

CS 374 Assignment #6 Decision Trees

Due the week of March 10, 2008

This week we switch gears and look at a popular non-parametric technique for classifier learning that is quite different from those you've explored in the last two weeks: top-down induction of decision trees.

1 Decision Trees and the C4.5 Algorithm

We will focus our attention on a particular decision tree learning algorithm: C4.5, which builds decision trees by recursively selecting attributes on which to split. The criterion used for selecting an attribute is information gain.

Two of you are already familiar with C4.5. Since some of you know it quite well and others don't at all, I've devised two different assignments for the week. As you read the rest of this handout, please note the parts that are relevant to you.

1.1 Reading – All

There are many good sources of information on decision trees and the C4.5 algorithm. You might want to quickly skim Alpaydin, Sections 9.1-9.3 first. Then I recommend Mitchell, Chapter 3, which should be your primary source for this topic. If you're interested in other sources, you can also look at the following:

- Russell and Norvig (now on the Machine Learning shelf), Section 18.3,
- Ross Quinlan's paper "Induction of Decision Trees", which appeared in Volume 1 of the journal *Machine Learning*;
- Ross Quinlan's book, *C4.5: Programs for Machine Learning*, which you'll find on the shelf.

1.2 Seeing C4.5 in action – All

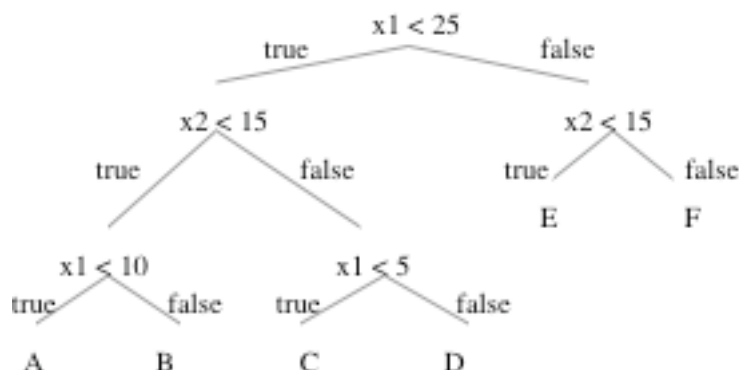
You might find it useful to run C4.5 to get a sense of what it does and how well it works. Once again, you can use weka to do this.

As usual, begin by selecting a data set. You can do this by clicking on "Open file..." and then selecting an "arff" file of your choice. (I've put some new data in the 374 directory, in the same place where you'll find Weka.)

Once you've selected a data file, click on "Classify" and then "Choose". In the "Trees" directory, click on J48 (which is really C4.5). Then click on "Start". The output of the classifier will appear in the right half of the window. For the smaller data sets (like "weather"), you might want to set "Test options" to "Use training set" to see the tree produced by the full data set.

1.3 Exercises – All

1. (From Dietterich) Consider the following decision tree:



- (a) Draw the decision boundaries defined by this tree. Each leaf is labeled with a letter. Write this letter in the corresponding region of instance space.
- (b) Give another decision tree that is syntactically different but defines the same decision boundaries.
2. (From an exercise by Terran Lane) Consider a two-category classification task with the following training data:

$attr_1$	$attr_2$	$attr_3$	$attr_4$	$class$
a	1	c	-1	c_1
b	0	c	-1	c_1
a	0	c	1	c_1
b	1	c	1	c_1
b	0	c	1	c_2
a	0	a	-1	c_2
a	1	a	-1	c_2
b	1	c	-1	c_2

Construct a complete (unpruned) decision tree for this data using information gain as your splitting criterion. Please show all entropy calculations.

3. (Modified from Russell and Norvig) In the recursive construction of decision trees, it sometimes happens that a mixed set of positive and negative examples remains at a leaf node, even after all the attributes have been used.

Suppose that you have learned a decision tree for a particular two-class problem, where 1 represents the positive class and 0 represents the negative class. Furthermore, assume that you have p positive examples and n negative examples at the leaf.

- (a) Show that the solution which picks the majority classification, minimizes the absolute error over the set of examples at the leaf.
- (b) Show that the class probability $p/(p+n)$ minimizes the sum of squared errors.

4. This exercise will have you consider an interesting property of the entropy function.

For all parts of this exercise, you should assume a binary classification task, where all attributes are binary as well.

