Pattern Recognition 60 (2016) 157–163

Contents lists available at ScienceDirect

Pattern Recognition

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

A hybrid classification algorithm by subspace partitioning through semi-supervised decision tree

Kyoungok Kim *

Information Technology Management Programme, International Fusion School, Seoul National University of Science & Technology (SeoulTech), 232 Gongreungno, Nowon-gu, Seoul 139-743, South Korea

ARTICLE INFO

Article history:
Received 7 December 2015
Received in revised form
21 April 2016
Accepted 25 April 2016
Available online 17 May 2016

Keywords:
Decision tree
Semi-supervised decision tree
Inhomogeneous measure
Subspace partitioning

ABSTRACT

Among data mining techniques, the decision tree is one of the more widely used methods for building classification models in the real world because of its simplicity and ease of interpretation. However, the method has some drawbacks, including instability, the nonsmooth nature of the decision boundary, and the possibility of overfitting. To overcome these problems, several works have utilized the relative advantages of other classifiers, such as logistic regression, support vector machine, and neural networks, in combination with a decision tree, in hybrid models which avoid the drawbacks of other models. Some hybrid models have used decision trees to quickly and efficiently partition the input space, and many studies have proved the effectiveness of the hybrid methods. However, there is room for further improvement by considering the topological properties of a dataset, because typical decision trees split nodes based only on the target variable. The proposed semi-supervised decision tree splits internal nodes by utilizing both labels and the structural characteristics of data for subspace partitioning, to improve the accuracy of classifiers applied to terminal nodes in the hybrid models. Experimental results confirm the superiority of the proposed algorithm and demonstrate the detailed characteristics of the algorithm.

1. Introduction

Classification is a data mining task that predicts categorical class labels. A classifier is a function that maps input samples consisting of several predictors to one of the class labels. Many different classification techniques have been developed and applied to several classification problems in various fields. Logistic regression [1,2], decision tree [3,4], neural networks [5] and support vector machine (SVM) [6,7] are the most popular methods to build a classification model.

A decision tree is simple to use, easy to understand, and offers various advantages compared with other methods. Thus, decision trees have been intensively studied and established as a useful technique to solve real-world problems. However, the decision tree algorithms that were suggested in the early stages had some disadvantages, such as being unstable and prone to overfitting. One approach to overcoming the weak points of decision trees was to combine the decision tree algorithm with other classification models [8–11]. In most cases, the decision tree algorithm was applied to the dataset first, and then other classification models were built based on samples at terminal nodes. In such cases the decision tree has the role of dividing a dataset into several subsets, and the combined classification models are used as classifiers for each subspace. These hybrid models have achieved better classification performance than single classification models.

Although combinations of decision trees and other models have been empirically proved to be effective for classification problems, the decision tree algorithm is not optimized for subspace partitioning. The decision tree creates child nodes by conducting a value test, which measures the ability of input features to estimate the dependent variable. As a result, subsets created by a decision tree often lack structural or topological homogeneity. Partitioning based on labels may improve the final classification accuracy of individual models at terminal nodes, because the ground assumption for classification is that observations with the same label are similar to each other in terms of explanatory variables.

However, the semi-supervised decision tree proposed in this paper is motivated by the likelihood that class distribution may be different depending on the regions where the samples are located. Unlike a traditional decision tree, the semi-supervised decision tree considers not only class labels but also distributions of input variables, which reflect and incorporate the structural or topological properties of the data. Split rules obtained by the proposed algorithm reflect the boundaries of the separate regions where input values are concentrated. This cutoff point is based on input

* Tel.: +82 29707286.
E-mail address: kyoungok.kim@seoultech.ac.kr

http://dx.doi.org/10.1016/j.patcog.2016.04.016
0031-3203/

features only, so the term "semi-supervised" is used to describe the proposed algorithm.

The rest of the paper is organized as follows. In Section 2, decision tree and hybrid decision tree models are described. In Section 3, the proposed semi-supervised decision tree algorithm is explained in detail. In Section 4, the experimental results obtained from the proposed algorithm and other comparative algorithms are presented. Finally, conclusion of the paper is given in Section 5.

2. Literature reviews

2.1. Decision tree

The decision tree is one of the data mining algorithms widely used for both classification and regression. Each interior node corresponds to one of the input variables and is split into child nodes based on the values of the input variable. Each leaf or terminal node represents the particular value of a target variable, for example, the specific class of a categorical variable for classification problem, and the certain real value of a continuous variable for regression problems.

