From Bag of Categories to Tree of Object Recognition

Haijing Wang¹, Tianwen Zhang¹ and Peihua Li²

¹Harbin Institute of Technology, China.
²Heilongjiang University, China.
Email: haijing.wang@yahoo.com

Abstract

We consider the problem of reducing multiclass categorization to multiple binary problems to recognize different category of objects. Traditional approaches require training different classifiers for each category. This can be slow and the performance of learned single classifier is poor for limited training samples. We present a multiclass object recognition tree, in which each leaf node corresponds to one category, and each non-leaf node is a set of classifiers to distinctive categories locating at the left and right subset of categories in the tree. The left and right subset corresponds to a bag of categories, respectively. Each classifier in non-leaf node is trained to capture the shared features of a bag of categories by AdaBoost. Recognition is then a progress to find a path from the root to a leaf, which represents a unique category. The very promising result on CalTech 101 database shows the robustness of the proposed approach.

Keywords: object recognition, bag of categories, multiclass object recognition tree, AdaBoost

1 Introduction

Object recognition [1, 2, 3, 4, 5] is a fundamental vision problem: put simply, what’s in the image, and where? To recognize different categories of objects in images, people usually reduce multiclass categorization problem to binary problem [6], in which each class is compared against all others, or all pairs of classes are compared to each other, or error correcting output codes (ECOC) are used. They all ignore one fact that the subset of categories may exist some common features. And these traditional approaches require training different classifiers for each category. This can be slow and the performance of learned single classifier is poor for limited training samples.

We propose a novel concept “bag of categories” to show how to share the common features among different categories. Just like in the context of document analysis, bag-of-words [7] assumes that the order of words in a document can be neglected. In computer vision, the bag of keypoints [8, 9] method is based on vector quantization of affine invariant descriptors of image patches. Bag-of-features [10, 11] methods represent an image as an orderless collection of local features. Bag of patches and bag of codewords [12] are also appeared in the context of vision analysis.

For “bag of categories”, two questions should be answered:

(Q1) How to combine these bags of categories?

(Q2) How to describe each bag of category?

For the first question, we built a multiclass object recognition tree, in which each leaf node corresponds to one category, and each non-leaf node is a set of classifiers to distinctive categories locating at the left and right subset of categories in the tree. We call each left and right subset of tree as “a bag of categories”, respectively. That is, bag of categories are combined by object recognition tree.

For the second question, each classifier in non-leaf node is trained to capture the shared features of a bag of categories by AdaBoost [13, 14].

In the proposed framework, object recognition turns into a progress to find a path from the root to a leaf, which represents a unique category, instead of the common nearest neighbor algorithm.

The whole paper is scheduled as follows. Binary decision tree for multiclass object recognition is presented in Section 2. Details of bag of categories are introduced in Section 3. In Section 4, experiment design and analysis result are given. Finally, the conclusion of the paper is presented.

2 Binary Decision Tree for Combining Bags of Categories

Decision tree employs a hierarchically structured decision function in a sequential fashion. To combine bags of categories, the binary decision tree
is constructed. For each non-leaf node, it corresponds to a binary classification. Thus, multiclass object recognition is transferred into many binary classification problems. It is the basis to construct a tree framework. In this section, we first describe the elements of the recognition tree and the splitting rules. Then we present the algorithm of tree construction for combining bags of categories.

2.1 From Multiclass Problem to Binary Classification Problem

Suppose we have \( m \) samples, each of which corresponds to an image,

\[ \{ x_i | x_i \in X, X \subset \mathbb{R}^n \}_{i=1}^{m} \]

where \( n \) is equal to the number of image pixels. The corresponding class label of each sample is

\[ \{ y_i | y_i \in Y, Y = \{1, \ldots, C\} \}_{i=1}^{m} \]

where \( C \) is the total number of categories.

There are many ways to reduce a multiclass problem to multiple binary classification problems [6], such as one-against-all approach, all-pairs approach, and error correcting output codes (ECOC).

- One-against-all approach is the simplest approach to create one binary problem for each of the \( C \) categories. That is, for any category label \( r \in Y \), all examples labeled \( y_i = r \) are considered as one class and all other examples are considered as the other class.

