Assignment 1: Supervised Learning

Due: February 9th 11:55pm
Please submit via T-Square

Why?

The purpose of this project is to explore core techniques in supervised learning. It is important to realize that understanding an algorithm or technique requires understanding how it behaves under a variety of circumstances.

You may program in any language that you wish (we do prefer java, python, matlab, Lisp or C++). In any case *it is your responsibility* to make sure that the code runs on the standard CoC linux boxes.

Read everything below carefully!

The Problems Given to You

You should make use of five learning algorithms. They are for:

- Decision trees with some form of pruning
- Neural networks
- Boosting
- Support Vector Machines
- *k*-nearest neighbors

Each algorithm is described in detail in your textbook, course lectures, and all over the web. Instead of implementing the algorithms yourself, we recommend making use of software packages/libraries/code that you find elsewhere *with attribution* (give credit, specify where you found it). Also, you will note that you have to do some fiddling to get good results, graphs and such, so even if you use another's package, you may need to be able to modify it in various ways.

Decision Trees. For the decision tree, you should implement or download a decision tree algorithm. Be sure to use some form of pruning. You are not required to use information gain (for example, there is something called the GINI index that is sometimes used) to split attributes, but you should describe whatever it is that you do use.

Neural Networks. For the neural network you should implement or steal your favorite kind of network and training algorithm. You may use networks of nodes with as many layers as you like and any activation function you see fit. However note you will need to justify these decisions.

Boosting. Implement a boosted version of your decision trees. As before, you will want to use some form of pruning, but presumably because you're using boosting you can afford to be much more aggressive about your pruning.

Support Vector Machines. You should implement or download SVMs. This should be done in such a way that you can swap out kernel functions. You will need to make use of at least two kernel functions.

***k*-Nearest Neighbors.** You should implement *k*NN. Use different values of *k*.

Testing. In addition to implementing the algorithms described above, you should design two interesting classification problems. For the purposes of this assignment, a "classification problem" is just a set of training examples and a set of test examples.
It is up to you to determine where to acquire training data. You can download some, take some from your own research, or make some up on your own. You will have to explain why the data is interesting, and use the same data in later assignments.

Given that there are two classification problems and five approaches, we expect to see a minimum of **10 train and test error rates**.

What to Turn In

You must submit via T-Square a tar or zip file
named *yourgtaccount*.{zip,tar,tar.gz} that contains a single folder or directory named *yourgtaccount*. That directory in turn contains:

1. a file named *README.txt* containing instructions for running your code
2. your code (see note below)
3. a file named *analysis.pdf* containing your writeup
4. any supporting files you need, such as your training and test sets (see note below).

Note below: if the data are way, way, too huge for submitting, see if you can arrange for an URL. This also goes for code, too. Mailing all of Weka isn't necessary, for example, because the TAs can get that online; however, you

should at least send any files you found necessary to change. In any case, include all necessary information to get your code running in *README.txt*

The file *analysis.pdf* should contain:

- A description of your classification problems, and why you feel that they are interesting. Think hard about this. To be at all interesting the problems should be non-trivial and capable of demonstrating differences between the five machine learning approaches.
- The training and testing error rates you obtained running the various learning algorithms on your problems. At the very least you should include graphs that show performance on both training and test data as a function of training size (note that this implies that you need to design a classification problem that has more than a trivial amount of data) and--for the algorithms that are iterative--training times.
- Analyses of your results. Why did you get the results you did? Compare and contrast the different algorithms. What sort of changes might you make to each of those algorithms to improve performance? How fast were they in terms of wall clock time? Iterations? Would cross validation help (and if it would, why didn't you implement it?)? How much performance was due to the problems you chose? How about the values you choose for learning rates, stopping criteria, pruning methods, and so forth (and why doesn't your analysis show results for the different values you chose?)? Which algorithm performed best? How do you define best? Be creative and think of as many questions you can, and as many answers as you can. **Note: Analysis write-up is limited to 10 pages.**

One example of analysis is available in the resource section of T-Square.

Grading Criteria

This assignment is out of **10 points**, each representing a percentage point of your final grade. The breakdown for grading is as follows:
1 point: A working implementation of all 5 approaches listed above for your two classification problems.
9 points: Your analysis. You are being graded on your analysis more than anything else. However, analysis without proof of working code will harm your analysis grade.

The key thing is that your explanations should be both thorough and concise. Dig into many angles of analysis, but do not dig too deeply.

Follow the directions carefully. Failure to turn in files without the proper naming scheme, or anything else that makes the grader's life unduly hard will lead to an ignored assignment. There will be no late assignments accepted.