Pattern Recognition ■ (■■■■) ■■■–■■■

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Multivariate alternating decision trees

Hong Kuan Sok [a,*], Melanie Po-Leen Ooi [a,b], Ye Chow Kuang [a], Serge Demidenko [a,c]

[a] Advanced Engineering Platform & Electrical and Computer Systems Engineering, School of Engineering, Monash University, 47500 Bandar Sunway, Malaysia
[b] School of Engineering & Physical Sciences, Heriot-Watt University, 62200 Putrajaya, Malaysia
[c] School of Engineering & Advanced Technology, Massey University, Private Bag 1102904, Auckland 0745, New Zealand

ARTICLE INFO

Article history:
Received 5 May 2015
Received in revised form
7 July 2015
Accepted 17 August 2015

Keywords:
Alternating decision tree
Boosting
Multivariate decision tree
Lasso
LARS

ABSTRACT

Decision trees are comprehensible, but at the cost of a relatively lower prediction accuracy compared to other powerful black-box classifiers such as SVMs. Boosting has been a popular strategy to create an ensemble of decision trees to improve their classification performance, but at the expense of comprehensibility advantage. To this end, alternating decision tree (ADTree) has been proposed to allow boosting within a single decision tree to retain comprehension. However, existing ADTrees are univariate, which limits their applicability. This research proposes a novel algorithm – multivariate ADTree. It presents and discusses its different variations (Fisher's ADTree, Sparse ADTree, and Regularized Logistic ADTree) along with their empirical validation on a set of publicly available datasets. It is shown that multivariate ADTree has high prediction accuracy comparable to that of decision tree ensembles, while retaining good comprehension which is close to comprehension of individual univariate decision trees.

1. Introduction

Decision trees are among the most powerful and popular classifiers available. They are acyclic-directed graphical models that solve classification problems using symbolic representation, i.e., a graph of decision nodes that are connected via edges (Fig. 1(a)). As a result, they follow the flowchart-like human logic and reasoning, making them highly comprehensible. Decision trees model the domain problem as a set of decision rules. Such a model is transparent and is understandable to specialists in relevant application areas [1]. For example in [2], medical experts used the quantitative information obtained from the alternating decision tree model to gain a better understanding between disease phenotypes and affection status. The comprehensibility trait therefore, makes decision trees highly accessible to users outside just a machine learning community, and therefore they can be found in a wide range of applications such as business [3], manufacturing [4], computational biology [5], bioinformatics [6], etc.

It is often possible to further improve the classification accuracy of an individual decision tree by combining a number of decision trees to make majority-voted decisions [7]. There are two popular strategies to achieve this: bagging [8] and boosting [9]. Unfortunately, an ensemble of decision trees results in many variations in the symbolic representation which causes the overall classifier to be large, complex, and difficult to interpret. This negates the comprehensibility advantage of being a decision tree [10]. The issue with large and incomprehensible boosted decision trees led to the invention of the alternating decision tree (ADTree), which was designed to retain interpretability in the boosting paradigm [10]. Rather than building a decision tree at every boosting cycle, a much simpler decision stump is created.

Fig. 1(b) shows a graphical illustration of the ADTree. Similar to the decision tree in Fig. 1(a), ADTree is also an acyclic-directed graphical model. However, the symbolic meaning of each node and the manner in which they are connected are different. It does not use leaf nodes at the terminal nodes, or decision nodes as the internal nodes. Instead, many decision stumps (or one-level decision trees) are combined to obtain a special representation where each of the stumps consists of a decision node and two prediction nodes.

ADTree can be viewed as a loose generalization to standard decision trees, boosted decision trees, and boosted decision stumps [10] due to the following reasons. First, ADTree can be used as an alternative to represent any standard decision tree model with the same functionality. In addition to that, ADTree allows multiple decision stumps under the same prediction node to get majority-voted decisions. Boosting can be implemented directly within the same tree as opposed to the conventional way of creating boosted decision trees or boosted decision stumps. There are a number of extensions of ADTree such as multi-label ADTree [11], multi-class ADTree [12] and complex feature ADTree [13]. ADTree has been successfully implemented in various applications such as genetic disorders [14], corporate performance prediction [3], and bioinformatics [15].

Unfortunately, there are two major drawbacks of using univariate decision nodes in ADTree. First, as with any other univariate decision tree, the splitting based on a single feature is axis-parallel partitioning of the input space. This leads to a high bias and generates large decision

* Corresponding author. Tel.: +60 35 514 6238; fax: +60 35 514 6207.
E-mail address: sok.hong.kuan@monash.edu (H.K. Sok).

http://dx.doi.org/10.1016/j.patcog.2015.08.014
0031-3203/

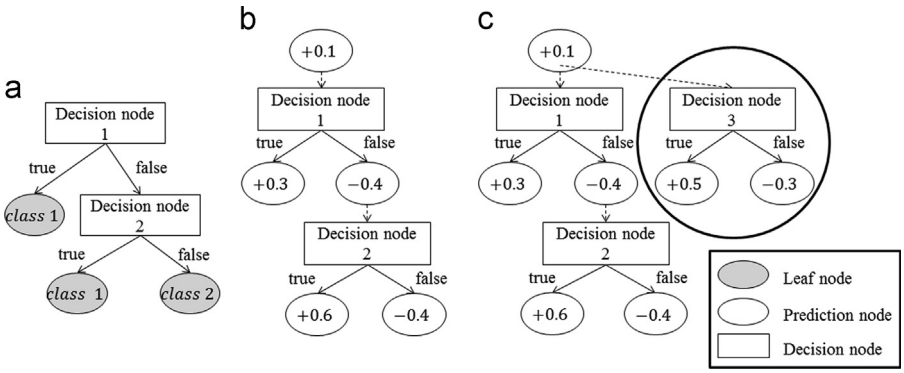

Fig. 1. Decision trees:(a) a classical decision tree consisting of decision nodes as internal nodes and leaf nodes as terminal nodes; (b) an alternating decision tree which can be used to represent the standard decision tree shown in part (a) to make the same prediction; and (c) the accommodation of boosting in the ADTree, whereby more decision stumps can be added to any existing prediction nodes (highlighted in circle) to obtain majority-voted decisions.