- In all-pairs approach, for each distinct label pair \( r, s \in Y \), examples labeled \( y_i = r \) are considered as one class, and those labeled \( y_i = s \) belong to the other class. All other examples left are simply ignored.

- Error correcting output codes (ECOC) is to associate each class \( r \in Y \) with a row of a “coding matrix” \( M \in \{-1, 0, +1\}^{C \times l} \) for some \( l \), where \( l \) is number of classifiers. The binary classification problem is then run once for each column of the matrix.

Different from the one-against-all, all-pairs, and ECOC approach, we construct the multiclass object recognition tree by subdividing the whole category set into two subsets, as the two sides of the binary classification tree, and recursively to apply this kind of subdivision until all category is distinguished each other. To construct the decision tree, some elements should be considered first.

2.2 Elements of Decision Tree

Binary decision tree is known as a class of nonlinear classifiers. The root node of the tree \( T \) is associated with the training set \( X \), which corresponds to all categories to be recognized. Each node, \( t \), represents a specific subset \( X_t \) of the training set \( X \), which corresponds to a subset of all categories. We call categories in one subset as “a bag of categories”, since we describe these categories in the subset using the same classifiers with sharing features. Splitting of a node is equivalent to split the subset \( X_t \) into two disjoint descendant subsets, \( X_{tL}, X_{tR} \). For every split, the following equation (1) and (2) need to be satisfied:

\[ X_{tL} \cap X_{tR} = \Phi \]  
\[ X_{tL} \cup X_{tR} = X_t \]

Here \( X_{tL} \) and \( X_{tR} \) correspond to two bags of categories.

In order to develop a binary decision tree, the following design elements have to be considered in the constructing phase:

1. **A splitting criterion must be adopted according to which the best split from the set of candidate one is chosen**.
2. **A stop splitting rule is required to control the growth of the tree and to declare a node as a terminal leaf**.
3. **A rule is required that assigns each leaf to a specific category**.

A leaf node is formed when the subset \( X_t \) only includes the samples which belong to a single category. That is, a leaf node corresponds to a single category. And non-leaf node is a bag of categories. The code word from root to each leaf node represents a specific category. Details on splitting criterion are described in the next subsection.

2.3 Splitting Criterion

The descendant nodes are associated with two new subsets, i.e., \( X_{tL}, X_{tR} \), respectively. For the tree growing methodology, from the root node down to the leaves, every split must generate subsets that are more “class homogeneous” compared to the ancestor’s subset \( X_t \). This means that the training feature vectors in each one of the new subsets show a higher preference for specific categories, whereas data in \( X_t \) are more equally distributed among the categories. The splitting algorithm is described as follows.

Algorithm 1. Splitting algorithm

Input:
• Matrix of feature values $G_t$ of samples $X_t$ with corresponding label set $Y_t$.
• Desired number of clusters $L$, here $L = 2$ for left and right node of the binary tree.
• Iteration times $N$ of splitting.

Algorithm:

(S1) Choose arbitrary initial clustering estimates $\theta_j(0)$ for the $\theta_j$’s. Here $j = 1, 2$ corresponds to the left and right node, respectively.

(S2) Repeat $N$ iterations.
For $i = 1$ to $m_t$ ($m_t$ is the number of samples to be split)
– Determine the closest representative, say $\theta_j$, for $g_i$.
– Set $b(i) = j$.
For $j = 1$ to $L$
– Parameter updating: Determine $\theta_j$ as the mean of the vectors $g_i \in G$ with $b(i) = j$.

(S3) For each category, count number of samples located in each child node.
For $j = 1$ to $L$
For $k = 1$ to $C_t$ ($C_t$ is the number of categories to be split.)
– Statistics
$$\lambda_{jk} = \sum_{i=1}^{m} \delta(x_i)$$
where
$$\delta(x_i) = \begin{cases} 
1 & \text{if } y_i = k \text{ and } b(i) = j \\
0 & \text{otherwise} 
\end{cases}$$

(S4) Decide the final splitting
For $i = 1$ to $m_t$
– Set $b(i) = \begin{cases} 
1 & \text{if } \lambda_{1y_i} > \frac{\lambda_{1y_i} + \lambda_{2y_i}}{2} \\
2 & \text{otherwise} 
\end{cases}$

Output:
• $X_{tL} = \{x_i | b(i) = 1\}, X_{tR} = \{x_i | b(i) = 2\}$

2.4 Algorithm for Constructing Binary Decision Tree

After the discussion on the major elements needed for the growth of a decision tree, we are now ready to summarize the basic algorithmic steps for constructing a binary decision tree for bag of categories.

