Data Mining - Lecture 3 Decision Tree Induction

Prof. Dr. Dewan Md. Farid

Professor, Department of Computer Science & Engineering, UIU

October 30, 2012

Classification

Classification is a data mining function that describes and distinguishes data classes or concepts. The goal of classification is to accurately predict class labels of instances whose attribute values are known, but class values are unknown. It is a form of data analysis that extracts models (called classifier) describing important data classes. It is a two-step process:

Learning step	(or training phase) where a classification model, classifier is constructed. A classification algorithm builds the classifier by analysing a **training dataset** made up of instances and their associated class labels.
Classification step	where the classification model is used to predict class labels for given data.

Classification Accuracy

- The classification accuracy of a classifier on a given test set is the percentage of test set instances that are correctly classified by the classifier.
- If the accuracy of the classifier is considered acceptable, the classifier can be used to classify future data instances for which the class label is not known.

Data Instance

- In the context of classification in data mining or machine learning, instances can be referred to as *data points, examples, tuples, samples*, or *objects*, which making up the training set are referred to as training instances and are randomly sampled from the database under analysis.
- Given a dataset, $D = \{x_1, x_2, \cdots, x_n\}$, each instance, x_i, is represented by an n-dimensional attribute vector, $x_i = \{x_{i1}, x_{i2}, \cdots, x_{in}\}$.
- The dataset, D, contains the following attributes $\{A_1, A_2, \cdots, A_n\}$.
- Each attribute, A_i, contains the following attribute values $\{A_{i1}, A_{i2}, \cdots, A_{ih}\}$, which represents a *feature* of x_i.
- The dataset, D, also belong to a set of classes $C = \{c_1, c_2, \cdots, c_m\}$.
- Each instance, x_i, is belong to a predefined class label, c_i.

Table: Commonly used symbols and terms.

Symbol	Term
D	Training Data
x_i	A data instance
X	A subset of instances
A_j	A feature
a_j^i	A feature's value
c_l	A class label
DT	A decision tree

Tree

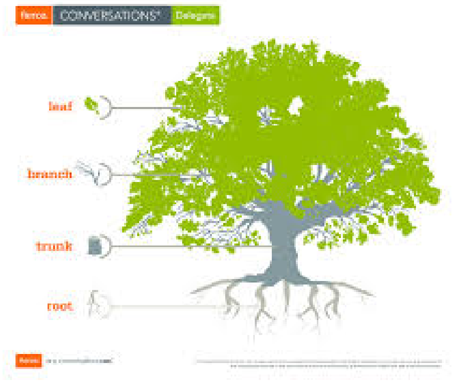

Figure: A picture of tree.

Decision Tree

Decision tree (DT) induction is the learning of decision trees from class-labeled training instances, which is a top-down recursive divide and conquer algorithm. The goal of DT is to create a model (classifier) that predicts the value of a target class for an unseen test instance based on several input instances. DTs have various advantages:

1. Simple to understand.
2. Easy to implement.
3. Requiring little prior knowledge.
4. Able to handle both numerical and categorical data.
5. Robust.
6. Dealing with large and noisy datasets.
7. Nonlinear relationships between features do not affect the tree performance.

Iterative Dichotomiser 3 (ID3)

The goal of DT is to iteratively partition the data into smaller subsets until all the subsets belong to a single class or the stopping criteria of DT building process are met.

- ID3 (Iterative Dichotomiser 3) algorithm that used information theory to select the best feature, A_j. The A_j with the maximum *Information Gain* is chosen as root node of the tree.
- To classify an instance, $x_i \in D$ the average amount of information needed to identify a class, c_l is shown in Eq. 1. Where p_i is the probability that x_i belongs to the class, c_l and is estimated by $|c_l, D|/|D|$.

$$Info(D) = -\sum_{i=1}^{N} p_i log_2(p_i) \tag{1}$$

ID3 (con.)

