CSCI 5525: Machine Learning (Fall'12)

Homework 2, Due 10/31/12

1. **(35 points)** The Mushroom dataset has 2 classes (edible, poisonous) with 8124 samples, each having 22 nominal features. Feature 11 (stalk-root) has missing values, and will be ignored for the homework. Train and evaluate the following classifiers using 10-fold cross-validation:

   (i) (10 points) A decision stump, i.e., 1 layer decision tree, using Information Gain,

   (ii) (12 points) A 2 layer decision tree using Information Gain.

   (iii) (13 points) A 5 layer decision tree using Information Gain.

   Report: You must include in your report a summary of the approaches used and clearly write down the algorithms (including equations for splitting criterion). Also provide a table of the training-set error rates, test-set error rates, and standard deviations for each fold as well as the average for each case. Finally, include a drawing of the decision tree generated on the entire training set (without cross-validation) for each case.

   Submit: For part (i), you will have to submit code for `myDstump.m` (main file) which takes the filename for the dataset and the number of folds as input, and outputs a vector of error rates for each fold. The filename will correspond to the mat-file for a dataset, containing a column vector "labels" and a matrix "data," where each row correspond to the features of a data point. Put comments in your code so that one can follow the key parts and steps in your code. A typical run of your code from the prompt will look like:
   `testError = myDstump(mushroom,10);`

   The main subroutine for part (i) will be `dstump` which takes a training set (X_{train}, y_{train}) as arguments, and outputs a single feature to split on.

   For parts (ii) and (iii), you will have to submit code for `myDtree.m` (main file) which takes the filename for the dataset, depth of the tree, and the number of folds as input, and outputs a vector of error rates for each fold. The filename will correspond to the mat-file for a dataset, containing a column vector "labels" and a matrix "data," where each row correspond to the features of a data point. Put comments in your code so that one can follow the key parts and steps in your code. A typical run of your code from the prompt will look like:
   `testError = myDtree(mushroom,2,10);`

   The main subroutine for parts (ii) and (iii) will be `dtree` which takes a training set (X_{train}, y_{train}), a test set (X_{test}, y_{test}), and a depth as input, prints the decision tree structure (see below) and outputs the error-rate on the training set and test set. The depth determines the maximum depth of any leaf of the decision tree and splitting is always done using information gain. A typical call of the sub-routine for depth-2 trees (part (ii)) would be of the form:
   `[trainErr,testErr] = dtree(`$X_{train}, y_{train}, X_{test}, y_{test}, 2$`);`

   The decision tree structure can be printed in any reasonable human understandable format. For example, you can simply print the node criteria from left to right with each level as a

new line, e.g.,
$x_1 < 5$
$x_2 < 3$, $x_1 < 7$
`label, label,` $x_2 < 4$`, label`
`label, label`

Briefly describe the format you are using in your report.

2. **(25 points)** Consider the single layer perceptron with a sigmoid transfer function, i.e., for input $\mathbf{x} \in \mathbb{R}^d$, the predicted output

$$\hat{y}(\mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x})} ,$$

and if $y \in \{0, 1\}$ is the true class label, the prediction error is measured by

$$E(\mathbf{w}) = (y - \hat{y}(\mathbf{w}))^2 .$$

Minimizing the above error with respect to $\mathbf{w}$ leads to local minima problems. Consider the following two approaches to getting around the local minima problems:

(a) Replacing the transfer function from the sigmoid function $\hat{y}(\mathbf{w}) = 1/(1 + \exp(-\mathbf{w}^T x))$ to the linear function $\hat{y}(\mathbf{w}) = \mathbf{w}^T x$, and

(b) Replacing the loss function from square loss $(y - \hat{y}(\mathbf{w}))^2$ to the Bernoulli relative entropy

$$E(\mathbf{w}) = y \log \frac{y}{\hat{y}(\mathbf{w})} + (1 - y) \log \frac{(1 - y)}{(1 - \hat{y}(\mathbf{w}))} .$$

Prove that each of these changes get rid of the local minima problem, i.e., the corresponding error function is convex in $\mathbf{w}$. List one potential advantage (other than convexity) and disadvantage of each of the modified formulations.

3. **(40 points)** We consider boosting using different loss functions, and evaluate their performance on the Mushroom dataset using decision stumps.

(a) (10 points) Recall that an additive model constructed using the exponential loss function $L(y, f(x)) = \exp(-yf(x))$ gives Adaboost. Derive the corresponding additive model (known as logitboost) using the logistic loss function $L(y, f(x)) = \log(1 + \exp(-yf(x)))$.

(b) (15 points) Train and evaluate the logitboost classifier using decision stumps using 10-fold cross-validation.

(c) (15 points) Train and evaluate the adaboost classifier using decision stumps using 10-fold cross-validation.

Both boosting algorithms should be run with the following number of decision stumps in the additive model: 5, 10, 15, 20.

Report: You must include in your report a summary of the approaches used and clearly write down the algorithms. Also provide a table of training-set error rates, test-set error rates, and standard deviations for each fold as well as the average for each case.

Submit: For part (ii), you will have to submit code for `myLogitBoost.m` (main file) which takes the filename for the dataset, the number of decision stumps, and the number of folds as input, and outputs a vector of error rates for each fold. The filename will correspond to the mat-file for a dataset, containing a column vector "labels" and a matrix "data," where each row correspond to the features of a data point. Put comments in your code so that one can follow the key parts and steps in your code. A typical run of your code from the prompt will look like:

```
testError = myLogitBoost(mushroom,10,10);
```

For part (iii), you will have to submit code for `myAdaBoost`, with all other guidelines staying the same.

Instructions

Follow the rules strictly. If we cannot run your functions, you get 0 points. Also be sure to cite any and all sources used.

- **Things to submit**
  1. hw2.pdf: The report that contains the solutions to Question 1, Question 2, and Question 3.
  2. myDstump.m, dstump.m, myDtree.m, and dtree.m: The Matlab code for Question 1.
  3. myLogistBoost.m and myAdaBoost.m: The Matlab code for Question 3.
  4. README.txt: README file that contains your name, student ID, email, instructions on how to your run program, any assumptions you're making, and any other necessary details.
  5. Any other files, **except the data**, which are necessary for your program.