Algorithm 2. Tree constructing algorithm

(S1) Begin with the root node, i.e., $X_t = X$
(S2) For each new node $t$
– Learn a group of classifiers with AdaBoost based on sample set $X_{tL}$ and $X_{tR}$.
(S3) Do all samples of the subset $X_t$ belong to a single category?
– If yes, stop splitting and declare node $t$ as a leaf and designate it with a category label. End.
– If not, generate two descendant nodes $t_L$ and $t_R$ with associated subsets $X_{tL}$ and $X_{tR}$. Return to (S2).

We first classify the whole sample set into two subsets. In the first stages of binary tree, root node includes all categories to be classified. The existed classifiers in the node of tree are used to estimate the image’s category label for its children nodes. Next, we classify each subset into two new subsets following Equ.(1) and Equ.(2). Each subset corresponds to a bag of categories. We end up the classification until each category is a sole part. This is a binary decision tree model for multiclass object recognition.

3 Bag of Categories for Visual Category Vocabulary

The classifier with selected features in each node of the tree corresponds a visual category vocabulary. To make the visual category vocabulary describe the bag of categories more clearly, we learn a group of classifiers for each node. Since these classifiers are independent, we propose a two-stage voting strategy to help the final decision of each bag of categories. After describing the bag of categories with visual category vocabulary and combing the bag of categories with binary decision tree, the recognition process for a new sample can be finished by the traversal of the tree from root to leaf.

3.1 Visual Category Vocabulary

Each category vocabulary is a AdaBoost strong classifier. Once the samples are grouped into a
binary tree, we train one classifier for each node of the binary tree. AdaBoost[13, 14] is used to train each classifier, and original haar-like features proposed by viola and Jones[15] are taken as feature set. AdaBoost algorithm, proposed in the Computational Learning Theory literature, is a method to find a highly accurate hypothesis (a strong classifier) by combining many “weak” hypotheses, each of which is based on the reweighted version of the training data in order to emphasize those which are incorrectly classified by previous weak classifiers, and only moderately accurate. The final strong classifier is a weighted combination of weak classifiers followed by a threshold. The decision of a strong classifier is bias. Thus, we learn a group of independent strong classifiers at each node. The final decision is created with the voting rule described as follows.

3.2 Voting Rule

The combined decision is obtained by a majority vote of the individual classifiers. Majority vote does not assume prior knowledge of the behavior of the individual classifiers.

We assume that there are $n$ classifiers, and each classifier produces a unique decision regarding the identity of the sample. In our approach, the following conditions are satisfied:

(C1) The number of voters is odd.

(C2) Each voter has the same probability of voting one way.

(C3) The individual decisions is independent, since each classifier is trained independently and is based on different feature set.

We take two-stage voting strategy.

Algorithm 3. Two-stage voting algorithm

(S1) If $|k_t - \frac{n_t + 1}{2}| > \tau$, the sample is assigned to the side when $k$ experts are agreed.

(S2) If $|k_t - \frac{n_t + 1}{2}| \leq \tau$, consider two descendant nodes $t_L$ and $t_R$ with associated subsets $X_{tL}$ and $X_{tR}$.

- If $|k_{tLL} + k_{tLR} - n_{tL}| > |k_{tRL} + k_{tRR} - n_{tR}|$, the sample is assigned to the left node.
- If $|k_{tLL} + k_{tLR} - n_{tL}| < |k_{tRL} + k_{tRR} - n_{tR}|$, the sample is assigned to the right node.
- If $|k_{tLL} + k_{tLR} - n_{tL}| = |k_{tRL} + k_{tRR} - n_{tR}|$, the sample is assigned to the node randomly.

