Pattern Recognition Letters 60–61 (2015) 57–64

Contents lists available at ScienceDirect

Pattern Recognition Letters

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

Sparse alternating decision tree[☆]

Hong Kuan Sok[*], Melanie Po-Leen Ooi, Ye Chow Kuang

Monash University Malaysia, Jalan Lagoon Selatan, Bandar Sunway 46150, Selangor D.E., Malaysia

ARTICLE INFO

Article history:
Received 20 May 2014
Available online 17 March 2015

Keywords:
Alternating decision tree
Decision tree
Boosting
Sparse discriminant analysis
Feature selection

ABSTRACT

Alternating decision tree (ADTree) is a special decision tree representation that brings interpretability to boosting, a well-established ensemble algorithm. This has found success in wide applications. However, existing variants of ADTree are implementing univariate decision nodes where potential interactions between features are ignored. To date, there has been no multivariate ADTree. We propose a sparse version of multivariate ADTree such that it remains comprehensible. The proposed sparse ADTree is empirically tested on UCI datasets as well as spectral datasets from the University of Eastern Finland (UEF). We show that sparse ADTree is competitive against both univariate decision trees (original ADTree, C4.5, and CART) and multivariate decision trees (Fisher's decision tree and a single multivariate decision tree from oblique Random Forest). It achieves the best average rank in terms of prediction accuracy, second in terms of decision tree size and faster induction time than existing ADTree. In addition, it performs especially well on datasets with correlated features such as UEF spectral datasets. Thus, the proposed sparse ADTree extends the applicability of ADTree to a wider variety of applications.

1. Introduction

Boosting is one of the most significant advances in machine learning research. Its underlying concept is to boost many "weak" base classifiers into an arbitrarily accurate classification model. Boosting initially weights every training sample equally. A weak learner is fitted to the training dataset based on this weight distribution. After each boosting cycle, the weights are updated such that correctly classified samples are weighted lower while incorrectly classified samples are weighted higher. The next boosting cycle uses the new weight distribution to train the weak classifier. A linear combination of these weak classifiers forms the output of the classifier.

Decision trees have been a popular choice of weak learner for boosting and have been shown to achieve good classification performance. Unfortunately, they are often large, complex and hard to interpret [16]. This also applies to bagged decision trees such as *Random Forest* [4]. The issue with decision tree ensemble has led to the invention of the *alternating decision tree* (ADTree) which brings interpretability to the boosting paradigm [16]. Rather than building a decision tree at each boosting cycle, a simple decision stump is used instead. This decision stump consists of a decision node and two prediction nodes. These one-level decision trees are then arranged in a special decision tree representation with subtle differences compared to classical decision trees such as C4.5 [30] and CART [5]. The ADTree has been successfully implemented in various applications such as genetic disorders [19], corporate performance prediction [8], management system [9], DNA microarray [32], automated trading [10], and modeling of disease trait information [25].

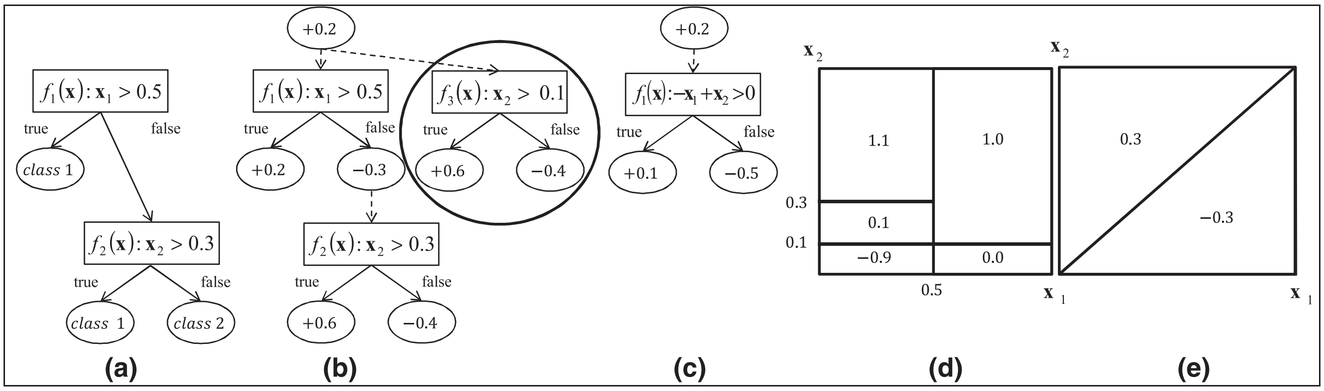
Existing variants of ADTree implement *univariate* decision stumps such that the decision node is split according to the value of a single feature [11,16,23,24]. This is effectively an axis-parallel partitioning method that divides a selected input feature into two disjoint ranges for discriminating purposes. In this way, the feature selection is performed implicitly as the "best" feature (axial direction) is selected. The use of univariate decision nodes simplifies its interpretation since important features are encoded within the alternating decision tree hierarchy.

