CS 440/ECE448 Homework 6

Tanishq Dubey (tdubey3)

May 7, 2016

Problem 1

a) **Over-fitting**: When a statistical model represents noise, or unwanted information. This usually happens when our model has unwanted information, has too much information, or is too small.

b) There are two methods that can be used to prune a decision tree:

1. **Reduced Error Pruning**: This method starts at the bottom of the tree and moves upwards. At each node, a algorithm decides which side (class) is most popular and removes the others. If this change does not change the accuracy of the tree for the worse, the change is kept. This method traverses the entire tree.
2. **Cost Complexity Pruning**: This method generates a series of trees, where each tree has a reduced number of sub-trees (essentially, pre-pruned) and the final tree is just the root. Next, each sub-tree is tested for error. The sub-tree that has the most error is removed. After all trees have been generated, the one with the least error is kept as the final pruned tree.

c) Pruning helps address the problem of over-fitting by removing sub-trees that do not contribute to the decision tree, but instead only add complexity and noise.

d) Since pruning is used to address the problem of over-fitting, the alternatives to pruning must also address the same problems. One such method is to reduce the various parameters of the decision tree, thus artificially reducing the chance of over-fitting. This includes limiting depth, height, or even the number of tests performed on the tree. Another method is to estimate parameters to try and make a optimal tree, but this should be combined with limiting the tree.

Problem 2

$$n = \frac{k^{h+1} - 1}{k - 1}$$

$$L = k^h$$

$$h = log_k h$$

$$n_{min} = \frac{k^{h+1} - 1}{k - 1} - L$$

$$n_{min} = \frac{k^{log_k L+1} - 1}{k-1} - L$$

Thus:

$$n_{min} = \frac{kL - 1}{k-1} - L$$

Problem 3

a. To calculate entropy, of a single attribute, the following formula can be employed:

$$E(s) = \sum_{i=1}^{c} -p_i log_2 p_i$$

Where, C is the attribute we wish to find the entropy of. In our case, we only wish to find the entropy of the entire set of data. Thus we can calculate this value as such:

$$-\frac{200}{400}log_2(\frac{200}{400}) - \frac{200}{400}log_2(\frac{200}{400}) = 1$$

b. To see information gain, the entropy of the various types of player abilities. With this we get the following sets of data:

Male Player Strength	Play Together Count	Not Play Together Count
Smash	100	100
Drop	100	100

Female Player Strength	Play Together Count	Not Play Together Count
Smash	100	100
Drop	100	100

Same Handedness	Play Together Count	Not Play Together Count
YES	70	140
NO	130	60

We can then calculate entropies for each attribute:
Male Player Strength:

$$E(PlayCount, MalePlayerStrength) = P(Smash) * E(100, 100) + P(Drop) * E(100, 100)$$

$$= (.5 * -\frac{100}{200}log_2(\frac{100}{200}) - \frac{100}{200}log_2(\frac{100}{200})) + (.5 * -\frac{100}{200}log_2(\frac{100}{200}) - \frac{100}{200}log_2(\frac{100}{200}))$$
$$= .5 * 1 + .5 * 1$$
$$= 1$$

Female Player Strength

$$E(PlayCount, FemalePlayerStrength) = P(Smash) * E(100, 100) + P(Drop) * E(100, 100)$$

$$= (.5 * -\frac{100}{200}log_2(\frac{100}{200}) - \frac{100}{200}log_2(\frac{100}{200})) + (.5 * -\frac{100}{200}log_2(\frac{100}{200}) - \frac{100}{200}log_2(\frac{100}{200}))$$
$$= .5 * 1 + .5 * 1$$
$$= 1$$

Same Handedness

$$E(PlayCount, SameHandedness) = P(YES) * E(70, 140) + P(NO) * E(130, 60)$$
$$= (.525 * -\frac{70}{210}log_2(\frac{70}{210}) - \frac{140}{210}log_2(\frac{140}{210})) + (.475 * -\frac{130}{190}log_2(\frac{130}{190}) - \frac{60}{190}log_2(\frac{60}{190}))$$
$$= .525 * .9183 + .475 * .8997$$
$$= .9094$$

Thus the information gain for the three split categories is:

Split	Information Gain
Male Player Strength	0
Female Player Strength	0
Same Handedness	.0905

Based on this data, "Same Handedness" would be the root of the tree as per the ID3 algorithm.

c. As stated in problem 2, the minimum number of nodes needed for a k-ary decision tree with L distinct classes is:

$$n_{min} = \frac{kL - 1}{k - 1} - L$$

Using this formula with $k = 2$ and $L = 8$, we get:

$$n_{min} = \frac{16 - 1}{1} - 8$$
$$n_{min} = 12$$

Using the generic height of tree formula - $log_2 N$, with N being the number of nodes - we can get:

$$log_2(12) = 3.5849625007211561814537389439478165087598144076924810604557526545410982277943585625222804749180882420909806624750591673437175524410609248221420839506216982994936575922385852344415825363027476853069780516875995544737266834624612364248850047581810676961316404807130823233281262445248670633898014837234235783662478390118977006466312634223363341821270106098049177472541357330110499026268818... \approx 3$$

Thus the height of our tree is 3.

d. If we have a tree limited to a height of one, then to calculate the error of the tree, we simply compare what we expected to come out in our tree, to what we actually obtain. After parsing this tree, we can see that our error is 32.5%. This is because we use the equation

$$Error = \frac{Observed - Expected}{Expected}$$

to find the error. Our observed value is the value of how many people played together when they should have, while our expected is as stated.

e. Simply by looking at the data, it can be seen that a simple sorting algorithm for this data is seeing if the players have opposite attributes. By picking this split, we get 100% accuracy for our data with a tree that looks like such:

$$\begin{tikzcd}
 & \boxed{Stats} \arrow[dl] \arrow[dr] & \\
\boxed{OppositeStats} \arrow[d] & & \boxed{NotOppositeStats} \arrow[d] \\
\boxed{YES} & & \boxed{NO}
\end{tikzcd}$$

f. This tree has a height of 2, which is smaller than the ID3 tree because it is based off of patterns that a human can see but the ID3 algorithm cannot. In this case then, the smaller tree corresponds to the more accurate version

g. As previously stated, the ID3 will not produce a globally optimal solution because it cannot see all the patterns present in the data, and thus cannot build a globally perfect tree.