Here $\tau$ is voting factor. $k_1$, $k_{LL}$, and $k_{LR}$ are the number of experts (classifiers) agreeing on the identity in the node, left child, and right child node, respectively. $k_{tLL}$ and $k_{tLR}$ are left and right child of the left child $t_L$. And $k_{tRL}$ and $k_{tRR}$ are left and right child of the right child $t_R$.

3.3 Category Encoding

In the proposed framework, each node is a set of classifiers. The bag of categories refers to a set of classifiers to describe a group of categories. More precisely, the category recognition is a two-step process.

(a) The local regions (patches) corresponding to the features of each classifier are sampled from image.

(b) The sample is assigned to the left or right node of the binary decision tree.

We repeat (b) until we reach a leaf node classifier to make a global decision about the test image.

Decision tree is a multistage system, in which categories are sequentially rejected until we reach a final accept category. To this end, the feature space is split into unique regions, corresponding to each category, in a sequential manner. Recognition is achieved from the root node to the leaf node. Each leaf node corresponds to a code book, and each category can be noted by a binary code. This is a deep first traversal of tree.

4 Experiments

We use the Caltech 101 object class dataset. The Caltech 101 dataset\(^1\) contains 9,197 images comprising 101 different object categories, plus a background category, collected via Google image search in September 2003 by Fei-Fei Li, Marco Andreetto, and Marc Aurelio Ranzato [16]. Each category contains about 40 to 800 images. Most categories have about 50 images.

In our experiment, “Faces” and “Faces_easy” category are combined into one category. Therefore, the total number of categories is 100. Fig. 1 statistics the average width and height of CalTech 101 samples. In Fig. 1, fifteen pairs of width and height are excluded because their small values in width, height, or both. In order to accelerate the training speed, we resize the width and height to $100 \times 90$ following the scale of average width and height $230 \times 265$.

Thirty samples were chosen randomly from each of the 100 object categories, yielding a total 3000

\(^1\)www.vision.caltech.edu/ImageDatasets/Caltech101/
Average width vs. height of CalTech 101 samples. Each circle represents a pair of width and height of one category. Some small width and height pairs are excluded. The solid circle point with coordinate (230, 205) is the average width and height of samples for all categories.

Figure 1: Average width vs. height of CalTech 101 samples. For each class, testing images are ones excluding those used as training samples. Samples are not alignment before training.

Experimental results on Caltech 101 dataset show the robustness of our proposed approach. The results are shown in Table 1 and Fig. 2 for both 15 and 30 training samples per category.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Samples</th>
<th>15</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fei-Fei, Fergus, &amp; Perona [2]</td>
<td></td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>Holub, Perona, &amp; Welling [5]</td>
<td></td>
<td>37</td>
<td>43</td>
</tr>
<tr>
<td>Berg, Berg, &amp; Malik [1]</td>
<td></td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>Mutch &amp; Lowe [3]</td>
<td></td>
<td>51</td>
<td>56</td>
</tr>
<tr>
<td>Proposed approach</td>
<td></td>
<td>42</td>
<td>56.3</td>
</tr>
</tbody>
</table>

Table 1: Correct rate in percentage with 15 or 30 training samples per category on CalTech 101 dataset.

The final decision tree of our experiment based on CalTech 101 dataset is given in Fig3. Each row corresponds to the coding of each category. For example, the ID of the second row is 23, and the code “00000010” is the code word of the category 23 named “crab”. It exists in the left child node from stage 1 to stage 6, the right child node of the stage 7, and the left child node of the final stage 8.

5 Conclusion

We cluster multiple categories into two different subset, in which a subset corresponds to a bag of categories, then build a multiclass object recognition tree. Recognition is a progress to find a path from the root to a leaf, which represents a unique category, instead of the common nearest neighbor algorithm.

We answer three main issues for object recognition. How do we represent the object? We represent object by code book and bag of categories. Using this representation, how do we learn a particular object category? We construct binary decision tree for multiclass object recognition. And finally how do we use the model we have learnt to find further instances in query images? This is a deep first traversal of tree.

References