Clearly, the choice of univariate decision nodes can be limiting in some cases, and may result in a large decision tree that complicates its comprehensibility [6]. This is because univariate decision nodes ignore potentially important interaction between features. By replacing the decision node to a multivariate variant, better accuracy and smaller tree size can be achieved [3]. However there is no reported literature on a multivariate variant of ADTree.

For multivariate decision nodes, the feature selection has to be incorporated explicitly. The number of features in a decision node is often a measure of decision node complexity and hence multivariate decision trees are often criticized for the loss of interpretability. This is a particularly challenging problem for the ADTree, which is made up of several weak classifiers. In order to induce a multivariate variant of ADTree that remains comprehensible, we propose to find a *sparse* set

[☆] This paper has been recommended for acceptance by F. Tortorella.
[*] Corresponding author. Tel.: +6 035 514 6238; fax: +6 035 514 6207.
E-mail address: sok.hong.kuan@monash.edu, sok1002@hotmail.com (H.K. Sok).

http://dx.doi.org/10.1016/j.patrec.2015.03.002
0167-8655/© 2015 Elsevier B.V. All rights reserved.



$f_1(\mathbf{x}): \mathbf{x}_1 > 0.5$
true
false
class 1
$f_2(\mathbf{x}): \mathbf{x}_2 > 0.3$
class 1
class 2
(a)
+0.2
$f_3(\mathbf{x}): \mathbf{x}_2 > 0.1$
+0.2
−0.3
+0.6
−0.4
(b)
$f_1(\mathbf{x}): -\mathbf{x}_1 + \mathbf{x}_2 > 0$
+0.1
−0.5
(c)
$\mathbf{x}_2$
1.1
1.0
0.1
−0.9
0.0
0.3
0.1
0.5
$\mathbf{x}_1$
(d)
0.3
−0.3
(e)

Fig. 1. (a) General decision tree (e.g. C4.5 and CART) (b) univariate ADTree, all dotted arrows under a prediction node are evaluated and (c) sparse ADTree with multivariate decision node. (d) Axis-parallel input space partitioning based on univariate ADTree in (b). (e) Input space partitioning based on sparse ADTree in (c). The values inside the partitions for both (d) and (e) indicate the summed prediction values.

of features in each decision node to discriminate between different classes. We achieve this by applying *sparse linear discriminant analysis* (SLDA) [7]. However, the sparse discriminant analysis cannot be applied directly under boosting paradigm because it is unable to adapt to the reweighting of the training dataset. Therefore, we propose a simple modification on sparse discriminant analysis to allow for this.

The proposed sparse ADTree algorithm is empirically tested on public datasets from the University of California, Irvine (UCI) Machine Learning Repository [15] as well as spectral datasets from the University of Eastern Finland (UEF) [34]. We benchmarked our algorithm against both univariate (existing ADTree, C4.5 and CART) and multivariate (Fisher's decision tree and a single multivariate oblique Random Forest) trees and SLDA. We show that the sparse ADTree achieves the best average rank in terms of prediction accuracy, second in terms of decision tree size and faster induction time than existing ADTree. We also show that it performs especially well on the dataset with highly correlated features such as spectral datasets from UEF.

