CNSCC.361 Artificial Intelligence

Lab Session 8 Decision Trees

This lab is about decision trees. You will try to build decision trees with a well-known "iris" data set. You do not have to finish everything here during the 2-hour lab session, and can look at this data set more on your own.

Computing Information Gain

Recall from lecture that the entropy (or amount of uncertainty) of an answer to a question with possible answers v_1, ..., v_n and associated probabilities is:

$$I(P(v_1), \ldots, P(v_n)) = \sum_i -P(v_i) \log_2 P(v_i)$$

And at each node of the tree, we choose which attribute to split the data on by choosing one that ideally reduces uncertainty and result in a higher "information gain". The information gain (IG) or reduction in entropy from using attribute A is:

$$IG(A) = \text{Entropy before} - \text{AverageEntropy after choosing A}$$

At each node of the tree, we compute this IG score for each attribute and then choose the attribute with the highest score. And we repeat this down the tree at every node, until all the data is classified to one class at every leaf node of the tree, in which case we have our decision tree.

Training Dataset ("train_data.mat" & "train_label.mat")

Use the above method to build a decision tree for the "iris" dataset. In this part, we only use training dataset. Start by trying to find the root node of the tree. Go through each of the four attributes and compute an information gain score for each. In order to split the data to compute a score, you have to manually choose a value (i.e. a real number). This is slightly different from the examples in lecture, as the values in this data set are real numbers (and not just discretised categories). Try to choose a value that splits the data into separate classes as much as possible (i.e. this is the same as reducing entropy as much as possible). Since you cannot try "all" possible values, there is no right or wrong answer as to which value you pick. If you look at the data, you should get a sense of what kind of values to try.

Instead of writing some code to do this, you can manually look through the data and compute the scores directly from the MATLAB command line. It helps you to understand the decision tree method as a whole, if you try to build a tree with a dataset manually. In MATLAB, you can use the "sortrows()" function to sort the data to get a sense of what kind of values to split the data on, and the log2() function to compute logarithms.

For the root node of the tree, what is the "best" attribute and value to split on? Draw this out so you can visualise it (use the last blank page).

Repeat the above process down the tree. At each node where the data is not classified to only one class, go through each attribute, find a value to split the data on, and compute the scores to choose

the “best” attribute/value to split the data. Then, split the data, and draw the results in your tree. This process finishes when the data at each leaf node has been classified to only one class, in which case you have your decision tree. Draw the whole decision tree. If you have done this correctly, the tree should be “as small as possible”.

Testing Dataset (“test_data.mat” & “test_label.mat”)

Once you have the tree, you can test its performance by testing the tree on a new set of data. The idea is to take each new sample, and use the four attribute values and your tree to classify that sample. And then see if the classification from your tree matches with the class in the testing data. You can do this for all the samples to get a percentage (%) of the testing data that is correctly classified. This % gives you a sense of the classification accuracy that your tree provides.

Try this with the testing data in the “iris_data_25x3_test.txt” file. What percentage (%) of this testing data is correctly classified by your tree?

Additional Testing

Note that the two data files (“test_data.mat”, “train_data.mat”) are from the same data set – the original “iris” data set. We have just separated the whole data set of 150 samples into two parts so you can test your tree built with the training data with some new testing data. So, depending on the data set and how we split it into training and testing data, we may come up with different trees and different % accuracies, even on the same data set.

For example, you could have used the testing data above to build the tree, and then the training data above to test the tree. Try this. Are there any differences with your tree, and the % accuracy?