# CSE 176 / EECS 276
# Machine Learning
# Specification for Programming Assignment #3

David C. Noelle

**Due:** 11:45 P.M. on Sunday, March 17, 2013

## Overview

The primary learning goal of this programming assignment is to provide you with a more thorough understanding of the most standard method for learning decision trees from labeled examples. Your efforts to implement the specified program are intended to expose you to issues surrounding the representation of decision trees and the use of information theoretic measures to construct such trees.

In order to meet this goal, you will write a program that implements a simple version of the `ID3` algorithm, as it is described in Mitchell's textbook entitled *Machine Learning*. Your program will support boolean input feature vectors (zero or one) with arbitrary dimensionality, and it will produce binary classifications. Once your program is complete, you will demonstrate its application on a provided set of training and testing patterns.

## Submission for Evaluation

In order to submit your solution to this assignment for evaluation, you must generate a GNU Zipped TAR archive file, as described below, containing all of your source code and an assignment report. This archive file must then be delivered to the teaching team through the UCMCROPS system prior to the assignment due date.

The submitted archive file must unpack into a directory (i.e., "folder") called "`P3`". This directory must contain an assignment report file in PDF format named "`report.pdf`". The "`P3`" directory should also contain a subdirectory called "`code`", which should contain all of the source code necessary to compile and execute your program. (The primary exception to this requirement is that the "`code`" subdirectory need *not* contain standard libraries, including the *GNU Scientific Library*.) Program building and configuration files, such as "makefiles", are appreciated but are not required. The assignment report file should contain:

- a brief description of the program that was generated, including the programming language that was used, any interesting libraries or other tools that were used, instructions for compilation, and instructions for execution
- a trace of the program running on the training set and testing set that will be provided to you
- a graphical depiction of the decision tree that was learned from the given training set
- an analysis of the likely true error of the learned decision tree, based on performance on the testing set
- a brief summary of what was learned from this programming assignment
- a listing of acknowledgments and references, appropriately citing *all* assistance that was received on this project

This report should be clear and easy to read, and it must be provided as a PDF file.

In order to generate the required archive file on a Unix or Linux computer, such as those available in the class laboratory, simply construct a directory called "`P3`" with the contents described above and then execute the following command from the directory containing "`P3`":

```
$ tar czvf P3.tgz P3
```

This will generate a GNU Zipped Tape Archive (TAR) file called "`P3.tgz`". This file is to be submitted for evaluation using the UCMCROPS system.

To submit this archive file, log onto the course `UCMCROPS` website, and navigate to the "Assignments" section. Then, locate "Programming Assignment #3" and select the option to submit your solution for this assignment. Provide your archive file as an attachment to this submission. Comments to the teaching team should largely appear in your report document and as inline comments in your source code, but you may also include brief written comments using the assignment submission web form.

Submissions must arrive by 11:45 P.M. on Sunday, March 17th. Please allow time for potential system slowness immediately prior to this deadline. You may submit assignment solutions multiple times, and only the most recently submitted version will be evaluated. (If there are difficulties with performing multiple submissions, make use of the "Drop Box" feature on UCMCROPS to submit subsequent versions *and* send a note to the instructor indicating that you have done so.) As discussed in the course syllabus, late assignments will not be evaluated.

If your last submission for this assignment arrives by 11:45 P.M. on Friday, March 15th, you will receive a 10% bonus to the score that you receive for this assignment. This bonus is intended to encourage you to try to complete this assignment early.

# Activities

You are to provide a computer program that implements the basic `ID3` decision tree learning algorithm. Specifically, your program is to read a training set of patterns from a file and a testing

set of patterns from an additional file. Using only the training patterns, the program should produce an output for each of the testing set patterns, and it should report this classification output for each pattern, along with a measure of output accuracy.

