# Probabilistic Boosting-Tree: Learning Discriminative Models for Classification, Recognition, and Clustering

Zhuowen Tu
Integrated Data Systems Department
Siemens Corporate Research, Princeton, NJ, 08540

## Abstract

*In this paper, a new learning framework–**probabilistic boosting-tree** (PBT), is proposed for learning two-class and multi-class discriminative models. In the learning stage, the* probabilistic boosting-tree *automatically constructs a tree in which each node combines a number of weak classifiers (evidence, knowledge) into a strong classifier (a conditional posterior probability). It approaches the target posterior distribution by data augmentation (tree expansion) through a divide-and-conquer strategy. In the testing stage, the conditional probability is computed at each tree node based on the learned classifier, which guides the probability propagation in its sub-trees. The top node of the tree therefore outputs the overall posterior probability by integrating the probabilities gathered from its sub-trees. Also, clustering is naturally embedded in the learning phase and each sub-tree represents a cluster of certain level. The proposed framework is very general and it has interesting connections to a number of existing methods such as the $A^*$ algorithm, decision tree algorithms, generative models, and cascade approaches. In this paper, we show the applications of PBT for classification, detection, object recognition. We have also applied the framework in segmentation.*

## 1. Introduction

The task of classifying/recognizing, detecting, and clustering general objects in natural scenes is extremely challenging. The difficulty is due to many reasons: large intra-class variation and inter-class similarity, articulation and motion, different lighting conditions, orientations/viewing directions, and the complex configurations of different objects. The first row of Fig. (1) displays some face images. The second row shows some typical images from the Caltech-101 categories of objects [5]. Some of them are highly non-rigid and some of the objects in the same category bear little similarity among each other. For the categorization task, it requires very high level knowledge to put different instances of a class into the same category.

The problem of general scene understanding can be viewed in two aspects: *Modeling* and *Computing*. *Modeling* addresses the problem of how to learn/define the statistics of general patterns/objects of variation. *Computing* tackles the inference problem. Let $x$ be an image sample and its interpretation be $y$. Ideally, one wants to obtain the generative models $p(x|y)$ for a pattern to measure the statistics about any sample $x$. Unfortunately, not only are such generative models often out of reach, but also they create big computational burdens in the computing stage. For example, faces are considered a relatively easy class to study. Yet, there is no existing generative model which captures all the variations for a face such as multi-view, shadow, expression, occlusion, and hair style. Some sample faces can be seen in Fig. (1). Alternatively, one seeks to directly learn the discriminative model $p(y|x)$ in which $y$ is just a simple variable, to say, “yes” or “no”, or a class label.

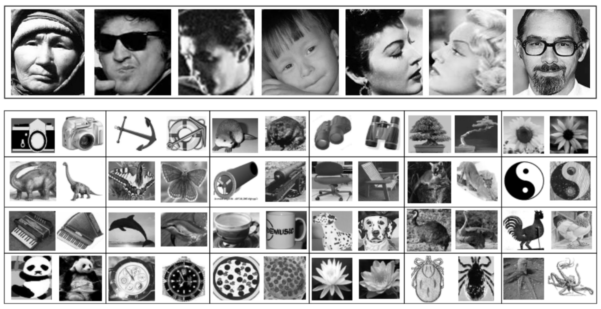


Figure 1: Faces cropped from natural images and some typical objects cropped from the Caltech-101 categories by Fei-fei et. al. [5].

AdaBoost, invented by Freund and Schapire [8], and its variants [9] have been successfully applied in many problems in vision and machine learning. They have been shown [9] to approach the posterior $p(y|x)$ by selecting and combining a set of weak classifiers into a strong classifier. However, there are several problems with the current AdaBoost algorithm. *First*, though it asymptotically converges to the target distribution, it may need to pick hundreds of weak classifiers. This poses a big computational burden. *Second*, the order in which features are picked in the training stage is not preserved. The order of a set of features may correspond to high-level semantics and, thus, it is very important for the understanding of objects/patterns [1]. *Third*, the re-weighting scheme of AdaBoost may cause samples previously correctly classified to be miss-classified again. *Fourth*, though extensions from two-class to multi-class classification have been proposed [8, 18], learning weak classifiers in the multi-class case using output coding is more difficult and computationally expensive.

In this paper, a new learning framework called *probabilistic boosting-tree* (PBT) is introduced to tackle the above problems. In the training stage, a tree is recursively constructed in which each tree node is a strong classifier. The input training set is divided into two new sets, left and right ones, according to the learned classifier. Each of which is then used to train the left and right sub-trees recursively. We show that the discriminative model obtained at the top of the tree is approaching the target posterior distribution by data augmentation. Each level of the tree is an augmented variable. Also, clustering is intrinsically embedded in the learning stage with clusters automatically discovered and formed in a hierarchical way. The procedure of constructing such a tree is simple. It has connections to many existing methods such as the cascade approach, $A^*$ algorithm, decision tree, and generative models.

For the multi-class classification problem, the goal is to learn a discriminative model while keeping the nice hierarchical tree structure. This is done by treating the multi-class classification problem as a special two-class classification problem. At each node, either a positive or negative label is assigned to each class in minimizing the total entropy. Through this procedure, the multi-class and two-class learning procedures become unified. Clusters of multi-classes are again directly formed.

