Machine Learning - Lecture Tree Pruning

Prof. Dr. Dewan Md. Farid

Professor of Computer Science
United International University

June 13, 2023

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 1 shows the AVC-set of outlook attribute of Table **??**.

Table: AVC-set of Outlook attribute of Table **??**

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.

Evaluating Classifier Performance

The performance measures of a classifier that how accurate the classifier predicting the class label of instances (both training and testing instances). There are four terms we need to know that are used in computing many evaluation measures.

Table: Class Prediction - Comparing classification performance to information retrieval.

True Positive, **TP**	False Negative, **FN**
False Positive, **FP**	True Negative,**TN**

Evaluating Classifier Performance (con.)

True positives, TP: These refer to the positive instances (e.g., $Play = Yes$) that were correctly classified/ labeled by the classifier. Let TP be the number of true positives.

True negatives, TN: These are the negative instances (e.g., $Play = No$) that were correctly classified/ labeled by the classifier. Let TN be the number of true negatives.

False positives, FP: These are the negative instances that were misclassified/ incorrectly labeled as positive by the classifier (e.g., instances of $Play = No$ for which the classifier predicted $Play = Yes$). Let FP be the number of false positives.

False negatives, FN: These are the positive instances that were misclassified/ incorrectly labeled as negative by the classifier (e.g., instances of $Play = Yes$ for which the classifier predicted $Play = No$). Let FN be the number of false negatives.

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. In the pattern recognition literature, this is also referred to as the overall **recognition rate** of the classifier, that is, it reflects how well the classifier recognises instances of the various classes. The classification accuracy is measured either by Equation 1 to Equation 2.

$$accuracy = \frac{TP + TN}{P + N} \tag{1}$$

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

$$accuracy = \frac{\sum_{i=1}^{|X|} assess(x_i)}{|X|}, x_i \in X \tag{3}$$

Error rate

The **error rate** or **misclassification rate** of a classifier on a given test set is the percentage of test set instances that are misclassified by the classifier, which is also known as the **resubstitution error**. The error rate is shown in Equation 4.

$$errorRate = \frac{FP + FN}{P + N} \tag{4}$$

Sensitivity

The **sensitivity** is referred to as the *true positive* (recognition) rate (i.e., the proportion of positive instances that are correctly identified), shown in Equation 5.

$$sensitivity = \frac{TP}{P} \tag{5}$$

Specificity

The **specificity** is referred to as the *true negative* rate (i.e., the proportion of negative instances that are correctly identified), shown in Equation 6. A perfect classifier would be described as 100% sensitivity and 100% specificity.

$$specificity = \frac{TN}{N} \tag{6}$$

Precision & Recall

The *precision* and *recall* measures are also widely used in classification. **Precision** can be thought of as a measure of exactness (i.e., what percentage of instances labeled as positive are actually such), whereas **recall** is a measure of completeness (what percentage of positive instances are labeled as such). If recall seems familiar, that?s because it is the same as sensitivity (or the true positive rate).

$$precision = \frac{TP}{TP + FP} \tag{7}$$

$$recall = \frac{TP}{TP + FN} = \frac{TP}{P} \tag{8}$$

Table: Evaluation Measures - Classifier performance

Measure	**Formula**
Accuracy/ Recognition rate	$\frac{TP+TN}{P+N}$
Error rate/ Misclassification rate	$\frac{FP+FN}{P+N}$
Sensitivity/ True positive rate/ Recall	$\frac{TP}{P}$
Specificity/ True negative rate	$\frac{TN}{N}$
Precision	$\frac{TP}{TP+FP}$
$F/\ F_1/\ F-score$	$\frac{2*precision*recall}{precision+recall}$
F_β, where β is a non-negative real number	$\frac{(1+\beta)^2*precision*recall}{\beta^2*precision+recall}$

k-fold Cross-Validation

- In k-fold cross-validation, the initial data are randomly partitioned into k mutually exclusive subsets or "folds", $D_1, D_2, \cdots, D_k$, each of which has an approximately equal size.
- Training and testing are performed k times.
- In iteration i, the partition D_i is reserved as the test set, and the remaining partitions are collectively used to train the classifier.
- 10-fold cross validation breaks data into 10 sets of size N/10.
- It trains the classifier on 9 datasets and tests it using the remaining one dataset. This repeats 10 times and we take a mean accuracy rate.
- For classification, the accuracy estimate is the overall number of correct classifications from the k iterations, divided by the total number of instances in the initial dataset.

*** THANK YOU ***