During the classification tree learning process, samples at each interior node are split into subsets based on an attribute, and this process is repeated on each derived subset in a recursive manner called "recursive partitioning." The recursion is finished when a subset at a node has the same target value, when splitting does not improve prediction, or when splitting is impossible because of user-defined constraints.

At every step when growing a decision tree, one of the input variables is selected to split samples. Based on the chosen variable, the split point is determined by an attribute value test. Gini impurity and entropy are the most widely used tests for classification trees. Following two equations describe entropy [12] and Gini impurity [3], respectively:

$$H_e(S) = -\sum_{y \in C} p(y)\log_2 p(y) \tag{1}$$

$$H_g(S) = \sum_{y \in C} p(y)(1-p(y)) = 1 - \sum_{y \in C} p(y)^2 \tag{2}$$

where S represents the dataset, C is a set of classes, and $p(y)$ is the proportion of the number of samples with the class label, y in C. Both Gini impurity and entropy have 0 when there is the only one class in C and reach the maximum value when all classes are equally probable.

The split rule is determined by a reduction in entropy and Gini impurity after the split, which is called information gain:

$$G(r,S) = H(S) - \sum_{t} p(t)H(t) \tag{3}$$

where r is a certain split rule and t represent the child nodes induced by r on the set S at the current node. $p(t)$ is the proportion of the number of samples corresponding to t. The attribute and the value for split rule are determined to obtain the maximum information gain at the current node.

After the growth of the full tree, the output classes of the samples are determined at terminal nodes. Each terminal node decides on a target value based on the majority class at the terminal node. When a new sample is observed, it is classified as one of the terminal nodes of the tree depending on input variables. Then, the target is predicted to be the majority class at the leaf node.

There are several variations in decision tree algorithms, with different details. Iterative Dichotomiser 3 (ID3) is a proposed algorithm which uses entropy to calculate information gain [4]. This algorithm tests all of the unused attributes of a dataset at each iteration and then splits each interior node into two children. C4.5 is an extension of the ID3 algorithm that builds a tree in the same way as ID3, i.e., by using entropy [13,14]. The difference in C4.5 properties is that C4.5 can handle both continuous and categorical attributes and learn from a dataset with missing values. Moreover, C4.5 introduces a pruning step after tree growth to avoid overfitting. One of the widely used decision tree algorithms is classification and regression tree (CART) [3]. Unlike ID3 and C4.5, CART can create both classification and regression trees. The basic procedures used for both classification and regression have several similarities but have some differences, such as the method to determine where to split.

2.2. Characteristics of decision trees

The decision tree has advantages when solving classification or regression problems. First, the interpretation of results in a tree is very simple. In the test phase, the tree is useful for rapidly predicting new observations, and the result is easily interpreted using a few simple if–then statements. This property is powerful for addressing real business problems because the decision tree can offer some intuition about the relationships between input and output.

Second, the decision tree is nonparametric. During training, it creates logical if–then rules which do not require an implicit assumption about the underlying distributions or relationships of the input and output variables. Thus, the decision tree is appropriate for general data mining problems where minimal prior knowledge about the data is common.

Third, the decision tree is inherently nonlinear. It can use the same variable several times to create split rules during tree growth, revealing a nonlinear relationship between the variables. Although introducing nonlinearity often increases model complexity and reduces the ability of interpretation in other data mining techniques, the decision tree can implement nonlinearity in a simple way.

Finally, the decision tree can easily handle categorical and numerical variables in the same algorithm, unlike many other data mining algorithms, such as SVM, generalized linear or logistic regression models, and neural networks. In most algorithms, handling categorical variables is complicated.

2.3. Hybrid decision tree combining with other models

The decision tree has many positive points, but it also has limitations. The decision boundary obtained from the trained tree is not smooth, because the decision tree considers one variable at each node and splitting is performed at a certain point. In addition to this point, the probabilities of classes estimated from the decision tree are finite, and the score of the specific terminal node is shared by all cases in that node. Another flaw of the decision tree is instability, in that small changes in input features can lead to large changes in the final tree.

To overcome these drawbacks, hybrid models, which combine a decision tree with other classifiers, have been proposed. One of the advantages of decision trees is that the decision tree algorithm can generate simple if–then statements to make subsets purer than the original dataset in terms of the target variable. Using generated rules, observations including train data and new data are assigned into one of the subsets, and this process can be viewed as subspace partitioning. Therefore, the decision tree acts as a preprocessing step to divide a dataset into small subsets.

In [15], CART was combined with logistic regression to analyze motor vehicle injury data. The shallow decision tree was produced by CART, and logistic regression models were trained on each