A relevant algorithm to PBT is *AdaTree* [11] proposed by Grossmann, which also combines the AdaBoost with decision tree. However, PBT and AdaTree are different in many aspects. First, the main goal of the AdaTree algorithm is to speed up the AdaBoost algorithm by pruning, whereas the major focus of PBT is to construct a general framework for learning the posterior distribution $p(y|x)$ for complex pattern x, which is of great importance to vision and machine learning. Second, the AdaTree algorithm learns a strong classifier by combining a set of weak classifiers into a tree structure. But PBT constructs a tree in which each node itself is a strong classifier. Third, the multi-class classification problem was not addressed in [11].

In this paper, we show two applications using PBT: (I) Categorization of multi-class objects of the ETH-80 image set [15], NORB jittered-cluttered images [13] , and the Caltech-101 categories [5]; (II) Multi-view face detection.

## 2. Generative vs. Discriminative

Let an image sample $x$ and its interpretation $W$ be

$$(x, W), \quad W = (y, \alpha, \Theta),$$

where $y$ is the label of the object, e.g., category id. $\Theta$ is the underling template dictionary from which $x$ is generated and $\alpha$ specifies all the variations such as transformations and lighting changes which govern the generation process of $x$. The likelihood term $p(x|\alpha, y; \Theta)$ decides the sample $x$. In classification, the label $y$ is often of interest. Thus,

$$p(y|x;\Theta) = \int_\alpha p(y|\alpha, x; \Theta)p(\alpha)d\alpha. \tag{1}$$

The integral is troublesome for both modeling and computing. Instead, one may seek to directly learn the relations between $x$ and $y$ as a generative model $\mathbf{p}(\mathbf{x}|\mathbf{y})$or a discriminative model $\mathbf{p}(\mathbf{y}|\mathbf{x})$ without explicitly specifying the hidden variables $\alpha$.

Next, we briefly discuss the discriminative model learning framework, AdaBoost, and the generative model learning framework, MinMax Entropy Principle.

**AdaBoost**

It is shown in [9] that the AdaBoost procedure is essentially approaching logistical regression. Further, AdaBoost and its variations are shown to approximate the true posterior distribution $p(y|x)$ by

$$q_\lambda(y|x) = \frac{1}{\sum_y exp\{\sum_{j=1}^T \lambda_j f_j(x,y)\}} exp\{\sum_{j=1}^T \lambda_j f_j(x,y)\}, \tag{2}$$

where $f_j(x, y)$ is an evaluation function. For example in binary AdaBoost, $f_j(x, y) = \frac{1}{2} y h_j(x)$, $y \in \{-1, +1\}$, and $h_j$ is a weak classifier. At each step, AdaBoost selects $h_j$ from a set of candidate weak classifiers and estimates $\lambda_j$ by minimizing the exponential loss.

**MiniMax Entropy Principle**

The MiniMax entropy principle [22] learns the model

$$q_\lambda(x|y) = \frac{1}{\sum_x exp\{-\sum_{j=1}^T \lambda_j h_j(x)\}} exp\{-\sum_{j=1}^T \lambda_j h_j(x)\}, \tag{3}$$

where $h_j(x)$ is a feature of $x$, which can also be a vector, say the histogram of a projection of $x$. Under this principle, it picks the $h_j$ that minimizes the entropy $Entropy(q_\lambda(x|y))$, for the $\lambda$ that achieves the maximum likelihood.

Both methods have a procedure to perform feature selection (greedy), and they both have direct relations to maximum likelihood when estimating the parameters. However, eq. (2) is in general much easier to calculate than eqn. (3) since its partition function (the normalization term) in on $y$ instead of $x$. Learning the discriminative model is more computationally tractable. For the rest of the paper, we focus on the learning of discriminative model $p(y|x)$.

## 3. Probabilistic Boosting-Tree

Sect. (2) shows that AdaBoost is approximating the posterior distribution, $p(y|x)$. Our goal is to learn this distribution for $x$ with high complexity. Also, the configuration of the features, like those in grammar, poses very important knowledge for understanding complex patterns. This shall be learned and preserved. Further, we are seeking a natural and direct way to extend the two-class learning scheme to the multi-class case.

### 3.1. Two-class Probabilistic Boosting-Tree

The general AdaBoost algorithm[8] and its variants learn a strong classifier by combining a set of weak classifiers $H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$ in which $h_t(x)$ is a weak classifier. For each sample $x_i$ with probability $w_i$, the error rate $\epsilon = \sum_i w_i [sign[H(x_i)] \neq y_i]$ is shown to be bounded by

$$\epsilon \leq 2^T \prod_{t=1}^{T} \sqrt{\epsilon_t (1 - \epsilon_t)}.$$

When dealing with $x_i$ with a complex distribution, $\epsilon_t$ quickly approaches $\frac{1}{2}$, and the convergence becomes slow. One possible remedy lies in designing more effective weak classifiers that are better separating the positives from the negatives. Unfortunately, it is often hard to obtain good weak classifiers and we are also constrained by the computational complexity in computing these classifiers and features. One of the key ideas in the AdaBoost is that samples incorrectly classified receive more weights the following time. Due to the update rule and normalization for $D_t$, previously correctly classified samples may be miss-classified again and thus receive penalty. Therefore, after some steps, weak classifiers become ineffective. Instead of putting all the weak classifiers together into a single strong classifier, we instead take a divide-and-conquer approach.