- (a) Show that the entropy function is concave. (Please don't just draw a picture.) If you do so by finding the second derivative of the entropy function, you'll find it convenient to substitute \ln for \log_2 .
- (b) Suppose that a binary-valued attribute splits a set of examples E into subsets E_1 and E_2 , and that the subsets have p_1 and p_2 positive examples and n_1 and n_2 negative examples, respectively. Show that the attribute has 0 information gain if the ratios $p_1/(p_1 + n_1)$ and $p_2/(p_2 + n_2)$ are the same.
- (c) A function $f(x)$ is concave on an interval $[a, b]$ if for any two points x_1 and x_2 in $[a, b]$ and any λ , where $0 < \lambda < 1$,

$$f(\lambda x_1 + (1 - \lambda)x_2) \geq \lambda f(x_1) + (1 - \lambda)f(x_2)$$

That is, the value at the midpoint of every interval in the domain exceeds the average of its values at the ends of the interval.

Use this to show that every attribute has non-negative information gain.

- (d) What does part c imply about the information content of the data and about the process of constructing a decision tree?

1.4 Implementation – non-CSCI 373 alums, with advice from 373 alums

In order to get a better sense of C4.5 (and to have fun doing more coding), please implement a decision tree learner. You may do this alone or in a group. (All three of you may do the implementation together, if you'd like.) I'm happy to help with ideas about design, debugging, etc, but I hope the 373 alums will also share their experience with you.)

Your program should handle data in the usual format. It should handle data sets with discrete-valued attributes, an arbitrary number of classes, and an arbitrary number of instances. You need not handle missing values (though doing so is a fairly simple extension – see suggestions below).

As usual, the design of the user interface is up to you. You will need to read the appropriate data files, produce a human-readable version of the decision tree created from the data set, and be able to use the decision tree to classify test examples (producing, of course, interesting statistics such as accuracy on the test set, etc.). For reasonable-sized data sets, don't forget to reserve a random test set. You might even want to implement 10-fold cross validation this time.

The decision tree learning algorithm Your algorithm must use the information gain splitting rule. As described in the reading, this information theoretic metric selects the feature to split on based on the gain in information (measured in bits) obtained by that split. In particular, if T is the set of training data (containing $|T|$ instances) representing a concept with c classes, and $freq(C_j, T)$ is the frequency of class C_j occurring in that set, then the expected information needed to identify a given class in T is:

$$info(T) = - \sum_{j=1}^c \frac{freq(C_j, T)}{|T|} \times \log_2 \left(\frac{freq(C_j, T)}{|T|} \right)$$

bits.

When a feature, X , with v values, has been selected as a test feature, then the expected information needed to identify a class under that test is:

$$info_X(T) = \sum_{i=1}^v \frac{|T_i|}{|T|} \times info(T_i)$$

where T_i is the subset of T all of whose instances possess value i for feature X .

The information gain, then, is simply the difference between the expected information needed to identify a class with and without the test on feature X :

$$gain(T) = info(T) - info_X(T)$$

Thus, the feature yielding the highest information gain is selected as the current split. Note that when splitting at a node of the tree other than the root, the set of instances under examination, T , will be only a subset of the whole set of training instances (in particular, it is the subset of training instances that have propagated down through the nodes that are parents of the node under consideration).

There are many ways in which you can extend this assignment. Possible extensions include:

- Handling continuous-valued attributes.
- Handling missing values.
- Implementing reduced-error pruning: This pruning criterion replaces a subtree with a single leaf when that leaf represents an error rate no worse than the subtree. The pruning procedure thus has the following steps:
 - After the full tree has been constructed from the training data
 - In a postorder fashion traverse the decision tree, replacing a non-leaf node with a leaf node when the errors at the leaf node do not exceed the errors made by the subtree. Measure the errors on a separate set of instances (called the pruning set).

2 Difficult Data

Of course, no learning algorithm is perfect. Decision tree learning algorithms have difficulty handling certain types of “hard” data.

2.1 Reading – All

Read “Skewing: An Efficient Alternative to Lookahead for Decision Tree Induction” by Page and Ray, which appeared in IJCAI-03.

2.2 Exercise – CSCI 373 alums

Write a thorough summary and critical analysis of the “Skewing” paper. For the tutorial session you should be prepared to present your summary and critique and to *lead the group discussion of the paper*.