$Info_A(D)$ is the expected information required to correctly classify $x_i \in D$ based on the partitioning by A_j. Eq. 2 shows $Info_A(D)$ calculation, where $\frac{|D_j|}{|D|}$ acts as the weight of the jth partition.

$$Info_A(D) = \sum_{j=1}^{n} \frac{|D_j|}{|D|} \times Info(D_j) \tag{2}$$

Information gain is defined as the difference between $Info(D)$ and $Info_A(D)$ that is shown in Eq. 3.

$$Gain(A) = Info(D) - Info_A(D) \tag{3}$$

C4.5

Quinlan later presented C4.5 (a successor of ID3 algorithm) that became a benchmark in supervised learning algorithms. C4.5 uses an extension of *Information Gain*, which is known as *Gain Ratio*. It applies a kind of normalisation of *Information Gain* using *Split Information* defined analogously to $Info(D)$ as shown in Eq. 4.

$$SplitInfo_A(D) = -\sum_{j=1}^{n} \frac{|D_j|}{|D|} \times log_2\left(\frac{|D_j|}{|D|}\right) \tag{4}$$

The A_j with the maximum *Gain Ratio*, which is defined in Eq. 5 is selected as the splitting feature.

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo(A)} \tag{5}$$

Gini Index

The *Gini Index* is used in Classification and Regression Trees (CART) algorithm, which generates the binary classification tree for decision making. It measures the impurity of D, a data partition or X, as shown in Eq. 6, where, p_i is the probability that $x_i \in D$ belongs to class, c_l and is estimated by $|c_l, D|/|D|$. The sum is computed over M classes.

$$Gini(D) = 1 - \sum_{i=1}^{N} p_i^2 \tag{6}$$

It considers a binary split, a weighted sum of the impurity of each resulting partition. For example, if a binary split on A partitions D into D_1 and D_2 the *Gini Index* of D given that partitioning is shown in Eq. 7.

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2) \tag{7}$$

Gini Index (con.)

For each A_j, each of the possible binary splits is considered. The A_j that maximises the reduction in impurity is selected as the splitting feature, shown in Eq. 8.

$$\Delta Gini(A) = Gini(D) - Gini_A(D) \tag{8}$$

The time and space complexity of a tree depend on the size of the data set, number of features and the size of the generated tree. The key disadvantage of DTs is that without proper pruning (or limiting tree growth), trees tend to overfit the training data.

Algorithm 1 Decision Tree Induction

Input: $D = \{x_1, \cdots, x_i, \cdots, x_N\}$
Output: A decision tree, DT.
Method:

```
1:  $DT = \emptyset$;
2:  find the root node with best splitting, $A_j \in D$;
3:  $DT$ = create the root node;
4:  $DT$ = add arc to root node for each split predicate and label;
5:  for each arc do
6:      $D_j$ created by applying splitting predicate to $D$;
7:      if stopping point reached for this path then
8:          $DT'$ = create a leaf node and label it with $c_l$;
9:      else
10:         $DT'$ = DTBuild($D_j$);
11:     end if
12:     $DT$ = add $DT'$ to arc;
13: end for
```

Tree using ID3 - An Illustrative Example

- To illustrate the operation of DT, we consider a small dataset in Table 2 described by four attributes namely Outlook, Temperature, Humidity, and Wind, which represent the weather condition of a particular day.
- Each attribute has several unique attribute values.
- The Play column in Table 2 represents the class category of each instance. It indicates whether a particular weather condition is suitable or not for playing tennis.

Table: The playing tennis dataset

Day	Outlook	Temperature	Humidity	Wind	Play
D_1	Sunny	Hot	High	Weak	No
D_2	Sunny	Hot	High	Strong	No
D_3	Overcast	Hot	High	Weak	Yes
D_4	Rain	Mild	High	Weak	Yes
D_5	Rain	Cool	Normal	Weak	Yes
D_6	Rain	Cool	Normal	Strong	No
D_7	Overcast	Cool	Normal	Strong	Yes
D_8	Sunny	Mild	High	Weak	No
D_9	Sunny	Cool	Normal	Weak	Yes
D_{10}	Rain	Mild	Normal	Weak	Yes
D_{11}	Sunny	Mild	Normal	Strong	Yes
D_{12}	Overcast	Mild	High	Strong	Yes
D_{13}	Overcast	Hot	Normal	Weak	Yes
D_{14}	Rain	Mild	High	Strong	No

