2014 22nd International Conference on Pattern Recognition

Budding Trees

Ozan İrsoy
Department of Computer Science
Cornell University
Ithaca, NY 14853-7501, USA
Email: oirsoy@cs.cornell.edu

Olcay Taner Yıldız
Department of Computer Engineering
Işık University
Şile, İstanbul 34980 Turkey
Email: olcaytaner@isikun.edu.tr

Ethem Alpaydın
Department of Computer Engineering
Boğaziçi University
Bebek, İstanbul 34342 Turkey
Email: alpaydin@boun.edu.tr

***Abstract*—We propose a new decision tree model, named the budding tree, where a node can be both a leaf and an internal decision node. Each bud node starts as a leaf node, can then grow children, but then later on, if necessary, its children can be pruned. This contrasts with traditional tree construction algorithms that only grows the tree during the training phase, and prunes it in a separate pruning phase. We use a soft tree architecture and show that the tree and its parameters can be trained using gradient-descent. Our experimental results on regression, binary classification, and multi-class classification data sets indicate that our newly proposed model has better performance than traditional trees in terms of accuracy while inducing trees of comparable size.**

I. Introduction

A decision tree is a hierarchical structure for supervised learning tasks, composed of internal decision nodes and terminal label nodes [1]–[3]. Given an input vector (including a bias term) $x = [1, x_1, ..., x_d]^T$, the response at node m has the following recursive definition:

$$y_m(\boldsymbol{x}) = \begin{cases} \rho_m & \text{if } m \text{ is leaf} \\ y_{ml}(\boldsymbol{x}) & \text{else if } g_m(\boldsymbol{x}) > 0 \\ y_{mr}(\boldsymbol{x}) & \text{else if } g_m(\boldsymbol{x}) \le 0 \end{cases} \qquad (1)$$

If m is a leaf node, for binary classification, $\rho_m \in [0, 1]$ is the probabilistic response denoting the probability that the instance is positive; for regression $\rho_m \in \mathbb{R}$ is the numeric response. If m is not a leaf but an internal node, depending on the outcome of the test $g_m(\boldsymbol{x})$, we take the left or right branch, y_{ml} and y_{mr} respectively, and continue recursively.

Frequently [2], $g_m(\boldsymbol{x})$ uses only one of the input attributes:

$$g_m(\boldsymbol{x}) = x_j + w_0$$

and this is called the *univariate tree*. The *multivariate tree* [4], [5] is a generalization where we define

$$g_m(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x}$$

hence defining arbitrary *oblique* splits. The univariate tree is a special case where $w_{mi} = 1$ for some $i \in \{1, ..., d\}$ and $w_{mj} = 0$ for all $j \notin \{0, i\}$, and as such defines a split that is orthogonal to the axis x_i. If we relax the linearity assumption on $g_m(\cdot)$, we have the *multivariate nonlinear tree*. If the above constraints on $g_m(\cdot)$ are dependent on the node m itself, then we have the *omnivariate tree* [6].

Regardless of the type of the decision node, learning a tree is a difficult problem [3]. Finding the smallest decision tree that can classify all the instances in a training set is NP-hard [7]. That is why, decision tree induction algorithms are greedy—they do not guarantee finding the smallest decision tree but they learn in reasonable time.

Basically, a decision tree induction algorithm is composed of two steps:

1) *Growing the tree:* Starting from the root, at each node, given the data reaching that node, we look for the best decision function $g_m(\boldsymbol{x})$ (univariate or multivariate) that splits the data into two. If this split leads to an improvement (for example, in terms of entropy), the split is accepted, the node becomes a decision node with two children and tree generation continues at the two children recursively. If the split does not lead to any improvement, the node is not split further and remains as a leaf and a proper probability or numeric value is stored in it.

2) *Pruning the tree:* Once the tree is grown to its total length, we check if pruning a subtree, that is, replacing it with a leaf, leads to improvement over a separate pruning set. We are basically checking if the tree is overfitting at this stage and if so, by replacing a subtree with a leaf we are getting rid of variance.

That is, tree induction is composed of a first phase of greedily adding subtrees and the second phase of greedily replacing subtrees with leaves.

Our proposal in this work is to have a tree architecture where each node, which we call a *bud node*, can be an internal node and a leaf *at the same time*. We allow splitting and pruning at the same phase in tree learning, instead of having two separate phases each allowing only one type of change. A bud starts as a leaf, and if necessary may grow into a decision node and get children, but always retains its value as a leaf, and at any stage later on may lose its children and become a leaf again.

As we will see later, training a budding tree is an incremental process where small changes are done at each step and in Section II, we discuss the soft decision tree that is better suited for such small adjustments and as such forms the basis of the budding tree. In Section III, we discuss the budding tree model and how it is learned. We give experimental results in Section IV where we compare the budding tree with various univariate and multivariate tree models on many regression, binary, and multi-class classification data sets and we conclude in Section V.