Alternating Decision Forests

Yufeng Jiang

Abstract

This paper introduces a novel clssification method termed Alternating Decision Fforestl (ADFs), which formulates the training of Random Forests explicitly as a global loss minimization problem. During training, the losses are minimized through keeping an adaptive weight distribution over the training samples, similar to Boosting methods. In order to keep the moethod as flexible and general as possible, authors adopt the principle of employing gradient descent in function space, which allows to minimize arbitrary losses. Futhermore, they also show how ADFs can be easily integrated into an object detection application.

1. Introduction

In recent years, Random Forests [1, 2] (RFs) have emerged as very useful classifiers for a large variety of computer vision tasks, including object recognition, semantic segmentation, or data clustering. Although the method is relatively simple, it has many characteristics that make it particularly interesting for computer vision problems: (i) fast training and evaluation, (ii) robustness to label noise, (iii) inherent multi-class capability, (iv) suitability for parallel processing, and (v) good performance for high-dimensional input data [3]. Finally, RFs are able to yield state-of-the-art performance in various classification and regression tasks and compare favorably with other machine learning algorithms [4].

In this paper, authors propose a novel classifier termed $AlternatingDecisionForests(ADFs)$ that extend RFs by globally minimizing any given differentiable loss function, however, without losing the main characteristics and benefits of the original RF method as discussed above. They achieve this by borrowing ideas from Boosting methods, where during training a globally tracked weight distribution guides the loss minimization.

2. Alternating Decision Forests

Before introducing and discussing Alternating Decision Forests, authors shortly review standard Random Forests.

Random Forests [1, 2] are ensembles of T binary decision trees $\mathcal{T}_t(\mathbf{x}) : \mathcal{X} \rightarrow \mathcal{R}^K$, where $\mathcal{X} = \mathbb{R}^M$ is the M-dimensional feature space and $\mathcal{R}^K = [0,1]^K$ descibes the space of class probability distributions over the label space $\mathcal{Y} = \{1, \ldots, K\}$. During testing, each decision tree thus returns a class probability distribution $p_t(y|\mathbf{x})$ for a given test sample $\mathbf{x} \in \mathbb{R}^M$, and the final class label y^* is then obtained via averaging:

$$y^* = \arg\max_y \frac{1}{T} \sum_{t=1}^{T} p_t(y|\mathbf{x}). \tag{1}$$

During training of a RF, the decision trees are provided with a random subset of the training data (*i.e.* bagging [2]) and are trained independently from each other. Training a single decision tree involves recursively splitting each node such that the training data in the newly created child nodes is pure according to class labels. Each tree is grown until some stopping criterion, *e.g.* the maximum tree depth, is reached and the class probability distributions are estimated in the leaf nodes.

A splitting function $s(\mathbf{x}; \Theta)$ is typically parameterized by two values: (i) a feature dimension $\Theta_1 \in \{1, \ldots, M\}$ and (ii) a threshold $\Theta_2 \in \mathbb{R}$. The splitting function is then defined as

$$s(\mathbf{x}; \Theta) = \begin{cases} 0 & \text{if } \mathbf{x}(\Theta_1) < \Theta_2 \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

where the outcome defines to which child node the sample $\mathbf{x}$ is routed.

Each node chooses the best splitting function Θ^* out of a randomly sampled set $\{\Theta^i\}$ by optimizing

$$I = \frac{|L|}{|L| + |R|} H(L) + \frac{|R|}{|L| + |R|} H(|R|), \tag{3}$$

where L and R are the sets of data samples that are routed to the left and right child nodes, according to $s(\curvearrowleft; \Theta^i)$; $H(S)$ is the local score of a set S of data samples (L or R), which is either the entropy or the Gini index [2]. Throughout this paper, they always use the entropy, which is defined as

$$H(S) = -\sum_{k=1}^{K} [p(k|S) \cdot \log(p(k|S))], \tag{4}$$

where K is the number of classes, and $p(k|S)$ is the probability for class k, estimated from the set S.

As reviewed above, training in RFs is performed via recursively splitting training data into child nodes, where each split locally tries to optimize Eq.3. The local score $H(S)$ can generally be definied as

$$H(S) = \sum_{k=1}^{K} [p(k|S) \cdot l_{mm}(p(k|S) - \frac{1}{K})], \qquad (5)$$

where $l_{mm}(\cdot)$ is a margin maximizing loss function. Hence, it is easy to see that minimizing the entropy is equivalent to minimizing the log likelihood, a typical Fisher-consistent loss. Thus, via greedily minimizing such a local score, a decision tree aims at minimizing a Fisher-consistent loss function, and is hence approximating Bayes optimal learners.

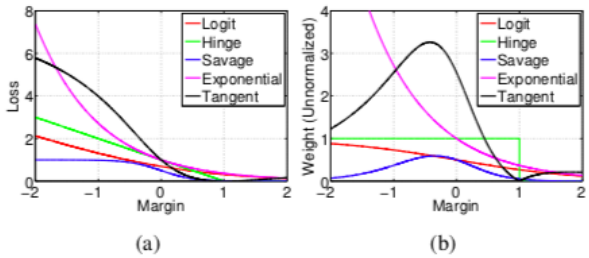

Figure 1. Loss functions (a) and their corresponding weight update functions (b), both with respect to the margin.

There exist several different Boosting variants, most of them differing by the loss funciton they use, which in turn determines the shape of the weight distribution. In Fig.1 they illustrate prominent loss functions and the deriatives of frequently used losses.

References

[1] Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9(7):1545–1588, 1997. 1

[2] G. Biau, L. Devroye, and G. Lugosi. Consistency of random forests and other averaging classifiers. *JMLR*, 9(9):2015–2033, 2008. 1

[3] R. Caruana, N. Karampatziakis, and A. Yessenalina. An empirical evaluation of supervised learning in high dimensions. In *ICML*, 2008. 1

[4] Y. Freund and L. Mason. The alternating decision tree learning algorithm. In *ICML*, 1999. 1