.15	0.025	0.025	softAB_w
	0	**0.8267**	0.133	0.013	0.027	softAB_exstv
	0.026	0.787	0.133	0.027	0.027	AB_w
Point forward	0	0.06	**0.94**	0	0	softAB_w
	0	0.12	0.88	0	0	softAB_exstv
	0	0.11	0.89	0	0	AB_w
Point side	0.06	0	0	**0.94**	0	softAB_w
	0.08	0	0	0.89	0.03	softAB_exstv
	0.11	0	0	0.88	0.01	AB_w
Open side	0	0	0.03	0.04	**0.93**	softAB_w
	0	0.027	0.03	0.053	0.89	softAB_exstv
	0	0	0.05	0.06	0.89	AB_w

Recognition results: The results of hand pose recognition by soft margin AdaBoost classifier using the weak classifier learning (softAB_w) as described in section 3.2 are summarized in table 1 in the form of a confusion matrix. More precisely, the decision tree with softAB_w classifier correctly classifies 94% of the images of 'Point forward'(refer to the third row) but misclassifies 6% of them as 'Fist up'. Notice that, the sidewise poses { open side, fist side, point side} are generally confused among themselves. For the sake of comparison, we also generated the results using regular AdaBoost (AB_w) using the weak classifier learning and using soft margin AdaBoost *without* using the weak classifier learning (softAB_exstv). To get around with the scenario where the weighted error for the first weak hypothesis is zero in regular AdaBoost(i.e. ε_1^j at the first stage of boosting), we applied a majority vote classifier by aggregating the weak hypotheses having zero error on the data subset at hand. As we observe, softAB_w has the best performance among all the classifiers used in the study. The inferior classification result for the pose 'fist up' is due to the complexity in the data subset itself, which is why all the learning algorithm worked poorly for recognizing images from this set.

Analyzing margins for generalization: Since the clustering algorithm is performing very well on the training set, we do not expect the classifiers to experience highly noisy data to learn on. This is the reason why AB_w is also performing fairly well in this case. Nonetheless, the softAB_w algorithm produces significantly better results than that produced by AB_w. The over-

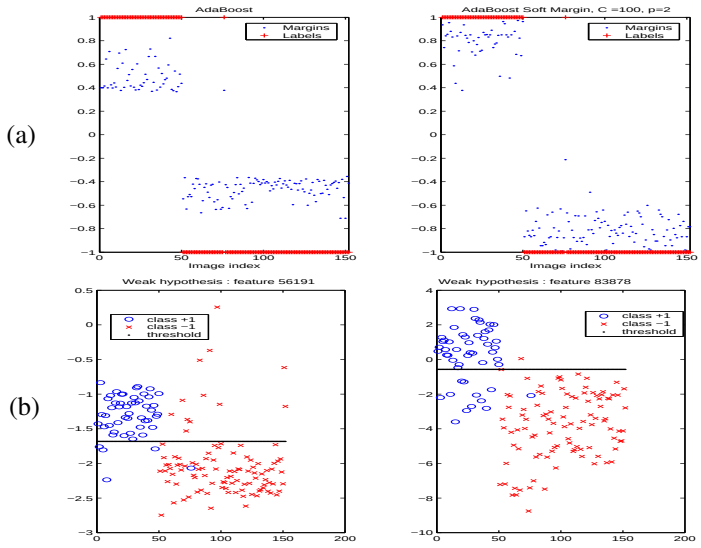

Figure 4: Comparison of margin and base hypothesis. (a) Comparison between classifier margins of AB_w and softAB_w. (b) Comparison between two approaches to learn the weak classifier in the first boosting step at node 2. Left, by linear regression and right, by exhaustive search

fitting of AB_w leads to an inferior generalization to the test set images. To justify the claim, let us consider the 4th row in table 1 showing the results for the pose 'point side'. We observe that more images from this class were confused by AB_w as pertaining to the pose 'fist side' than that were confused by softAB_w. Let us examine the node where the algorithm driving the query image to a wrong direction, namely node 2 of the tree.

In figure 4(a),we plot the classifier margins (label multiplied by the linear combination of the weak classifier outputs) against the class labels. We can easily sort out the noise image sitting at the middle of the x-axis of the plot with an image index of 76. The regular Adaboost assigns it a +1 label whereas the soft margin version of it assigns a -1. But observe that the regular Adaboost margin for this noise image is approximately 0.4 which is also the margin of many other images of class +1. The reason behind this is, regular Adaboost adds many weak hypothesis just to learn this particular sample after a certain number of boosting steps (when the probability measure π^t_{76} grows too high for this sample). Therefore, the importance of the other images are underestimated and eventually results in a low margin output. The soft margin version, on the other hand, does not emphasize on learning any particular example (refer to the definition of the soft margin in equation 10) and concentrates on maximizing the overall margin of all examples in the training set. This characteristic is exhibited in figure 4(a), where the average absolute value of the softAB_w margin is approximately 0.8 which is much larger than that of AB_w which is approximately 0.5.

