Boosting with control

January 13, 2011

1 Introduction

We introduce a general framework in which we can do multi-instance boosting, cascades and DAGs, attentional boosting, and sparse boosting. Alternating decision trees can also be formulated in the framework.

2 The setup

We consider classification as a sequential decision process. We have a set of nodes $\mathcal{J} = \{1, \ldots, M\}$. In each node $j \in \mathcal{J}$ there is a multi-class discriminant function $\mathbf{g}_j$ that, given an input observation, outputs a discriminant vector $\mathbf{g}_j(\mathbf{x})$ with the usual semantics. In each node there is also a controler $a_j : \mathcal{J} \to \mathcal{J}^*$ that selects the next set of nodes to visit. The full classifier at a given node j can be defined recursively as

$$\mathbf{f}_j(\cdot) = \begin{cases} \mathbf{0} & \text{if } j = j_{\text{terminal}}, \\ \mathbf{g}_j(\cdot) + \sum_{j' \in a_j} \mathbf{f}_{j'}(\cdot) & \text{otherwise,} \end{cases}$$

where j_{terminal} is a special node with discrimant function $\mathbf{0}$ and $a_{j_{\text{terminal}}} = \emptyset$.

The input of a_j can be a lot of things. Of course it can take the observation $\mathbf{x}$ and the output $\mathbf{g}_j(\mathbf{x})$. If $\mathbf{g}_j$ itself is a boosted classifier $\mathbf{g}_j(\cdot) = \sum_{t=1}^{T_j} \alpha_j^{(t)} \mathbf{h}_j^{(t)}(\cdot)$, a_j can take the whole output matrix $\alpha_j^{(t)} \mathbf{h}_j^{(t)}(\mathbf{x})$. It is also possible that formally there is only one controler a (beside $a_{j_{\text{terminal}}}$) in which case it would make sense to give the parameters of each $\mathbf{h}_j^{(t)}$ explicitly as an input to a.

3 Examples

3.1 Alternating decision trees

Alternating decision trees were originally defined for binary classification, but they can be easily extended to multiclass classification is we assume that $\mathbf{h}(\mathbf{x}) = \mathbf{v}\varphi(\mathbf{x})$ where $\mathbf{v} \in \{-1,1\}^K$ or $\mathbf{v} \in \mathbb{R}^K$, and $\varphi : \mathbb{R}^d \to \{-1,1\}$ is a scalar classifer ("cut"). Each node j has $2T_j$ children, that is, two children for every base classifier $\mathbf{h}_j^{(t)}$, $c_{j+}^{(t)}$ and $c_{j-}^{(1)}$. Then the formal control function is

$$a_j(\mathbf{x}, \varphi_j^{(1)}, \ldots, \varphi_j^{(T_j)}) = \left\{ c_{j\mathrm{sign}\left(\varphi_j^{(1)}(\mathbf{x})\right)}, \ldots, c_{j\mathrm{sign}\left(\varphi_j^{(T_j)}(\mathbf{x})\right)} \right\}.$$

Each cut $\varphi_j^{(t)}(\mathbf{x})$ participates in the discriminant function $\mathbf{h}_j$ but also defines a binary decision to follow. The advantage is, of course, that we have a formal boosting algorithm to learn an alternating decision tree, mainly because each time we add a base classifier $\mathbf{h}$ to any of the nodes, we know *exactly* what happens with the controler. On the other hand, this is their disadvantage, too: the controler is tightly coupled with the classifier, it is very complex (in the generalization sense, not computationally), so we will overfit (of course, this is to be verified experimentally). The idea would be to decouple the controler from the classifier, so we have more control over its design. The question is then, of course, how to learn the controler and the classifier.

3.2 Cascades

This is probably the simplest architecture. It is defined for binary node classifiers $\mathbf{g}(\mathbf{x}) = g(\mathbf{x}) : \mathbb{R}^d \to \{-1,1\}$. There is a chain of nodes, and the controler is simply

$$a_j(g(\mathbf{x})) = \begin{cases} \{j+1\} & \text{if } g(\mathbf{x}) = +1, \\ \{j_{\text{terminal}}\} & \text{otherwise.} \end{cases}$$

Simple it is, there is no known boosting algorithm to learn this structure.

3.3 Classification-controled DAGs

Cascades can be extended to allow both left and right moves. They can also be extended to multiclass node-classifiers. There main characteristics of classification-controled DAGs is that the controler only gets the output vector $\mathbf{g}(\mathbf{x})$ as input.

The simple arhitecture would be a tree, but there is no reason why the sequence of decisions on to different paths could not converge to the same node. On the other hand, cycles should obviously be avoided.

3.4 Attentional boosting

This setup is inspired by []. The main feature is that base classifiers are grouped together by a natural clustering in their parameter space. Each subset can look at a subset of the features. Each node in the decision process is assigned to one of the groups. The controler gets everything in the node plus the parameter of the node.

3.5 Multi-instance boosting

This is a setup when bags of input are classified positively if at least one of the elements is positive, negatively otherwise. The controler should formalize the process of "looking for" the object. It is a special case of attentional boosting in the sense that only at the end when the object is found should we make a classification.

3.6 Sparse boosting

Each node is one base classifier, coming from a pool. The goal is to reach a decision by looking at a fixed (usually small) number of classifiers by navigating in the pool. If used together with autoassociative boosting, it could be used to build a deep booster: for the next level, represent the input as a sparse binary (or use the α's) vector, and learn it at the next level. I already tried something like that, that's what ParasiteLearner is about in multiboost.

floatboost

3.7 Probabilistic decision trees

3.8 Martingale boosting

3.9 Decision lists

3.10 Peer-to-peer boosting

learning plus control plus communication

4 Ideas for learning

Alternate between boosting and controling. Sparse boosting is just one iteration (learn a large pool then learn a controler). Given the node classifiers (as in attentional boosting), we should be able to learn the control using MDPs. Given the controler, fix the data set at each node, and boost. We have to make sure that some error bound decreases in each iteration.

Multi-instance: boost one on the bag, find the best element in each bag, iterate. Of course, we assume some kind of structure among bag elements, as in images.