trees in classification problems that have co-dependent features. The resultant large and complex decision tree complicates the interpretation process. Second, ADTree induction is based on a *probably approximately correct* (PAC) learning framework which requires a weak learner to achieve an error rate ε that is slightly better than random guesses for binary class problems; formally $\varepsilon \leq 0.5 - \Psi$ for a small constant Ψ (known as *edge*). Unfortunately, simple univariate decision stumps sometimes do not satisfy the weak learning condition. This causes the boosting procedure to fail in generating a functioning ADTree model [16].

The aim of this paper is to present a novel multivariate alternating decision tree learning algorithm with boosting capability that offers improved classification performance of decision trees while remaining comprehensible. The goals are to:

1. Outperform the existing univariate ADTree and multivariate (unboosted) decision trees in terms of prediction accuracy while offering good comprehensibility;
2. Match the performance of univariate decision trees for univariate problems while outperforming them on multivariate datasets;
3. Provide superior comprehensibility compared to the ensemble-based decision trees.

There are several different subsections in the existing ADTree algorithm that can be restructured in order to induce a multivariate ADTree. In this paper, three possible variations are explored, namely *Fisher's ADTree*, *Sparse ADTree* [17], and *regularized Logistic ADTree*. The Sparse ADTree presented in the earlier paper was the first attempt to induce a multivariate ADTree. The current paper presents significantly new and further developed results that cover two additional multivariate alternating decision tree (ADTree) designs. This increases the material coverage and comprehension as well as applicability by practitioners and researchers in the field. In addition, the experiments are significantly more vigorous with extended discussions on the validity, usage and applicability of the multivariate alternating decision trees. All the three variants of multivariate ADTree were tested on a set of real-world datasets against a number of established decision tree learning algorithms such as the original univariate ADTree [10]; univariate decision trees: C4.5 [18] and CART [19]; the multivariate decision tree – Fisher's decision tree [20]; and ensemble of decision trees: Boosted C4.5 and oblique Random Forest [21]. Note that there are other variants of decision trees presented in the literature (e.g., [22,23]). However, the benchmarking algorithms are selected based on availability of the source codes, and they are used as representatives of the different decision tree families. This was done in order to compare the overall prediction accuracy, induction time, tree size and complexity/comprehensibility against different families of decision trees. For statistical verification and comparisons, the standard 10×10 fold stratified cross-validations were performed on all datasets to generate performance estimations.

The rest of this paper is as follows: Section 2 provides a brief literature review on supervised learning, boosting, and ADTree. The proposed multivariate ADTree algorithms are presented in Section 3. The experimental setup and obtained results are given in Section 4 together with the detailed discussions. Section 5 presents the conclusion and outlines the future work.

2. Background on alternating decision tree

2.1. Supervised learning framework

For better readability, the notations used in this paper are first described. Vectors are typed in bold (e.g., $\mathbf{x}$) and they are all column vectors unless specified otherwise. Scalars are typed in regular (e.g., λ). Matrices are given in capital bold (e.g., $\mathbf{X}$). Specific entries in vectors are indexed with a scalar. For example, the ith entry of a column vector $\mathbf{x}$ is denoted as $\mathbf{x}_i$. For matrices, the entry of ith row and jth column of a matrix $\mathbf{X}$ is denoted as $\mathbf{X}_{ij}$. The entire ith row of a matrix $\mathbf{X}$ is denoted as $\mathbf{X}_{i\bullet}$ and the entire jth column of a matrix $\mathbf{X}$ is denoted as $\mathbf{X}_{\bullet j}$.

Under supervised learning, a training dataset $[\mathbf{X}, \mathbf{y}]$ consists of a set of n labeled samples, where each sample $\mathbf{x} \in R^P$ is a real-valued column vector of p features and its corresponding label $y \in \{+1, -1\}$ assumes either positive or negative class for a binary classification problem. The dimension of the design matrix $\mathbf{X}$ is $n \times p$, and the column vector $\mathbf{y}$ is of the length n. The ith row of the design matrix $\mathbf{X}$ or $\mathbf{X}_{i\bullet}$ refers to the ith sample, a transposed vector, i.e., $\mathbf{x}^T$. The goal of a decision tree learning algorithm is to learn a single classification model. For ensemble learning, the weak learner is repeatedly called to learn multiple models.

2.2. Boosting

Boosting is an important development in the field of machine learning. It allows for any choice of a prevalent learning algorithm as long as the weak learning condition $\varepsilon \leq 0.5 - \Psi$ is satisfied for binary class problems. Paper [24] shows that decision trees are the popular choice as weak learners due to their inherent instability to small variations in training datasets. Boosting creates such variations through weight distribution over the training samples by sequential reweighting. This paper implements two different boosting algorithms to induce multivariate ADTree, namely, AdaBoost and LogitBoost (see Table 1).

AdaBoost initializes the weight distribution $\mathbf{w}$ as a uniform one with an initial weight value of $1/n$. The weight of the ith sample at the tth boosting procedure is indicated as $\mathbf{w}_i^{(t)}$. AdaBoost then repeats for T boosting procedures to obtain a weak model $f_t(\mathbf{x})$ from the weak learner, determines the linear coefficient of the weak model α_t