Comparison of weak classifier learning approaches: For the exhaustive search approach to learn the weak classifiers, we checked $Q \in \{10, 20, \cdots, 90\}$ quantile values of the feature responses of each feature to find out the threshold. Figure 4(b) compares weak hypotheses learned this way with that learned using the proposed approach in section 3.2 at node 2 (left child of root) of the decision tree in the first boosting round. Both of the approaches are selecting almost perfect weak classifiers, but the exhaustive search increases the time required for training by an order of 10. Increasing the number of values to check for a threshold did not produce significantly better results and slows the learning process even further. The reason why softAB_exstv producing inferior recognition results is, since we are not relying on any model to determine the optimum threshold for the weak classifiers, it is not generalizing well to the test data.

The results from this work is not comparable to that of [15] because we recognize the pose when presented with an hand image rather than detecting hand and poses from an image. Furthermore, the authors in [15] have used only three types of hand poses, namely open side, point

side, and fist side, for detection. Examining the whole dataset, we have found that images form these classes of hand poses have much stronger edge responses and the exact same shape of the hand throughout the subset of the images corresponding to the hand pose. These are all highly favorable characteristics for any classification method to be used for pose estimation. As a result, the best classifier in [15] was reported to have 99% accuracy with 5% false detection rate. Our dataset contains images of two more types of hand poses which are much more complicated than these images with regard to the edge responses and shape of the hand within each class. Nonetheless, our algorithm is producing very good results on these images.

5 Conclusion

This study proposes a boosted classification tree learned in an unsupervised fashion for selection and sharing of features in multi-class object recognition. The results of the implementation of the targeted dataset has shown very good results for hand pose estimation from images with random background, variation in illumination, clothing and the exact shape of the hand within each pose. Since we are not exploiting any characteristics of the hand itself for classification, this decision tree can be easily extended for any type of object recognition.

Acknowledgement: We would like to thank Dr. B. Stenger for providing us with the hand dataset used in this study.

References

[1] V. Athitsos and S. Sclaroff. An appearance based framework for 3d hand shape classification and camera viewpoint estimation. In *Proceedings of Fifth AFGR*, page 45, 2002.

[2] V. Athitsos and S. Sclaroff. Estimaing 3d hand pose from a cluttered image. In *Proceedings of Fifth CVPR*, 2003.

[3] S. Y. Cheng and M. M. Trivedi. Hand pose estimation using expectation-constrained-maximization from voxel data. In *Technical Report*. Computer Vision and Robotics Research (CVRR) Lab, UC San Diego, 2004.

[4] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[5] D. M. Gavrila. Pedestrian detection from a moving vehicle. In *Proceedings of Sixth ECCV*, volume II, pages 37–49, 2002.

[6] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer, 1st edition, 2001.

[7] M. Jones and P. Viola. Fast multi-view face detection. In *Technical Report TR 2003-96*. MERL, 2003.

[8] M. Kölsch and M. Turk. Analysis of rotational robustness of hand detection with a viola-jones detector. In *Proceedings of Seventeenth ICPR*, 2004.

[9] M. Kölsch and M. Turk. Robust hand detection. In *Proceedings of AFGR*, 2004.

[10] Marina Meila and Liang Xu. Multiway cuts and spectral clustering. In *Technical Report*. Univ. of Washington, 2003.

[11] E. Ong and R. Bowden. A boosted classifier tree for hand shape detection. In *Proceedings of Sixth AFGR*, 2004.

[12] G. Rätsch, T Onoda, and K. R. Müller. Soft margins for adaboost. *Machine Learning*, 42:287–320, 2001.

[13] Robert E. Schapire, Yoav Freund, Peter barlett, and Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics*, 26(5), 1998.

[14] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[15] B. Stenger, A. Thayananthan, P. Torr, and R. Cipolla. Hand pose estimation using hierarchical detection. In *International Workshop on Human-Computer Interaction, Lecture Notes in Computer Science*, pages 102–112. Springer, 2004.

[16] Antonio B. Torralba, Kevin P. Murphy, and William T. Freeman. Sharing visual features for multiclass and multiview object detection. In *CVPR*, 2004.

[17] Y. Wu and T. S. Huang. View-independent recognition of hand postures. In *Proceedings of CVPR*, volume 2, pages 84–94, 2000.

[18] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1601–1608. MIT Press, Cambridge, MA, 2005.