Decision Tree Learning

Youyun Zhang

Introduction

In this assignment I implemented a decision tree learner and evaluate it on a mushroom dataset. The goal is to determine any new mushroom is either poisonous or not. The classification label in this dataset is binary but the attributes are multi-valued.

The dataset file mushroom.txt is a simple text file with complete data. The first line in the file contains the class name followed by the features names. In the rest of the data each line represents a sample mushroom with its class (either 0 or 1) followed by its features values. Each sample mushroom in the dataset has 22 discrete features such as cap shape, cap color, gill size, etc.

Building the Decision Tree

The decision tree is able to handle discrete multi-valued features. The optimal tree is a tree that is as consistent with the training data as it could be; and is as small as possible within the runtime constraint, thus we use heuristics. The algorithm uses a greedy divide and conquer strategy, which is testing on most important features prior than other features. Using Information Gain, the algorithm can decide the feature of each step that the tree splits on. Meanwhile we build the whole tree by recursively creating the subtrees, i.e., constructing the tree from the bottom.

80% of the data is used to build the decision tree (training data) and 20% (testing data) to evaluate its accuracy on both sets. The accuracy of a decision tree here is defined to be the percentage of sample mushrooms that the decision tree classifies correctly from that set.

Depth Bound

To avoid overfitting, the selection the decision tree's complexity is important. In this project I control the tree's complexity by setting a depth bound for the tree and evaluate.

To evaluate the accuracy of a bounded depth decision tree and choose the best depth bound, holdout cross-validation is chosen in this project. This resulting in splitting a validation set from the training data. 60% of the total data is used as training set, 20% as validation set.

The procedure of implementation is following: for each depth bound i between 1 and 15, build a decision tree with depth bound i using the training set and evaluate its accuracy on the validation set; denote the i that gives a maximum accuracy; then build a new decision tree with this depth bound using both the training and validation sets; finally, evaluate its accuracy using the rest of the data, i.e., testing set.

Executable

To run the files, simply run *make*, then run the new executable *dt*, with an argument of the data file *mushrooms.txt*.

The program first prints out the training and validation sets depth bounds 1,2,…,15. Then it decides the optimal depth bound by keeping track of the depth that gives maximum accuracy. Then a new decision tree is contracted using the optimal depth bound using both training and validation data, and print out its accuracy on the testing data.

Output

```
depth train%     valid%
1     89.1908     87.7768
2     90.254     89.0168
3     90.5493     89.194
4     90.697     88.8397
5     91.0514     87.6882
6     91.3763     85.3853
7     92.7939     80.5137
8     96.6037     80.868
9     99.1731     80.0709
10     99.9409     80.0709
11     99.9409     80.0709
12     99.9409     80.0709
13     99.9409     80.0709
14     99.9409     80.0709
15     99.9409     80.0709
max depth of training set is: 10 with an accuracy 99.9409
max depth of validation set is: 3 with an accuracy 89.194
accuracy of testing set using depth 3is 89.3711
```

Reference

USC CSCI 360 Artificial Intelligence class content.

Stuart J. Russell. Peter Norvig. *Artificial Intelligence: A modern Approach.* Third Edition. Chapter 18.