MSIS-DL 435 Session 2
Shaddy Abado
Data Warehousing & Data Mining
Winter 2013

Introduction

In this handout the basic concepts of classification and decision trees will be introduced. The parameters derived in this session to determine the decision trees' performances will also be applicable to measure the performances of other classification techniques which will be discussed in future sessions. This handout will be followed by a discussion of these principles to enhance the student's understanding of the learning objectives.

Reading for this session:

Chapter 4 of the textbook:

Tan, P., Steinbach, M., & Kumar, V. (2005). *Introduction to data mining*. Reading, MA: Addison-Wesley.

Classification:

As we discussed in the previous session, classification data mining techniques are systematic approaches which can be used to divide objects to one of a number of mutually exhaustive categories (classes). This means that each object must be assigned *only* to one class.

A number of classification techniques are available to describe or predict the class label of unknown records, such as: decision tree-based methods, neural networks, and Naïve Bayes-based methods. In this session, decision tree-based methods will be discussed, while neural networks and Naïve Bayes-based methods will be discussed in the next session.

Decision Trees:

Given a *training set* where the classes of its records are all known, a *decision tree* employs a learning algorithm to construct a descriptive classification model which uses a hierarchical structure to classify the given set of attributes. Decision trees consist of three types of nodes: *Root node*, *internal nodes*, and *leaf nodes*. The first two types of nodes contain attribute test conditions to classify the records, while the third type of node, leaf nodes, is assigned a class label. Decision trees are generated by repeatedly splitting the values of attributes at each node. There are various measures that can be used to determine the best way to split the records at the root node and internal nodes. Here, we will emphasize two of these indexes which measure the degree of nodes' impurity, the *Gini index* and the *Entropy index*. Denoting the fraction of records belonging to class i at a given node t as $p(i|t)$ then, the Gini index is defined as,

Gini index

$$Gini(t) = 1 - \sum_{i=1}^{N} p(i|t)^2$$

While the Entropy index is defined as,

Entropy index

$$Entropy(t) = -\sum_{i=1}^{N} p(i|t) \log_2 p(i|t)$$

Once the decision tree is constructed, its accuracy is determined based on data records whose classes are assumed to be unknown. Such a data record is called a *test set*, and it is used to evaluate the performance of the decision tree.

Some of the advantages of decision tree-based methods are that they can be easily interpreted for small-sized trees, inexpensive to construct, and they can classify unknown records extremely fast. However, when constructing a decision tree, one should be cautious not to over fit the model. *Overfitting* occurs when the model fits the training set too well. This leads to a larger decision tree that has a poor performance when predicting the test set records. Some of the causes for overfitting are lack of representative samples and presence of noise. Overfitting is usually overcome by *pruning* the decision tree. Pruning algorithms attempt to improve the decision tree accuracy by trimming the branches of the initial tree to improve its generalization capability, remove tree branches which reflect noise in the data, and reduce the decision tree's size.

An example of a training set, a test set, and a decision tree for the problem of classifying whether a tax payer will cheat in his tax return is shown in Table 1 and Figure 1. The three types of decision tree nodes are donated in Figure 1. It can be noticed that once the decision tree has been reconstructed, classifying a test record is straightforward. In Figure 1, the classification of the first test record is shown. Starting from the root node, the test conditions of the test record are applied to the decision tree (see bold red arrows). Here, it can be noticed that the decision tree classified the first two test set records correctly while the last two test set records are incorrectly classified.

Training Set

Refund	Marital Status	Taxable Income, $	Cheat
Yes	Single	125K	No
No	Married	100K	No
No	Single	70K	No
Yes	Married	120K	No
No	Divorced	95K	Yes
No	Married	60K	No
Yes	Divorced	220K	No
No	Single	85K	Yes
No	Married	75K	No
No	Single	90K	Yes

Test Set

Refund	Marital Status	Taxable Income, $	Cheat
No	Married	80K	No
Yes	Divorced	55K	No
No	Divorced	170K	No
No	Single	60K	Yes

Table 1

Refund	Marital Status	Taxable Income, $	Cheat
No	Married	80K	?

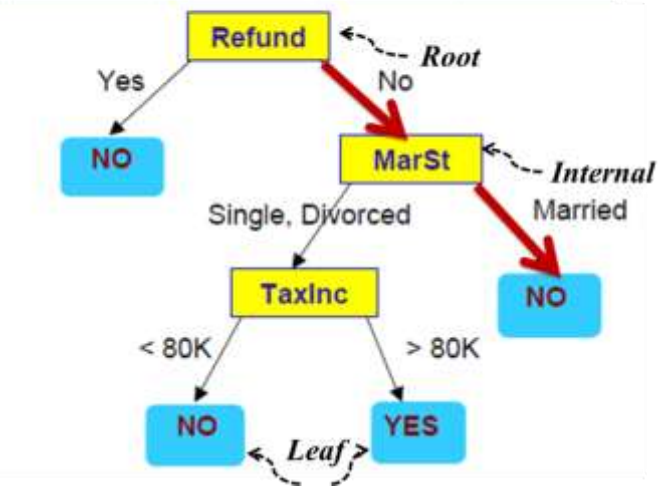

Figure 1

Measuring the Performance of Decision Trees

Once the decision tree is constructed, it is necessary to measure its performance. The decision tree's performance can be measured based on the counts of test records correctly and incorrectly predicted by the decision tree. These counts are presented in the *confusion matrix.* Each element in the matrix denotes the number of records correctly or incorrectly classified for each possible classification and class. The rows correspond to the correct classifications while the columns correspond to the predicted classification, as shown in Table 2.

