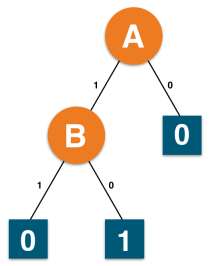
CIS 472/572, Winter 2016
Written Homework 1: Decision Trees

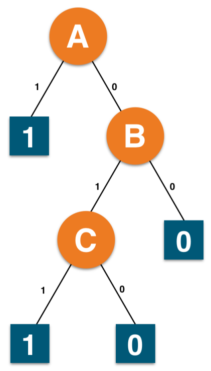
Wednesday, January 25th

Jacob Lambert, jlambert@cs.uoregon.edu

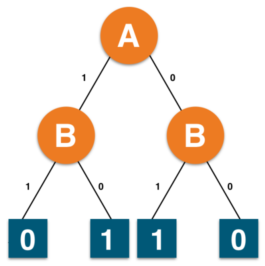
1. Give decision trees to represent the following boolean functions:



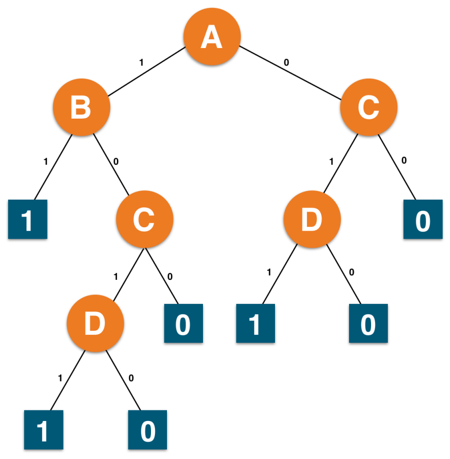
(a) $A \wedge \neg B$



(b) $A \vee [B \wedge C]$



(c) $A \oplus B$



(d) $[A \wedge B] \vee [C \wedge D]$

2. Consider the samples in the Play-tennis dataset from Table 3.2 in Mitchell's textbook (linked above). If you calculate the information-gain for all of the attributes of this set, you will observe that the attribute "Outlook" has the largest information-gain, which is equal to 0.246. Therefore, the attribute "Outlook" is the best heuristic choice for the root node.

   (a) List the labels of the new tree branches below the root node.
   Sunny, Overcast, Rain

   (b) Which partition of the data will be assigned to each branch by ID3? Please list the sample IDs that will be assigned to each branch.
   $S_{\text{sunny}} = \{D1, D2, D8, D9, D11\}$
   $S_{\text{overcast}} = \{D3, D7, D12, D13\}$
   $S_{\text{rain}} = \{D4, D5, D6, D10, D14\}$

   (c) Calculate the information gain for the remaining attributes in each branch, and determine which attribute will be chosen as the root of the sub-tree in each branch.

   $S_{\text{sunny}} = \{D1, D2, D8, D9, D11\}$

   $$Gain(S_{\text{sunny}},\ Humidity) = .970 - (3/5) * 0 - (2/5) * 0 = .970$$
   $$Gain(S_{\text{sunny}},\ Temperature) = .970 - (2/5) * 0 - (2/5) * 1 - (1/5) * 0 = .570$$
   $$Gain(S_{\text{sunny}},\ Wind) = .970 - (2/5) * 1 - (3/5) * .918 = .019$$

   $S_{\text{overcast}} = \{D1, D2, D8, D9, D11\}$

   $$Gain(S_{\text{overcast}},\ Humidity) = 0$$
   $$Gain(S_{\text{overcast}},\ Temperature) = 0$$
   $$Gain(S_{\text{overcast}},\ Wind) = 0$$

   $S_{\text{rain}} = \{D1, D2, D8, D9, D11\}$

   $$Gain(S_{\text{rain}},\ Humidity) = .970 - (3/5) * .918 - (2/5) * 0 = .419$$
   $$Gain(S_{\text{rain}},\ Temperature) = .970 - (2/5) * 1 - (3/5) * 0.918 - (0/5) * 0 = .0192$$
   $$Gain(S_{\text{rain}},\ Wind) = .970 - (3/5) * 0 - (2/5) * .0 = .970$$

   We can see that for the Sunny branch, we chose Humidity as the root for the highest gain (0.970). For Rain, we chose wind as the root (0.970). For the Overcast branch, we don't need a sub-tree because the entropy is zero.

3. Suppose a bank makes loan decisions using two decision trees, one that uses attributes related to credit history and one that uses other demographic attributes. Each decision tree separately classifies a loan applicant as "High Risk" or "Low Risk." The bank only offers a loan when both decision trees predict "Low Risk."

   (a) Describe an algorithm for converting this pair of decision trees into a single decision tree that makes the same predictions (that is, it predicts non-risky only when both of the original decision trees would have predicted non-risky).

   (b) Let n_1 and n_2 be the number of leaves in the first and second decision trees, respectively. Provide an upper bound on n, the number of leaves in the single equivalent decision tree, expressed as a function of n_1 and n_2.

Let T_1 represent the set of leaves in the first tree, and T_2 the set of leaves in the second. So,

$$|T_1| = n_1$$

$$|T_2| = n_2$$

Let l_1 be the number of low-risk leaves in T_1, and define l_2 similarly for T_2.

$$l_1 = |\{x \in T_1 \mid x \equiv \text{"low-risk"}\}|$$

$$l_2 = |\{x \in T_2 \mid x \equiv \text{"low-risk"}\}|$$

Without loss of generality, we can assume $l_1/n_1 \leq l_2/n_2$. To construct the new tree, for each low-risk leaf in T_1, we replace the leaf with the tree T_2. That is the leaf node in T_1 is replaced with the root node of T_2 and the associated sub-tree.

The resulting tree meets the requirements. Each high-risk leaf from T_1 will still result in a rejection leaf. Each low-risk leaf from T_1 will now result in walking over T_2, only reaching a low-risk leaf if both T_1 and T_2 are low risk.

The resulting number of leaves of the new tree will be,

$$(l_1) * (n_2) + (n_1 - l_1)$$

That is, the number of low-risk leaves in T_1 multiplied by the total number of leaves in T_2, plus the high-risk leaves in T_1. If $l_1/n_1 > l_2/n_2$, we can reverse the roles of T_1 and T_2 in the algorithm.