The remainder of this paper is organized as follows. Section 2 provides a detailed background on general decision trees and ADTree. Section 3 describes the proposed sparse ADTree in full details including modifications to fit into boosting framework. Experimental results are presented with further analysis in Section 4 and this paper concludes in Section 5.

2. Background

In a typical supervised learning framework, training dataset D which consists of N labeled samples is used for learning purpose. The goal is to learn a classification model $F(\mathbf{x})$ that generalizes to new unseen samples. Each sample $\mathbf{x}$ is a p dimensional vector with label y of either −1 or +1 for binary class problem. We address only binary class problems to achieve clarity in presentation. Multiclass problems with K classes can be handled by the mapping introduced by Allwein et al. [1] and Dietterich [13].

2.1. Decision trees

Decision tree is a classifier with directed acyclic graph with nodes and edges. It starts with a root node at the top of the model (see Fig. 1(a)). Internal nodes are decision nodes and terminal nodes are leaf nodes. Each decision node implements a decision function $f(\mathbf{x})$ and every edge branching out from it corresponds to a decision outcome. Each leaf node stores a class label. Top-down recursive partitioning is a common approach to split the input space populated by the training dataset which results in a decision tree model.

Depending on decision function, decision trees are generally divided into univariate and multivariate variants. Univariate decision trees split on individual feature for each decision node while multivariate decision trees split on a feature vector (linear combination is a common approach). Classical univariate decision trees are ID3, C4.5 [30] and CART [5].

CART-LC (multivariate version of CART) was proposed as well in their pioneering work in 1984 to induce oblique decision tree where deterministic hill-climbing heuristic with backward feature elimination is implemented to learn the multivariate decision node of the form (1) parameterized by $\boldsymbol{\beta}$,

$$f(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\boldsymbol{\beta} > 0. \tag{1}$$

OC1 [29] was an improvement of CART-LC with two forms of randomization to avoid local optimality of hill-climbing and standard feature selection methods can be applied in an ad-hoc approach. In [13], LMDT [6] was proposed to estimate the parameters in an iterative approach to minimize the misclassification cost. Discriminant analysis is a popular analytic approach in recent years to optimize the parameters. These works include LDT [36] and Fisher's decision tree [26]. Our work follows this discriminant approach.

2.2. Alternating decision tree

ADTree is a generalization of classical decision trees. It consists of alternating layers of decision nodes and prediction nodes (see Fig. 1(b)). Prediction node contains real-valued prediction value. For instance, the decision tree in Fig. 1(a) can be represented as ADTree in Fig. 1(b) excluding the decision stump highlighted in circle. The input space partitioning due to Fig. 1(b) is shown in Fig. 1(d). If a given test sample $(\mathbf{x}_1, \mathbf{x}_2)$ is $(0.4, 0)$, decision tree in Fig. 1(a) will classify it as class 2 following the right-most path and its counterpart ADTree (without the highlighted decision stump) returns sign(+0.2–0.3–0.4 = –0.5) = –1 or (class 2) by summing all the traversed prediction values. The magnitude reflects the confidence margin on the prediction made and the sign makes the class prediction. The evaluation simply becomes sign (+0.2–0.3–0.4–0.4 = –0.9) = –1 if the highlighted decision stump is taken into account as well. A higher confidence is achieved in this case (a magnitude of 0.9 instead of 0.5).

Unique representation of ADTree allows multiple decision stumps under the same prediction node as illustrated in Fig. 1(b) where additional decision stump highlighted in circle can be added. This is how boosting is supported within specifically designed ADTree representation to improve prediction accuracy.

ADTree model can be described mathematically as a sum of decision rules as shown in (2). Each decision rule $r(\mathbf{x})$ consists of a *precondition*, *condition* and real-valued prediction values: α^+, α^- and