Implementing a Binary Decision Tree with Categorical and Real Valued Attributes in C++ Programming Language

Alireza Zaeemzadeh
CAP 6676 - Machine Problem 1
University of Central Florida
Email: zaeemzadeh@knights.ucf.edu

Fall 2015

In this machine problem, a binary decision tree is implemented in C++ programming language. The dataset includes both categorical and real valued attributes. A dataset is chosen from the University of California Irvine repository. This repository is available online at https://archive.ics.uci.edu/ml/datasets.html.

For this report, the dataset **Abalone** is employed to train and test the decision tree. The default task for this dataset is multi-class classification. The goal is to estimate the age of the abalone by predicting the number of the rings in the shell. There exist 29 classes, representing different number of rings. To reduce the problem into a binary decision problem, data points from two classes is employed to train and test the decision tree.

The candidate hypotheses should be chosen in such a way that produce enough number of data points. Also, it is worthwhile to mention that if two classes with similar attributes are chosen for the classification task, the performance of the tree would decrease significantly.

To validate the performance of the tree, K-fold cross validation is used. In each fold, N/K data points are employed to test the data and the rest of data points are used to train the data. N is the total number of data points and the default value of K is set to 10.

In the algorithm, each node receives a vector of data points and splits it among the children node. This procedure is repeated until the stopping conditions are met. A node is considered as a leaf node if one of the following conditions is met:

- If **entropy** of the data points in each node is less than 0.2.
- If **number of data points** in the node is less than 10.
- If **depth** of the node in the tree is more than 8.

- If all of the **attributes** are employed in the parent nodes.

The default values for these conditions can be changed in the file "Node.h".

Moreover, each node is responsible to find the best attribute to use for the decision. The best attribute is the attribute that causes the largest *entropy gain*. Hence, each node chooses an attribute by a brute-force trial of all candidate attribute. Also, to avoid overfitting, the repeating usage of an attribute is avoided. Each node removes an attribute from the set of possible attributes before passing the set to its children.

To split the data by using a real valued attribute, a threshold value should be computed. The threshold value θ is computed by using the following closed form expression:

$$\begin{aligned} \theta &= \frac{-b}{w} \\ \text{where} \quad w &= \frac{\sigma_{xy}}{\sigma_x^2} \\ b &= \mu_y - w\mu_x. \end{aligned} \tag{1}$$

σ_{xy} is the covariance of the attribute and the labels, σ_x^2 is the variance of attribute values, and μ_x and μ_y are the mean of the attribute values and the labels, respectively. It is worthwhile to mention that after reading the data from the data set the labels are mapped to -1 and 1, to represent a binary decision problem.

Table 1 illustrates the results for different pairs of classes as the input of the algorithm.

Table 1: Average Performance Metrics for $K = 10$

Class Pair	**Accuracy mean/std**	**Precision mean/std**	**Recall mean/std**	**F_measure mean/std**
(4,15)	1/0	1/0	1/0	1/0
(5,14)	0.975/0.020	0.954/0.102	0.973/0.032	0.959/0.057
(7,12)	0.869/0.077	0.861/0.076	0.833/0.136	0.839/0.080
(8,11)	0.758/0.091	0.745/0.054	0.726/0.164	0.726/0.103
(9,10)	0.577/0.042	0.563/0.058	0.537/0.104	0.544/0.066

Table 2: Average Performance Metrics for $K = 10$

Class Pair	**Accuracy mean/std**	**Precision mean/std**	**Recall mean/std**	**F_measure mean/std**
(8,11)	0.729/0.071	0.717/0.478	0.683/0.120	0.695/0.072

To study the effect of overfitting, we can change the default values of stopping conditions and evaluate the results. For that, the maximum depth of the tree is increased to 40 and repeating usage of attributes is allowed. Table 2 shows the performance of the algorithm for the pair $(8, 11)$.

It is clear that the performance of the algorithm is degraded after increasing the depth of the tree. This example shows the effect of overfitting on the testing error.