Machine Learning, Spring 2019 — Chao Lan
EN 2105 — EN 4084A
MWF 9-9:50am — clan@uwyo.edu

Decision Tree

Decision tree is a tree that assigns an instance x to one branch of an internal node by thresholding one of its features until x reaches a leaf node.

Figure 1 shows an example of a decision tree. Instance x will be input through the root node, and assigned to the left branch if $x_{\cdot 1} \leq t_1$ and to the right branch otherwise. This process will be repeated whenever x arrives at a new internal node and will be terminated when x arrives at a leaf node. A leaf node will assign x to one class, e.g., leaf node R_4 may assign x to class 1 and leaf node R_1 may assign x to class 2. Different leaf nodes can assign x to the same class.

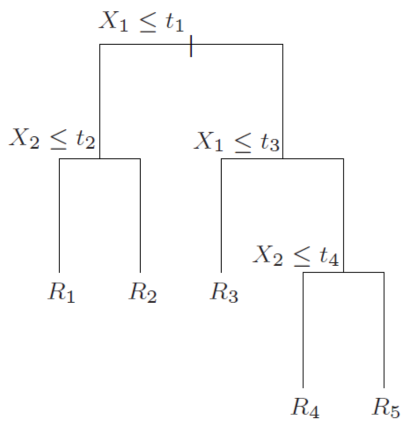

Figure 1: An Example of Decision Tree [ELS, Figure 9.2]

There are many algorithms to learn decision tree. We will introduce a classic one called <u>CART</u> (Classification And Regression Tree). It grows a tree by continuously splitting leaf nodes until some stopping criterion is met. It addresses the following three questions.

<u>Q1: Which class should a leaf node assign its arriving instances to?</u>

For a leaf node, if most arrived instances (training) are from class k, then it will classify future arriving instances (testing) to class k – this minimizes the probability of misclassification.

[*Discussion*] How does the above classification rule minimize misclassification probability?

Suppose a leaf node m assigns a set of instances S to class k. Let S_k be the subset of instances from class k and $S_{/k}$ be the subset not from class k. The classification error of m on S is

$$\begin{aligned} \text{er}_m(S) = \frac{1}{|S|}\sum\nolimits_{(x,y)\in S} 1_{f(x)\neq y} &= \frac{1}{|S|}\left(\sum\nolimits_{(x,y)\in S_{/k}} 1_{f(x)\neq y} + \sum\nolimits_{(x,y)\in m_k} 1_{f(x)\neq y}\right) \\ &= \frac{1}{|m|}\left(\sum\nolimits_{(x,y)\in S_{/k}} 1_{f(x)\neq y}\right) \\ &= \frac{|S_{/k}|}{|S|}. \end{aligned} \tag{1}$$

If $|S_k|$ is the largest among all possible k, then $|S_{/k}|$ is the smallest and $er_m(S)$ is minimized.

Q2: When to split a leaf node?

CART splits a leaf node if it is not pure. A node is pure if most of its arriving instances come from the same class – this can give smaller probability of misclassification.

[*Discussion*] Why a pure leaf node can give smaller misclassificaiton probability?

The purity of a node m is measured by its entropy, i.e.,

$$H(m) = -\sum\nolimits_{k=1}^{K} p_{mk} \cdot \log p_{mk}, \tag{2}$$

where p_{mk} is the probability that an instance from class k arrives at m. We can estimate p_{mk} from the training instances that arrive at m (through the constructed part of the tree).

Small $H(m)$ means node m is pure (and CART will treat m as a leaf node), and large $H(m)$ means node m is not pure (and CART will split it).

[*Discussion*] Give an example of the relation between entropy and purity.

[*Discussion*] When is entropy maximized?

Q3: Which feature to threshold when splitting a node?

CART splits a leaf node by thresholding one feature (which has not been thresholded before). It uses a cost function to measure the cost of each feature (e.g., how pure can the resulted child nodes be), and thresholds the feature with the largest/smallest cost. If a feature is continuous, it also examines the cost of a set of candidate thresholds.

After a tree is learned, we can prune it to avoid overfitting. Pruning is the task of cutting some branches of a tree (to reduce its size). It continuously cuts branches until classification accuracy is no longer improved. Another way to to avoid overfitting is to restrict the depth of the tree.