

# CSI 5325 Assignment 2

Greg Hamerly

assigned January 29, 2008; due February 12, 2008

## 1 Expectations

For all assignments in this class you should work individually, but feel free to discuss your work with me or your classmates.

The writeups you do for this course should be of high quality; complete but concise. Please structure your writeups well, introducing ideas logically (rather than simply chronologically). All this means that you should go over your writeup multiple times, editing as necessary.

Use graphics in your document where they will help – for example when describing a complex object, and especially when giving experimental results.

### 1.1 Submitting your work

You should turn in all your work in printed format in class, as well as by email to `hamerly@cs.baylor.edu`. Your email should have attached a zip file which has a single folder that contains all your materials. The zip file name should follow this pattern: ‘`lastname_nn.zip`’, where lastname is your last name, and nn is the assignment number (like ‘01’). Please DO include all source code with your emails, but DO NOT print out your source code to hand in (just print and turn in writeup).

### 1.2 Tools for the course

As suggested in assignment 1, your work should be composed in  $\text{\LaTeX}$ , and I suggest looking at MATLAB for programming.

## 2 Implement basic ID3

Implement a basic ID3 algorithm. Here ‘basic’ means having binary target values, and discrete attribute values. It might be helpful to play with Weka software to get an idea of how your trees should look, and to compare results on your tests.

Your software should be able to operate in 2 modes:

1. Learning from training examples. Input is a set of examples, output is a decision tree.
2. Making predictions on unseen examples. Input is a learned decision tree and a set of examples, output is a prediction for each example and an error rate on those examples whose true label is known.

### 2.1 Input/output format

Let’s use common formats for input and output of examples and decision trees.

### 2.1.1 Format for reading examples

A set of examples will use the following conventions:

- An set of examples should be plain text. There are two parts: the attribute descriptions come first, followed by the examples.
- The attribute description starts with a number  $d$  describing the number of attributes. The next  $d$  lines describe the attributes and their possible values, formatted as `name value1 value2 ...`.
- Each example will occupy exactly one line. Each example will start with a target label, which is either -1 (negative class), 1 (positive class), or 0 (which indicates ‘don’t know’). The ‘don’t know’ option is useful when we want a learned decision tree to make predictions on unknown examples. Following the target label will be a space-separated list of the value for each of the attributes. The values should come in the order the attributes were originally described.

Here is an example dataset adapted from your textbook, described in the above format:

```
4
Outlook Overcast Rain Sunny
Temperature Cool Hot Mild
Humidity High Normal
Wind Strong Weak
-1 Sunny Hot High Weak
-1 Sunny Hot High Strong
1 Overcast Hot High Weak
1 Rain Mild High Weak
1 Rain Cool Normal Weak
-1 Rain Cool Normal Strong
1 Overcast Cool Normal Strong
-1 Sunny Mild High Weak
1 Sunny Cool Normal Weak
1 Rain Mild Normal Weak
1 Sunny Mild Normal Strong
1 Overcast Mild High Strong
1 Overcast Hot Normal Weak
-1 Rain Mild High Strong
```

### 2.1.2 Format for writing/reading decision trees

A decision tree description (which is used for output after learning, and also input when making predictions) will use the following conventions:

- Print one node per line, in preorder format. Therefore, the root is first, followed by its first child, followed by its first child (if any), etc.
- Each non-leaf node should be printed in the following way: `parent_value name p n`, where `name` is the attribute name, `p` is the number of positive training examples that are sorted into the subtree rooted at this node, and `n` is the corresponding number for the negative training examples. The `parent_value` is the branch value of the parent node’s attribute that leads to this node. Of course, the root is does not have a parent node, so we will denote its `parent_value` as `ROOT`.
- Each leaf node should be printed just like the non-leaf nodes, but with the name `LEAF` (in capital letters).

Here is an example decision tree adapted from Figure 3.1 of your textbook. It is described in the above format, with indentation to make it easier to read. You can use indentation if you would like.

```
ROOT Outlook 9 5
  Sunny Humidity 2 3
    High LEAF 0 3
    Normal LEAF 2 0
  Overcast LEAF 4 0
  Rain Wind 3 2
    Strong LEAF 0 2
    Weak LEAF 3 0
```

### 3 Develop a learning problem for ID3

Come up with your own learning problem to be solved by your ID3 software. Use something you find interesting – spam identification, tic-tac-toe, predicting a whether a person will like a product, etc. Develop a dataset of examples and run experiments training your software on some of the data, while holding out some test data to judge the success of your method. Make sure you have sufficient data – the more the better.

Describe your dataset and your experiments. Report your results and discuss them.

You may develop a dataset with one other person, if you wish, but you should run your own experiments and write up your results individually. It could be interesting to develop the same datasets but choose different concepts (target values) to learn.

### 4 Expand your ID3 algorithm

Pick a particular way in which ID3 may be expanded, and add that to your software. It would be good if the expansion would be useful in the context of the experiments you ran in the previous step. Here are some ideas for expansion:

- Continuous-valued attributes.
- Multi-class classification (i.e. more than just positive/negative).
- Tree pruning to avoid overfitting (like rule post-pruning or reduced-error pruning, etc.).
- Incorporate a cost model for using various attributes.
- Allow unknown attribute values.
- Allow real target values; that is, real numbers instead of classes. For example, each leaf node could represent a constant value (such as the mean of the target value for the examples in the leaf). The method for attribute selection would have to use something other than information gain, such as average squared distance from the mean value. This sort of tree is called a regression tree.

After expanding your software, use it to run more experiments on your previous learning problem.

Describe the expansion you tried, how it has changed the problem, and discuss your experiments and results.