Fig. (2) gives the procedure for training a boosting-tree. Friedman et al. have shown in [9] that AdaBoost is approximating logistic regression. For notational simplicity, we denote the probabilities computed by each learned AdaBoost method as

$$q(+1|x) = \frac{exp\{2H(x)\}}{1 + exp\{2H(x)\}}, and \;\; q(-1|x) = \frac{exp\{-2H(x)\}}{1 + exp\{-2H(x)\}}.$$

The algorithm is intuitive. It recursively learns a tree. At each node, a strong classifier is learned using the standard boosting algorithm. The training samples are then divided into two new sets using the learned classifier, the left one and the right one, which are then used to train a left sub-tree and right sub-tree respectively. Variable $\varepsilon$ is used to control, to some degree, the overfitting problem. Those samples falling in the range of $[\frac{1}{2} - \varepsilon, \frac{1}{2} + \varepsilon]$ are confusing ones and will be used in both the left and the right sub-trees for training. If $\varepsilon = \frac{1}{2}$ then all the training samples are past into both the sub-trees with weights re-computed based on the strong classifier. PBT then become similar to Boosting. If $\varepsilon = 0$ then each sample is past either into right or left tree. Therefore, positive and negative samples are almost sure to be separated, if there are no identical ones. But it may overfit the data. In this paper, $\varepsilon$ is fixed at $0.1$ for all the experiments reported.

To train a tree of maximum depth of $L$:

- Given: A training set $S = \{(x_1, y_1, w_1), ..., (x_m, y_m, w_m)\}$; $x_i \in \chi$, $y_i \in \{-1, +1\}$, $\sum_i w_i = 1$.
- Compute the empirical distribution $\hat{q}(y) = \sum_i w_i \delta(y_i = y)$.
- On training set $S$, train a strong classifier using a boosting algorithm with T weak classifiers but exit early if $\epsilon_t > \theta$, e.g. $\theta = 0.45$.
- If the current tree depth is $L$ then exits.
- Initialize two empty sets $S_{left}$ and $S_{right}$.
- For each sample $(x_i, y_i)$, compute the probability $q(+1|x_i)$ and $q(-1|x_i)$ using the learned strong classifier.
- If $q(+1|x_i) - \frac{1}{2} > \varepsilon$ then $(x_i, y_i, 1) \rightarrow S_{right}$
  else If $q(-1|x_i) - \frac{1}{2} > \varepsilon$ then $(x_i, y_i, 1) \rightarrow S_{left}$
  else $(x_i, y_i, q(+1|x_i)) \rightarrow S_{right}$ and $(x_i, y_i, q(-1|x_i)) \rightarrow S_{left}$.
- Normalize all the weights of the samples in $S_{left}$.
- Repeat the procedure recursively.
- Normalize all the weights of the samples in $S_{right}$.
- Repeat the procedure recursively.

Figure 2: Training for a probabilistic boosting-tree. We set $\theta = 0.45$ and $\epsilon = 0.1$ for all the experiments in the paper. The strong classifier can be AdaBoost, RealBoost, or even algorithms other than Boosting.

If we split a training set into two parts, the new error rate

$$\epsilon_{split} = \sum_i w_i(l)[H_l(x_i) \neq y_i] + \sum_i w_i(r)[H_r(x_i) \neq y_i] \leq \epsilon, \quad (4)$$

where $\epsilon = \sum_i w_i [H(x_i) \neq y_i]$.

It is straight forward to see that the equality holds when $H_l = H$ and $H_r = H$. In general, reducing the number of input samples reduces the complexity of the problem leading to a better decision boundary.

Under this model, positive and negative samples are naturally divided into sub-groups. Fig. (3) shows an example of how a tree is learned and the training samples are divided. Samples which are hard to classify are passed further down leading to the expansion of the tree. Clustering of positives and negatives is naturally performed. One group serves as an auxiliary variable to the other group. Since each tree node is a strong classifier, it can deal with samples with complex distribution. Also, there is no need to pre-specify the number of clusters. The hierarchical structure of the tree allows us to report the clusters according to different levels of discrimination.

The testing stage is consistent with the training stage.

