Lab Session 6 Decision Trees

Key aims of the Session: Gain a working understanding of

1. Entropy
2. ID3 Algorithm
3. Decision Trees

Contents

1: Decision Trees 1

1.1 Computing Entropy and Information Gain 1

1.2 Problem Definition 2

1.3 Tasks 2

Editor vs Command Window

While you can complete this worksheet by entering commands sequentially into the command window, you may find it more useful to save and run the commands as a script. To open up the script editor, type `edit` into the command window. Don't forget to save your script so that you don't lose it. Once saved, you can run your script by clicking "Run" on the top ribbon or by typing the script's name into the command window. Please note that MATLAB script filenames should have a .m extension.

1: Decision Trees

In this section, we will build decision trees with a small dataset: `IvyLeague.txt`.

1.1 Computing Entropy and Information Gain

The **entropy** (uncertainty amount) E of an answer to a question with possible answers $v_1, \dots, v_n$ and associated probabilities is

$$E\big(P(v_1), \dots, P(v_n)\big) = \sum_{i=1}^{n} -P(v_i)\log_2 P(v_i)$$

At each node of a decision tree, we choose the attribute to split the data on by choosing one that reduces uncertainty and results in higher information gain. We calculate the information gain of using attribute A as:

$$\text{Gain}(S, A) = E(parent) - [Weighted\ Average]E(children)$$

We compute this gain score at each node for each attribute and then choose the attribute with the highest score. We repeat this down the tree at every node, until all the data is classified into one class at every leaf node of the tree. This results in our decision tree.

1.2 Problem Definition

In a typical classification task, we have a training and a validation / testing dataset. The training dataset includes the set of features and their class labels. We use a learning algorithm to train a classifier using the training dataset. In this lab, we will use the Decision Tree classifier. The validation dataset is a set of feature vectors to which the classifier must assign labels.

The folder `Wk6 Lab.zip` contains the key code that you will need, including the dataset file `IvyLeague.txt`, which contains 62 samples.

1.3 Tasks

1) Read through the code to understand what each part of the code is doing and relate it back to the theory. In particular:

   a) Locate the part of the code in `decisiontrees.m` that carries out the split of the dataset into training and validation sets and understand how this is achieved.

   b) Locate the `ID3()` function that is called by `decisiontrees.m` and is used to determine the optimal decision tree, and relate it back to the theory.

   c) The `ClassifyByTree.m` file takes a (validation) sample and pass it through the learned decision tree. Try to understand how it works.

   d) The `PrintTree.m` file takes a decision tree and outputs the structure (nodes and their connections) of the tree. Try to understand how it works.

2) On line 134 of the `ID3().m` file, there is some code missing for calculating information gain. Based on the code and the above, complete this line with your own code to calculate information gain.

3) Implement the code with the data using the `decisiontree` function contained in the zip file. You will need to specify:

   a) The input filename of the dataset

   b) The size of the training set, i.e. the number of examples from the dataset that will be used to train the model. We'll use 50.

   c) The number of trials to run, i.e. the number of times a decision tree will be built from a randomly selected subset of training examples. We'll use 1 initially.

4) Try changing the values of the training set size and number of trials. Try to understand how they work.

5) Read about what the `randsample` and `rng` functions do. What is the purpose of `rng(10)` in `decisiontree.m`? Try running the code several times with this line included, and then several times with this line commented out. Does anything change regarding the result? Why / why not?

6) Although the method should be fast, there are inefficiencies in the code. Try to find and fix them.