ITCS 6156 Spring 2017
Supervised Learning Project
Points: 100

The purpose of this project is to explore some 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. As such, you will be asked to implement some simple learning algorithms, and to compare their performance.
You may program in any language/environment that you wish.
Your programs must be well documented. Use comments to explain the steps throughout your code. If your code is not clear to the grader or cannot be executed, you will be asked to demonstrate you code and analysis.

1. **The Problems Given to You**
   You should implement six learning algorithms. They are for:
   - Decision trees with some form of pruning (with two different splitting criteria)
   - Neural networks
   - Boosting
   - Support Vector Machines
   - *k*-nearest neighbors
   - Naïve Bayes

   Each algorithm is described in detail in the textbook, the handouts, and all over the web. In fact, instead of implementing the algorithms yourself, you may use software packages that you find elsewhere; however, if you do so you should provide proper attribution. Also, note that you will 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. Make sure you examine and understand the code not just a blind-folded use.

   Decision Trees. For the decision tree, you should implement or borrow 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 borrow 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.

   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 ("download") SVMs. This should be done in such a way that you can swap out kernel functions. You are expected to use and compare at least two.

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

Naïve Bayes. You should implement a Naïve Bayes that finds the maximum likelihood parameters for the probability distribution. Make sure to use log-space representation for these probabilities, since they will become very small, and notice that you can accomplish the goal of naive Bayes learning without explicitly computing the prior probability $P(x_i)$. In other words, you can predict the most likely class label without explicitly computing that quantity.

2. **Testing.** In addition to implementing the algorithms described above, you are required to test your implementations using two classification problems. For the purposes of this assignment, a "classification problem" is just a set of training examples and a set of test examples.
   Download the datasets and descriptions available on the assignment description page in Canvas. In the ITCS6156_SLproject.zip archive, you should find two datasets:
   a. **Optical Recognition of Handwritten Digits dataset**
      http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits
   This data set consists of preprocessed normalized bitmaps of handwritten digits from a preprinted form. From a total of 43 people, 30 contributed to the training set and different 13 to the test set. 32x32 bitmaps are divided into non-overlapping blocks of 4x4 and the number of on pixels are counted in each block. This generates an input matrix of 8x8 where each element is an integer in the range 0..16. This reduces dimensionality and gives invariance to small distortions.
   The three files in this archive are:
   optdigits_test.csv: a comma delimited file of the training set
   optdigits_training.csv: a comma delimited file of the training set
   optdigits_names.txt: contains a description of the data
   b. **Amazon reviews sentiment analysis dataset**
      https://snap.stanford.edu/data/web-Amazon.html
   This dataset consists of Amazon baby product reviews a subset of a larger amazon review collection. The dataset is split into training and testing subsets.
   The two files in this archive are:
   Amazon_baby_test.csv: a comma delimited file of the testing set
   Amazon_baby_train.csv: a comma delimited file of the training set

3. **Data exploration Questions:**
   a. What is the number of attributes in each dataset?
   b. What is the number of observations?
   c. What is the mean and standard deviation of each attribute?
   d. What is the distribution of the different classes in each of the datasets?

4. **Report**. Write a summary of your results. Say what the data sets involved and what you aimed to achieve. Discuss any implementation decisions you made and the factors you decided to experiment with. Display and describe the plots you generated, any discoveries you made about the tuning process, what worked best, and your hypotheses on why the results were as you saw. Be clear concise.

Assignment Submissions

What to submit using Canvas (Email submissions will NOT be accepted):
For each algorithm implementation and analysis you will have to submit:

1. **SL_<algorithmName>_results.pdf** – PDF document with your write up for the results

(report).

Note: Your first report should include the answers for the data exploration questions.

2. **SL_<algorithmName>.zip** - An archive of the entire programming project stored in a standard ZIP
   file. Make sure to include all packages and libraries used to run your programs if any.
3. **README.txt** –This file should detail all the files in your project archive, libraries and packages used and any special setup you have used in your programming environment.
4. **INFO.pdf** – PDF document with the following assignment information:
   a. Explanation of status and stopping point, if incomplete.
   b. Explanation of additional functions and analysis, if any.
   c. Discuss the easy and challenging parts of the assignment. How did you overcome all or some of the challenges?

5. **SL_finalReport.pdf** – After finalizing and submitting the implementation for the last algorithm you will need to submit a PDF document with write up including:
   a. Training and testing error rates you obtained running the various learning algorithms on your problems. You should include graphs that show performance on both training and test data as a function of training size and as a function of training times for the algorithms that are iterative.
   b. Analyses of your results. Summarize your key findings, including which factors proved most crucial, and what was the best generalization performance you achieved. 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?

Note: Assessment will be based on the following categories:
Effort: Did you thoughtfully tackle the problem? Did you iterate through methods and ideas to find a solution? Did you explore several methods, perhaps going beyond those we discussed in class? Did you think hard about your approach, or just try random things?
Technical Approach: Did you make tuning and configuration decisions using quantitative assessment? Did you compare your approach to reasonable baselines? Did you dive deeply into the methods or just try off-the-shelf tools with default settings?
Explanation: Do you explain not just what you did, but your thought process for your approach? Do you present evidence for your conclusions in the form of figures and tables? Do you provide references to resources you used? Do you clearly explain and label the figures in your report?