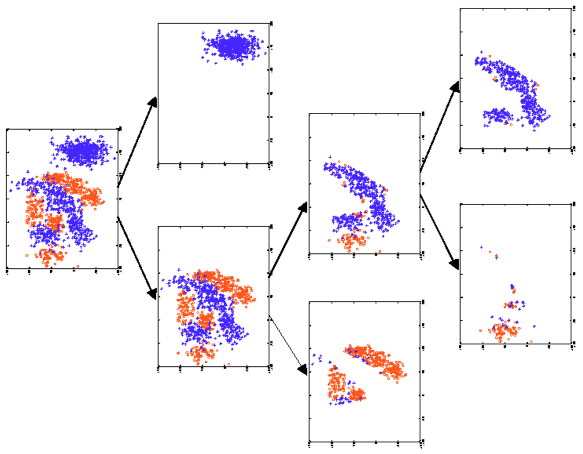


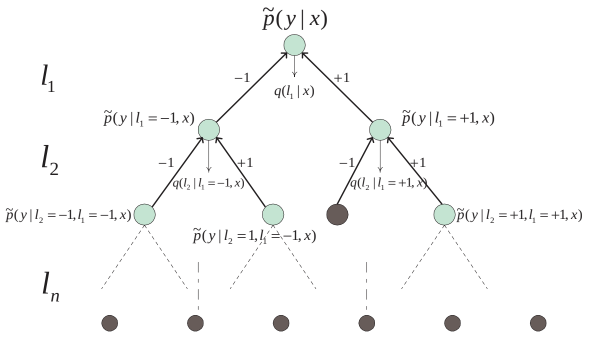
Figure 3: Illustration of PBT on a synthetic dataset of 2,000 points. Weak classifiers are likelihood classifiers on features such as position and distance to 2D lines. The first level of the tree divides the whole set into two parts. The right side mostly has blue (dark) points since they are away from the rest of the clouds. The tree expands on the parts where positive and negative samples are tangled.

```
Function $F_N(x, y)$: to compute posterior distribution $\tilde{p}_N(y|x)$ at tree node $N$.
  • For a given sample $x$, compute $q_N(+1|x)$ and $q_N(-1|x)$ using
    the learned boosting model at the current tree node $N$.
  • $\tilde{p}_N(y|x) = q(+1|x)\tilde{p}_{right}(y) + q(-1|x)\tilde{p}_{left}(y)$ in which
    $\tilde{p}_{right}(y)$ and $\tilde{p}_{left}(y)$ are computed below.
  • If $q(+1|x) - \frac{1}{2} > \varepsilon$ then
        $\tilde{p}_{right}(y) = F_{right(N)}(x, y)$, and
        $\tilde{p}_{left}(y) = \hat{q}_{left(N)}(y)$.
    else if $q(y + 1|x) - \frac{1}{2} > \varepsilon$ then
          $\tilde{p}_{right} = \hat{q}_{right}(y)$, and $\tilde{p}_{left} = F_{left(N)}(x, y)$.
      else
          $\tilde{p}_{right}(y) = F_{right(N)}(x, y)$, and
          $\tilde{p}_{left}(y) = F_{left(N)}(x, y)$.
```

Figure 4: Testing for probabilistic boosting-tree. The overall posterior is $\tilde{p}_1(y|x)$. $\hat{q}_{left(N)}(y)$ is the empirical distribution of the left tree.

Figure (4) gives the details of how to compute the approximated posterior $\tilde{p}(y|x)$. At the top of the tree node, it gathers the information from its descendants and reports an overall approximated posterior distribution. This algorithm can also be turned into a classifier which makes hard decision. After computing $q(+1|x)$ and $q(-1|x)$, one can decide to go into the right or left sub-trees by comparing $q(+1|x)$ and $q(-1|x)$. The empirical distribution $\hat{q}(y)$ contained at the leaf node of the tree is then passed back to the top node of the tree. However, the advantage of using probability is apparent. Once a PBT is trained, the $\tilde{p}(y|x)$ can be used as a threshold to balance between precision and recall. In contrast, a traditional cascade approach [21] needs to train different classifiers based on different precision requirements. Moreover, learning the discriminative models is of great importance to many vision problems, especially scene understanding. They can be used either directly in inference [12] or used as proposals to guide the generative model search [19].

**Understanding the tree**



$\tilde{p}(y\,|\,x)$
$l_1$
$l_2$
$l_n$
$-1$
$+1$
$q(l_1\,|\,x)$
$\tilde{p}(y\,|\,l_1=-1,x)$
$\tilde{p}(y\,|\,l_1=+1,x)$
$q(l_2\,|\,l_1=-1,x)$
$q(l_2\,|\,l_1=+1,x)$
$\tilde{p}(y\,|\,l_2=-1,l_1=-1,x)$
$\tilde{p}(y\,|\,l_2=1,l_1=-1,x)$
$\tilde{p}(y\,|\,l_2=+1,l_1=+1,x)$

Figure 5: Illustration of the probabilistic model of the tree. The dark nodes are the leaf nodes. Each level of the tree corresponds to an augmented variable. Each tree node is a strong classifier.

As shown in eqn. (1), a complex pattern $x$ is generated by a generation process which has a set of hidden variables. PBT can be viewed as having a similar aspect by doing implicit data augmentation. Fig. (5) gives an illustration. The goal of the learning algorithm is to learn the posterior distribution $p(y|x)$. Each tree level $l_i$ is an augmented variable.

$$\begin{aligned}
\tilde{p}(y|x) &= \sum_{l_1} \tilde{p}(y|l_1, x) q(l_1|x) \\
&= \sum_{l_1, l_2} \tilde{p}(y|l_2, l_1, x) q(l_2|l_1, x) q(l_1|x) \\
&= \sum_{l_1, .., l_n} \tilde{p}(y|l_n, ..., l_1, x), ..., q(l_2|l_1, x) q(l_1|x)
\end{aligned}$$

At a tree node, if the exact model can be learned, then

$$\tilde{p}(y|l_i, .., l_1, x) = \sum_{l_{i+1}} \delta(y = l_{i+1}) q(l_{i+1}|l_i, .., l_1, x),$$

which means the model $q(l_{i+1}|l_i, .., l_1, x)$ perfectly predicts the $y$ and, thus, the tree stops expanding. The augmented variables $l_i$ gradually decouples $y$ from $x$ to make a better prediction.

## 3.2. Multi-class Probabilistic Boosting-Tree

Section (3.1) introduces the two-class boosting-tree algorithm. Traditional boosting algorithms for multi-class classification [8, 9] require multi-class weak classifiers, which are in general much more computationally expensive to learn and compute than two-class weak classifiers. This is specially a problem when the number of class becomes large. Interestingly, different classes of patterns are often similar to each other in a certain aspect. For example, a donkey may look like a horse from a distance. Torralba et al. [18] designed a multi-class classification algorithm for different classifiers to share features/weak classifiers. However, each individual class still has its own classification function and it is not clear how a unified posterior distribution can be obtained. Also, clustering is not directly related to the learning procedure in their approach.

Next, we give the training algorithm for the multi-class boosting tree in Fig. (6).