Gain Calculation

$$Info(D) = -\frac{9}{14}log_2\left(\frac{9}{14}\right) - \frac{5}{14}log_2\left(\frac{5}{14}\right) = 0.940$$

$$Info_{Outlook}(D) = \frac{5}{14} * \left(-\frac{2}{5}log_2\frac{2}{5} - \frac{3}{5}log_2\frac{3}{5}\right) + \frac{4}{14} * \left(-\frac{4}{4}log_2\frac{4}{4}\right)$$
$$+\frac{5}{14} * \left(-\frac{3}{5}log_2\frac{3}{5} - \frac{2}{5}log_2\frac{2}{5}\right) = 0.694$$

Therefore, the gain in information from such a partitioning would be:

$$Gain(Outlook) = Info(D) - Info_{Outlook}(D) = 0.940 - 0.694 = 0.246$$

Gain Calculation (con.)

$$Info_{Temperature}(D) = \frac{4}{14} * \left(-\frac{2}{4} log_2 \frac{2}{4} - \frac{2}{4} log_2 \frac{2}{4}\right)$$
$$+\frac{6}{14} * \left(-\frac{4}{6} log_2 \frac{4}{6} - \frac{2}{6} log_2 \frac{2}{6}\right)$$
$$+\frac{4}{14} * \left(-\frac{3}{4} log_2 \frac{3}{4} - \frac{1}{4} log_2 \frac{1}{4}\right) = 0.854$$

Therefore, the gain in information from such a partitioning would be:

$$Gain(Temperature) = Info(D) - Info_{Temperature}(D) = 0.940 - 0.854 = 0.086$$

So, Information Gain of **Outlook = 0.246**, **Temperature = 0.086**, **Humidity = 0.154**, and **Wind = 0.197**.

Table: The playing tennis sub-dataset, Outlook=Sunny

Day	Temperature	Humidity	Wind	Play
D_1	Hot	High	Weak	No
D_2	Hot	High	Strong	No
D_8	Mild	High	Weak	No
D_9	Cool	Normal	Weak	Yes
D_{11}	Mild	Normal	Strong	Yes

Table: The playing tennis sub-dataset, Outlook=Overcast

Day	Temperature	Humidity	Wind	Play
D_3	Hot	High	Weak	Yes
D_7	Cool	Normal	Strong	Yes
D_{12}	Mild	High	Strong	Yes
D_{13}	Hot	Normal	Weak	Yes

Table: The playing tennis dataset, Outlook=Rain

Day	**Temperature**	**Humidity**	**Wind**	**Play**
D_4	Mild	High	Weak	Yes
D_5	Cool	Normal	Weak	Yes
D_6	Cool	Normal	Strong	No
D_{10}	Mild	Normal	Weak	Yes
D_{14}	Mild	High	Strong	No

Decision Tree

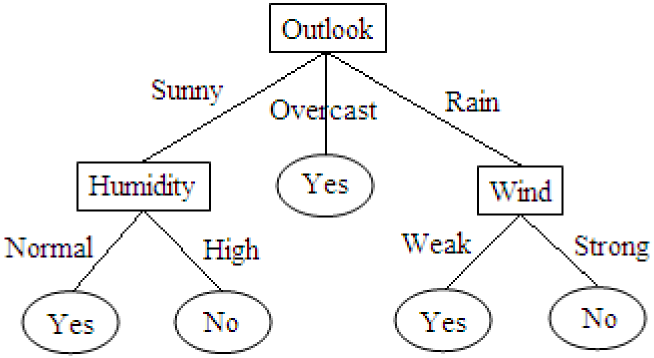

Figure: A DT generated by the playing tennis dataset.

Tree Pruning

- Tree pruning methods address the problem of overfitting the data.
- When a decision tree is built, many of the branches will reflect anomalies in the training data due to noise.
- There are two common approaches to tree pruning:
    1. Pre-pruning
    2. Post-pruning

Pre-pruning

- Pre-pruning a tree is pruned by halting its construction early (e.g. by deciding not to further split or partition the subset of training instances at a given node).
- **Upon halting, the node becomes a leaf**.
- The leaf may hold the most frequent class among the subset instances or the probability distribution of those instances.
- In choosing an appropriate threshold, high thresholds could result in oversimplified trees, whereas low thresholds could result in very little simplification.