Your program should take a single input, either on the command line or entered at the terminal upon execution. This input should be the name of a plain text "configuration" file that contains a description of the current run. Your program should proceed to open this configuration file and read its contents. It should then use the read information to prepare a decision tree learning system with the specified properties. Each configuration file is expected to contain the following fields, in this order, in plain text format, with one field per line:

1. $d_{in}$ — the dimensionality of the binary input vectors, as a whole number
2. $N_{train}$ — the number of patterns in the training set
3. training set file name — the name of a file containing the set of training patterns to be used
4. $N_{test}$ — the number of patterns in the testing set
5. testing set file name — the name of a file containing the set of testing patterns to be used
6. output file name — the name of the output file, which should be created and filled with plain text output

Given this configuration information, your program should read the set of training patterns and testing patterns. Each pattern will consist of $d_{in}$ binary input feature values and a single binary classification target value, indicating if the given pattern is a positive or negative example. (Values below $0.5$ should be treated as a binary zero — *off* or *false* — and larger values should be treated as a binary one — *on* or *true*.) Given this information, your program should then produce an output classification for each input vector in the testing set, based on a decision tree inferred from the training set. It should open a new plain text file with the given output file name, and it should write one line to this file for each of the testing set patterns. Each output line should contain a row of numbers, consisting of:

1. input vector — the current test pattern input vector should be written first on the line (ideally, this should contain only zero and one values)
2. output value — a single number, either zero or one, indicating assignment into either the non-target or target category, respectively
3. target value — the "target" output value for the current test pattern, either zero or one, as specified in the testing set file, should appear next on the line
4. error — a single value indicating errors, with zero indicating a correct output and one indicating a mismatch between the output and the target for this pattern

Thus, the output file will contain one line for each testing set pattern. One additional line should be appended to this output file, however, and this last line should contain a single real value: the fraction of testing set patterns misclassified (i.e., $error_S(h)$) by the decision tree.

The pattern files to be read by your program — both the training set file and the testing set file — are to be plain text files that contain one pattern on each line of the file. Thus, each line will contain a sequence of numbers separated only by whitespace. The first $d_{in}$ numbers on each line form the input vector for that pattern, and the remaining single value is the corresponding output target. For each pattern in the training set, the output values need to be recorded so that they can be used by the `ID3` algorithm. For each pattern in the testing set, the output values are provided to allow you to assess how well your learning algorithm generalizes. If a pattern set should contain a number that is not zero or one, values less than $0.5$ should be read as zero values and larger real numbers should be read as one values.

The source code that you submit for evaluation should be of your own design. You may use openly available standard libraries in order to construct your program, however. In particular, if you are writing your program in C or C++, you are encouraged to make use the rich collection of tools provided by the *GNU Scientific Library*. This library may be found at:

`www.gnu.org/software/gsl/`

This web site also contains extensive documentation describing the functionality provided by the library.

You will be provided with a training set file and testing set, called "`lenses-train.dat`" and '`lenses-test.dat`", respectively. These will be made available to you through the UCM-CROPS system. Once you have implemented your program, you may use these files to test your code. Additionally, you are expected to exercise your program on these pattern sets and briefly describe what you observe in the written report that you are required to submit with your source code. In particular, you should:

- provide a trace of your program running on the given training and testing sets
- generate a graphical depiction of the decision tree that is learned by your program
- calculate an upper bound on the true error of your decision tree that is $95\%$ likely to hold (i.e., the upper limit of a $95\%$ one-tailed confidence interval), based on performance on the testing set

These analyses should be considered minimal. You are welcome to exercise your program in additional ways and report your observations.

Note that your program need only implement the basic `ID3` algorithm. In particular, your program need not implement any sort of decision tree pruning method. If you decide to implement such pruning, you must provide an easy method to disable this feature, so the performance of your basic `ID3` algorithm can be examined.

Your implementation will be evaluated primarily for accuracy, with efficiency being a secondary consideration. Your source code *will* be examined, however, and the readability and style of your implementation will have a substantial influence on how your assignment is evaluated. As

a rough rule of thumb, consider the use of good software writing practices as accounting for approximately 10% of the value of this exercise. Note also that, as discussed in the course syllabus, submissions that fail to successfully compile will not be evaluated and will receive *no credit*.

As for all assignments in this class, submitted solutions should reflect the understanding and effort of the individual student making the submission. Not a single line of computer code should be shared between course participants. If there is ever any doubt concerning the propriety of a given interaction, it is the student's responsibility to approach the instructor and clarify the situation *prior* to the submission of work results. Also, helpful conversations with fellow students, or any other person (including members of the teaching team), should be explicitly mentioned in submitted assignments (e.g., in comments in the submitted source code files). Failure to appropriately cite sources is called *plagiarism*, and it will not be tolerated!

The members of the teaching team stand ready to help you with the learning process embodied by this assignment. Please do not hesitate to request their assistance.