To train a tree of maximum depth of $L$:

- Given: A training set $S = \{(x_1, y_1, w_1), ..., (x_m, y_m, w_m); \quad x_i \in \chi, \quad y_i \in \{1, .., n\}\}$, $\sum_i w_i = 1$.
- Compute the empirical distribution $\hat{q}(y) = \sum_i w_i \delta(y_i = y)$.
- For each feature $h_j$ at value $\nu_j$, compute the histogram $his_{left}(k) = \frac{1}{Z_{left}} \sum_i \delta(k = y_i) w_i$ for $y_i < \nu_j$ and $his_{right}(k) = \frac{1}{Z_{right}} \sum_i \delta(k = y_i) w_i$ for $y_i \geq \nu_j$
- Find the optimal $h_j$ and $\nu_j$ that achieve the minimum entropy $Z_{left} Entropy(his_{left}) + Z_{right} Entropy(his_{right})$.
- Create a new set $S' = \{(x_1, y'_1, w_1), ..., (x_m, y'_m, w_m); \ x_i \in \chi, \ y_i \in \{-1, +1\}$
- $y'_i = -1$ if $Z_{left} his_{left}(y_i) \geq Z_{right} his_{right}(y_i)$ and $y'_i = +1$ otherwise. (A sampling strategy can be adopted also.)
- Now it becomes a two-class learning problem. Follow the same procedure as in Fig. 2
- Repeat the procedure.

Figure 6: Training for multi-class probabilistic boosting-tree.

This algorithm first finds the optimal feature that divides the multi-class patterns into 2 classes and then use the previous two-class boosting-tree algorithm to learn the classifiers. Interestingly, the first feature selected by the boosting algorithm after transforming the multi-class into two-class is often the one chosen for splitting the multi-classes. Intuitively, the rest of the features/weak classifiers picked are supporting the first one to make a stronger decision. Thus, the two-class classification problem is a special case of the multi-class classification problem. Similar objects of different classes, according to the features, are grouped against the others. As the tree expansion continues, they are gradually clustered and set apart. The expansion stops when each class has been successfully separated or there are too few training samples. The testing procedure for multi-class PBT is nearly the same as that in the two-class problem. Again, the top node of the tree integrates all the probabilities from its sub-trees, and outputs an overall posterior probability. The scale of the problem is $O(log(n))$ w.r.t. to the number of classes, $n$. This multi-class PBT is very efficient in computing the probability due to the hierarchical structure. This is important when we want to recognize hundreds or even thousands of classes of objects, which is the problem human vision systems are dealing with every day. In the worst case, each tree node may be traversed. However, this is rarely the case in practice.

## 3.3. Relations with the Existing Methods

The proposed framework has many interesting connections to other methods. We briefly discuss them below.

*Cascade approaches*

The cascade approach used together with AdaBoost [21, 23] has shown to be very effective in doing rare event detection. We view the cascade method as a special case of the boosting-tree. In the cascade, a threshold is picked such that all the positive samples are pushed to the right side of the tree. However, pushing positives to the right side may cause a big false positive rate, specially when the positives and negatives are hard to separate. The boosting-tree, instead, naturally divides the training sets into two parts. In the case where there are many more negative samples than positive samples, most of the negatives are passed into leaf nodes close to the top. Deep tree leaves focus on classifying the positives and negatives which are hard to separate.

*Decision trees*

Decision tree algorithms [1] have been widely used in vision and AI. The key difference here is that each tree node is a strong decision maker and it learns a distribution $q(y|x)$. Whereas in the traditional decision tree, each node is a weak decision maker and thus the result at each node is more random. Also, boosting-trees are much more compact.

*$A^*$ algorithm*

The $A^*$ algorithm [3] is a heuristic search algorithm and it is guaranteed to find the global optimum if the heuristics are always conservative. At each tree node, we also face the problem of which side to go into. In the worst case, the entire tree is traversed. Similar to the $A^*$ algorithm, we use the learned $q(y|x)$ at each node in guiding our search for propagating probabilities.

*Generative model and EM*

As shown before, the PBT implicitly augments data to better approach the posterior. At the top of the node, it gathers information from its descendant trees by integrating out the augmented data, similar to the setup in EM. However, this augmented data may not directly relate to the hidden variables which control the underling generation processes of the $x$ in the generative model. This causes the framework to be hard to deal with, e.g., objects which are highly flexible. The positive side is that we don't need to specifically model any hidden variables.

*Semantics and grammar*

Grammar has been extensively used in AI but shows very little success in vision. It is often easy to define a grammar but the difficulty mostly comes in the computing stage. The tree structure here automatically picks the features/evidences/knowledge and implicitly combines them into semantics. It captures the sequential relations and configurations among the features, which can lead to high-level semantics.

## 4. Experiments

In this section, we show two applications using the proposed framework: object categorization and multi-view face detection.

### 4.1. Object Categorization

Several datasets have been recently provided for testing object categorization/recognition algorithms. We design an object categorization algorithm based on multi-class PBT proposed in the paper. The algorithm is tested on the ETH-80 dataset by Leibe and Schiele [15], the NORB database by LeCun et al. [13], and the Caltech-101 categories by Fei-Fei et al. [5].