Confusion matrix	**Predicted Class**		
True Class		**Class = C_0**	**Class = C_1**
	Class = C_0	**True Positive (TP)** (Number of correctly classified cases that belong to C_0)	**False Negative (FN)** (Number of cases incorrectly classified as C_1 that belong to C_0)
	Class = C_1	**False Positive (FP)** (Number of cases incorrectly classified as C_0 that belong to C_1)	**True Negative (TN)** (Number of correctly classified cases that belong to C_1)

Table 2

Based on the confusion matrix, two *performance metrics* can be defined to evaluate the performance of the classifier model: *accuracy*, which is defined as the proportion of instances that are correctly classified, and *error rate*, which is defined as the proportion of instances that are incorrectly classified. These two performance metrics are calculated as following:

Accuracy:

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

Error Rate:

$$Error\ Rate = \frac{FP + FN}{TP + FN + FP + TN}$$

To improve the performance of the decision tree, we would like to increase the accuracy value while decreasing the error rate value. However, the values of the accuracy and error rate parameters depend on the number of positive ($TP + FN$) and negative ($FP + TN$) instances. To overcome this limitation, two additional performance metrics can be defined, *True Positive Rate* (TPR) and *False Positive Rate* (FPR), and are calculated as follows:

True Positive Rate (TPR)

$$TPR = \frac{TP}{TP + FN}$$

False Positive Rate (FPR)

$$FPR = \frac{FP}{TN + FP}$$

For the same test set, we can inspect and determine the performance of different classifiers and decision trees by using the *Receiver Operating Characteristic* (ROC) graph. The performance of each classifier is represented as a single point on the ROC graph by plotting its TPR value versus the FPR value, see Figure 2. The diagonal line in Figure 2 which joins the bottom left and top right-hand corners corresponds to random guessing classifiers. Classifiers whose ROC graph points fall below this line are considered to be worse than a random guessing classifier. Consequently, for a good classifier, the point on the ROC graph is likely to be around the top left-hand corner. As it can be noticed from Figure 2, four unique cases of classifiers can be defined based on the ROC graph.

1. Perfect Classifier: Every instance is correctly classified ($TPR = 1, FPR = 0$).
2. Worse Possible Classifier: Every instance is wrongly classified ($TPR = 0, FPR = 1$).
3. The Ultra-Liberal Classifier: The classifier always predicts the positive class ($TPR = 1, FPR = 1$).
4. The Ultra-Conservative Classifier: The classifier always predicts the negative class ($TPR = 0, FPR = 0$).

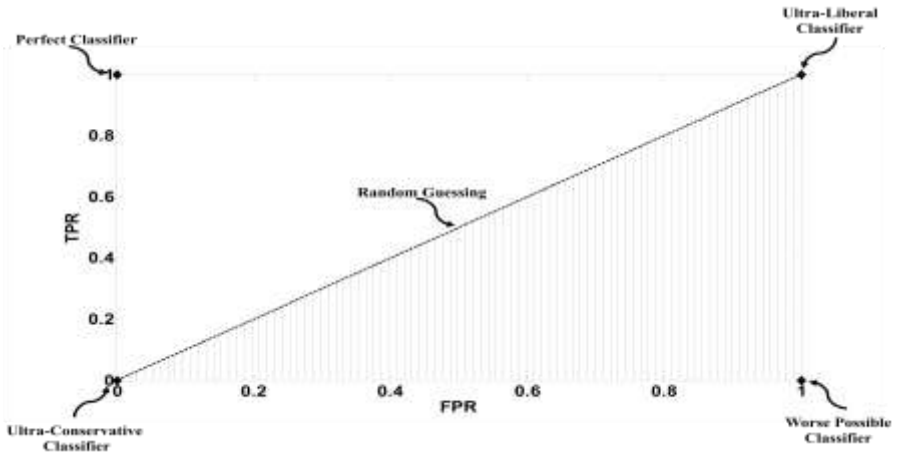

Figure 2

Weka

Throughout the course we will use the Java-based software Weka to implement various data mining tasks to different data sets. Weka is an open source software and can be downloaded from Weka's home page (http://www.cs.waikato.ac.nz/ml/weka/). A brief tutorial for Weka is available on Blackboard.

References and Further Readings

- Tan, P., Steinbach, M., & Kumar, V. (2005). "Introduction to data mining."
- Han, J., & Kamber M., (2006). "Data Mining: Concepts and Techniques."
- Bramer M. (2007). "Principles of Data Mining."