Post-pruning

- Post-pruning is commonly used tree pruning approach, which removes subtrees from a full grown tree.
- A subtree at a given node is pruned by removing its branches and replacing it with a leaf.
- The leaf is labeled with the most frequent class among the subtree being replaced.
- Post-pruning requires more computation than pre-pruning, yet generally leads to a more reliable tree.

Repetition &. Replication

- No single pruning method has been found to be superior over all others.

- Although some pruning methods do depend on the availability of additional data for pruning, this is usually not a concern when dealing with large databases.

- Decision trees can suffer from *repetition* and *replication*.

Repetition	occurs when an attribute is repeatedly tested along a given branch of the tree.
Replication	duplicates subtrees exist within the tree. These situations can impede the accuracy and comprehensibility of a decision tree.

Tree Pruning by Cost Complexity

- The cost complexity pruning algorithm used in CART (Classification and Regression Trees) is an example of the post-pruning approach.
- This approach considers the cost complexity of a tree to be a function of the number of leaves in the tree and the **error rate** of the tree (where the error rate is the percentage of instances misclassified by the tree).
- It starts from the bottom of the tree.
- For each internal node, N, it computes the cost complexity of the subtree at N, and the cost complexity of the subtree at N if it were to be pruned (i.e., replaced by a leaf node).
- The two values are compared. If pruning the subtree at node N would result in a smaller cost complexity, then the subtree is pruned. Otherwise, it is kept.

Pruning Set

- A pruning set of class-labeled instances is used to estimate cost complexity.
- This set is independent of the training set used to build the unpruned tree and of any test set used for accuracy estimation.
- The algorithm generates a set of progressively pruned trees.
- In general, the smallest decision tree that minimises the cost complexity is preferred.

Pessimistic Tree Pruning

- Pessimistic tree pruning is used by C4.5 decision tree induction algorithm, which is similar to the cost complexity method in that it also uses error rate estimates to make decisions regarding subtree pruning.

- Pessimistic pruning does not require the use of a prune set. Instead, it uses the training set to estimate error rates. Recall that an estimate of accuracy or error based on the training set is overly optimistic and, therefore, strongly biased.

- The pessimistic pruning method therefore adjusts the error rates obtained from the training set by adding a penalty, so as to counter the bias incurred.

Scalability & Decision Tree Induction

- In data mining applications, very large training sets of millions of instances are common.
- Most often, the training data will not fit in memory.
- The efficiency of existing decision tree algorithms, such as ID3, C4.5, and CART, has been well established for relatively small data sets.
- Therefore, decision tree construction becomes inefficient due to swapping of the training instances in and out of main and cache memories.
- More scalable approaches, capable of handling training data that are too large to fit in memory, are required.

RainForest Tree

RainForest adapts to the amount of main memory available and applies to any decision tree induction algorithm. The method maintains an **AVC-set** (Attribute-Value, Class-label) for each attribute, at each tree node, describing the training instances at the node. The AVC-set of an attribute A at node N gives the class label counts for each value of A for the instances at N. Table 6 shows the AVC-set of outlook attribute of Table 2.

Table: AVC-set of Outlook attribute of Table 2

Outlook	**Yes**	**No**
Sunny	2	3
Overcast	4	0
Rain	3	2

Bootstrapped Optimistic Algorithm for Tree construction (BOAT)

- It creates several smaller subsets of the given training data, each of which fits in memory. Each subset is used to construct a tree, resulting in several trees.
- The trees are examined and used to construct a new tree, T', that turns out to be very close to the tree that would have been generated if all the original training data had fit in memory.
- BOAT can use any attribute selection measure that selects binary splits and that is based on the notion of purity of partitions such as the Gini index.

BOAT (con.)

- BOAT was found to be two to three times faster than RainForest, while constructing exactly the same tree.
- An additional advantage of BOAT is that it can be used for incremental updates. That is, BOAT can take new insertions and deletions for the training data and update the decision tree to reflect these changes, without having to reconstruct the tree from scratch.

*** THANK YOU ***