**(I) ETH-80 dataset**

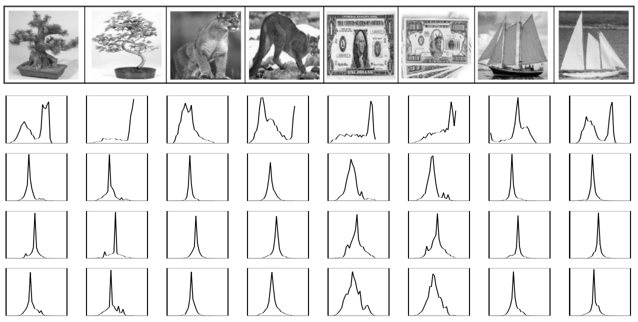


Figure 7: Histograms of four object images from [5] on intensity and three Gabor filtering results. Eight images here are from four categories, "bonsai", "cougar body", "dollar bill", and "ketch".

Haar type filters have been shown very effective in detecting faces. However, objects in the same category show a great deal of variation and the correct recognition of them often needs the help from high-level knowledge. PCA based patches are used to characterize object parts in [7, 5]. However, each category has it own set of PCAs and it is not clear how to generalize them to cover all the categories. Histograms [22] are shown to be robust against translation and rotation and have good discriminative power. Histograms on different filter responses of an image serve as different cues, which can be used and combined to perform scene analysis and object recognition. Fig. (7) shows four histograms of eight images on their intensity and Gabor filter responses. To learn discriminative models, we compute up to the 3rd order moments for each histogram $h(s)$ to make use of integral image for fast computing.

The goal is to learn a discriminative model so that it outputs the posterior distribution on the category label for each input image patch. Each object image is resized into an $80 \times 80$ patch. For each image patch, we compute the edge maps using Canny edge detector at three scales, orientation of the edges, and filtering results by 10 Gabor filters. These are the cue images for the image patch. We put 1,000 rectangles with different aspect ratios and sizes centered at various locations in the $80 \times 80$ image patch. Features are the moments of the histogram for each rectangle on every cue image. The multi-class PBT algorithm then picks and combines these features forming hierarchical classifiers. We randomly pick 29 out total 80 categories in the ETH-80 image set to illustrate the algorithm. There are 41 images for each category and some of them are shown in Fig. (8). These images are taken for objects at different view directions and illuminations. We randomly pick 25 images out of each category for training. Fig. (8) shows the clusters formed in the boosting-tree after learning. We can see the the algorithm is capable of automatically discovering the intra-class similarity and inter-class similarity and dissimilarity. For the images not picked in training we test the recognition/categorization rate. The one with the highest probability is considered as correct recognition. Table (1) shows the recognition rate on the remaining 16 images for each category.

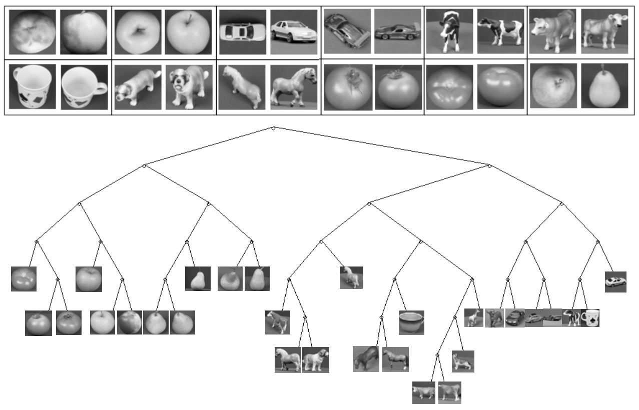


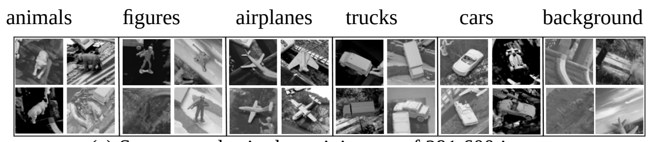
Figure 8: Some samples image in ETH-80 and the clusters learned. In the second row, only one image in the category is shown for illustration.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| apple1 | 100% | cup1 | 100% | tomato3 | 100% | horse1 | 94% |
| pear10 | 94% | apple4 | 94% | pear3 | 94% | pear9 | 94% |
| cup4 | 88% | cow1 | 88% | pear8 | 88% | dog2 | 81% |
| car1 | 81% | pear1 | 81% | apple3 | 75% | car9 | 75% |
| tomato1 | 75% | tomato10 | 75% | horse3 | 75% | cup9 | 75% |
| dog10 | 69% | dog1 | 69% | horse8 | 69% | car11 | 56% |
| car11 | 56% | cow2 | 50% | cow10 | 44% | horse10 | 44% |
| cow8 | 0.19 | | | | | | |

Table 1: Recognition results on the ETH-80 image set. 25 images in each category are randomly picked for training. The recognition rate is computed based on the remaining 16 image in each category. The average recognition rate is $76\%$

**(II) NORB jittered-textured dataset**

LeCun et al. created a data set [13] in which there are 5 generic categories, namely, four-legged animals, human figures, airplanes, trucks, and cars. There is an extra category of background for training. There are 10 object instances for each category. 1,944 stereo pairs of images for each object instance are captured under different viewing angles and lighting conditions. They embedded the captured images into different backgrounds to create different datasets. We consider single image only and focus on the most difficult dataset, jittered-textured. The 10 object instances are split into two groups, with 5 for training and 5 for testing. This gives a total of 291,600 training images and 291,600 testing images. Some of the training images are shown in Fig. (9).a and some testing images are displayed in Fig. (9).b. We use the same algorithm for ETH-80 and train it with 29,160 training samples of size $80 \times 80$. The categorization rate on the testing set after training is 75.3%, which is an improvement over the rate 60.1% reported in [13].



(a) Some samples in the training set of 291,600 images.

(b) Some samples in testing set of 291,600 images.

Figure 9: Some of the training and testing samples from the NORB jittered-cluttered dataset. There are 5 generic categories: four-legged animals, human figures, airplanes, trucks, and cars. The sixth category is random background image patch. The overall classification rate is 75.3%.

**(III) Caltech-101 dataset**

The task of categorizing Caltech-101 image categories is much more challenging. Some of the typical images are shown in Fig. (1). Instead of working on the original images, we cropped all the objects out and resize them to $80 \times 80$. Learning and testing are performed based on the cropped image. We use the same learning algorithm for the ETH-80 and NORB. 25 images are randomly selected from each category for training. Fig. (10) shows some of the clusters formed after training. However, the clusters are sparser than the ones in the ETH-80 case due to complex object categories. For each image category $y_j$, we compute the histogram

$$h(N) = \sum_i \delta(N - N(x_i))$$

where $N$ is a leaf node and $N(x_i)$ is the leaf node at which training sample $x_i$ is finally located. The entropy of the $h(N)$ tells us how tight the samples of each category are in the tree. For objects which are "similar" to each other in the category, they should form tight clusters. Wheres for objects having large variation, they are more scattered among the tree. In table 2, the third column after the category name gives the entropy measure for each category. Object categories like "ying yang" have very low entropy and, not surprisingly, the background category has the most

| object | r1 | r2 | entropy | object | r1 | r2 | epy |
|---|---|---|---|---|---|---|---|
| inline skate | 100% | 100% | 0.39 | garfield | 100% | 100% | 1.26 |
| yin yang | 83% | 83% | 0.52 | stop sign | 66% | 81% | 1.09 |
| revolver | 63% | 80% | 1.16 | metronome | 63% | 75% | 1.26 |
| dollar bill | 61% | 78% | 0.80 | motorbikes | 56% | 75% | 0.52 |
| . | . | . | . | . | . | . | . |
| joshua tree | 0% | 19% | 1.19 | beaver | 0% | 25% | 1.36 |
| chair | 0% | 9% | 1.75 | wild cat | 0% | 22% | 1.56 |
| crab | 0% | 0% | 1.92 | background | | | 2.82 |

Table 2: The categorization rates and entropy measure for the Caltech-101 image set [5]. 25 images are randomly picked for training for each category. Categorization is performed on the rest of the image in each category. Note that the number of images in each category varies a lot. $r1$ is measured as the category id has the biggest probability, and the average rate is $20\%$. $r2$ is measured as the id is among the top ten, and the average rate is $40\%$.

variability and the highest entropy. This entropy measure loosely tells us how difficult it will be to recognize each category. The categorization/recognition result is shown in table 2. The first column after the category name, $r1$ , is the recognition rate when the discriminative model outputs its category id as the one with the highest probability. The average recognition rate for $r1$ is $20\%$. A random guess would get the rate around $1\%$. The second column, $r2$, is the categorization rate when the category id is among the top ten choices. The average rate for $r2$ is $40\%$. The result in [5] is 16%. Recently, Berg et al. have obtained a rate of 48% using geometric blur and shape matching. The intention of the experiment using PBT in this paper is to illustrate a discriminative approach. Its advantage is the computational efficiency and intrinsic clustering. A generative approach is expected to outperform a discriminative approach, when the training samples are limited. The discriminative approach is better suited when a large number of training samples are available, like in the NORB dataset. To design an algorithm effectively detect and recognize a wide class of objects in the scene, an integration of discriminative and generative methods is probably needed.

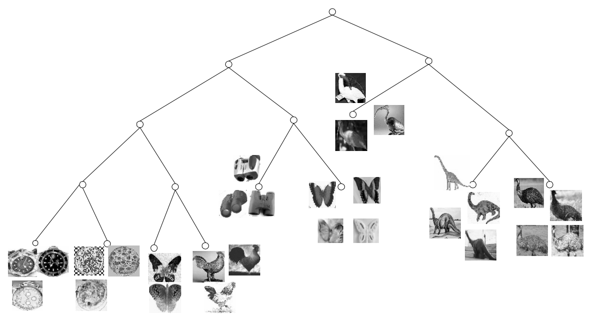


Figure 10: Some of the clusters formed in the 101 image category by PBT.

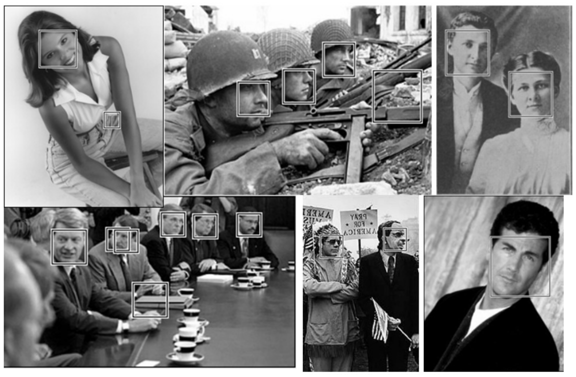


Figure 11: Some face detection results tested on the CMU frontal and profile image dataset.

## 4.2. Multi-view Face Detection

There has been significant progress made recently in the area of face detection [21, 17, 16, 10, 6], most of which focuses on faces in frontal view. In [21, 16], viewing angles are descritized and each training image is clustered beforehand. Instead, PBT can be applied for multi-view face detection without the explicit pre-clustering stage. The Haar features proposed by Viola and Jones [21] are used to form weak classifiers. The CMU frontal and profile training sets are used for testing. We have obtained some preliminary results and some of which are shown in Fig. 11. The speed of the algorithm is comparable to the algorithm in [21]. The result can be improved by using a larger template of a face and obtaining more training samples.

## 5. Discussions and Conclusions

In this paper, a new framework–probabilistic boosting-tree is introduced for learning and computing two-class and multi-class discriminative models. The PBT learns a target distribution by constructing a tree in which each node is a strong decision maker. Learning is carried out in a hierarchical way by a divide-and-conquer strategy. Also, clustering is intrinsically embedded in the learning procedure. The proposed framework integrates aspects of many existing algorithms such as decision tree, cascade approaches, boosting algorithms, and the $A^*$ algorithm. Experiments are reported on two challenging tasks in vision: multi-class object categorization/recognition and multi-view face detection. The scalability of the framework is very attractive due to its hierarchical structure and it has the potential to model hundreds of classes of patterns.

**Justification for probability**

Again, being able to compute the discriminative probability is very important for many problems in vision. It is especially useful for scene understanding in which objects have complex configurations and hard decisions can not be made locally. In separate work, we used PBT for computing pairwise affinity for grouping object parts in [20]. Also, we have applied the PBT to learn a complex appearance model for 3D segmentation.

**Limitations**

One major problem with the framework is overfitting, especially when there is limited number of training samples. The current framework can almost alway separate two samples, and this may overfit the training data. In the PBT algorithm, overfitting is controlled by limiting the maximum depth of a tree and passing confusing samples to both sides of the tree. Another possible way is to perform tree pruning [4]. The overfitting problem deserves further investigation.

**Acknowledgment**

I thank Xiang (Sean) Zhou, Xiangrong Chen, Tao Zhao, Adrian Barbu, Piotr Dollar, and Xiaodong Fan for helpful comments and assistance in getting some of the training datasets.

## References

[1] Y. Amit, D. Geman and K. Wilder, "Joint induction of shape features and tree classifiers", *PAMI*,19, 1300-1305, 1997.
[2] A. Berg, T. Berg, and J. Malik, "Shape Matching and Object Recognition using Low Distortion Correspondence",*Proc. of CVPR*, 2005.
[3] J. Coughlan and A. Yuille, "Bayesian A* Tree Search with Expected O(N) Node Expansions: Applications to Road Tracking", *Neural Computation*, 2003.
[4] F. Esposito, D. Malerba, and G. Semeraro, "A Comparative Analysis of Methods for Pruning Decision Trees", *PAMI*, vol. 19, no. 5, 1997.
[5] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models for 101 object categories", *Computer Vision and Image Understanding*, in press.
[6] F. Fleuret, and D. Geman, "Coarse-to-Fine face detection", *IJCV*, June, 2000.
[7] R. Fergus, P. Perona, and A. Zisserman, "Object Class Recognition by Unsupervised Scale-Invariant Learning", *CVPR*, 2003.
[8] Y. Freund and R. Schapire, "A Decision-theoretic Generalization of On-line Learning And an Application to Boosting", *Proc. of ICML*, 1996.
[9] J. Friedman, T. Hastie and R. Tibshirani, "Additive logistic regression: a statistical view of boosting", Dept. of Statistics, Stanford Univ. Technical Report. 1998.
[10] C. Garcia and M. Delakis, "Convolutional Face Finder: A Neural Architecture for Fast and Robust Face Detection", *PAMI*, vol. 26, no. 11, 2004.
[11] E. Grossmann, "AdaTree: Boosting a Weak Classifier into a Decision Tree", *Proc. CVPR workshop on learning in computer vision and pattern recognition*, 2004.
[12] S. Kumar and M. Hebert, "Discriminative Random Fields: A Discriminative Framework for Contextual Interaction in Classification", *Proc. of ICCV*, 2003.
[13] Y. LeCun, F. Huang, and L. Bottou, "Learning Methods for Generic Object Recognition with Invariance to Pose and Lighting", *CVPR*, 2004.
[14] G. Lebanon and J. Lafferty, "Boosting and Maximum Likelihood for Exponential Models", *Proc. of NIPS*, 2003.
[15] B. Leibe and B. Schiele, "Analyzing Appearance and Contour Based Methods for Object Categorization" , *Proc. of CVPR*, 2003.
[16] S. Z. Li and Z. Zhang, "FloatBoosting Learning and Statistical Face Detection", *IEEE Trans. PAMI*, vol. 26, no. 9, Sept. 2004.
[17] H. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection", *IEEE Trans. PAMI*, vol. 20, 1998.
[18] A. Torralba, K. P. Murphy, and W. T. Freeman, "Sharing Visual Features for Multiclass and Multiview Object Detection", *Proc. of CVPR*, 2004.
[19] Z. Tu, X. Chen, A. Yuille, and S.C. Zhu, "Image Parsing: Segmentation, Detection, and Object Recognition", *Proc. of ICCV*, 2003.
[20] Z. Tu, "An Integrated Framework for Image Segmentation and Perceptual Grouping", *Proc. of ICCV*, 2005.
[21] P. Viola and M. Jones, "Fast Multi-view Face Detection", *Proc. of CVPR*, 2001.
[22] S.C. Zhu, Y. Wu, and D. Mumford, "Minimax Entropy Principle and Its Application to Texture Modeling", *Neural Computation*, vol. 9, no. 8, 1997.
[23] J. Wu, J. M. Rehg, and M. D. Mullin, "Learning a Rare Event Detection Cascade by Direct Feature Selection", *Proc. of NIPS*, 2003.