# CS273a Homework #3

Introduction to Machine Learning: Fall 2012

**Due: Thursday November 15th, 2012**

**Write neatly (or type) and show all your work!**

## Problem 1: Decision Trees

In order to reduce my email load, I decide to implement a decision tree to ascertain whether or not I should read an email, or simply file it away instead. To train my model, I obtain the following data set of binary-valued features about each email, including whether I know the author or not, whether the email is long or short, and whether it has any of several key words, along with my final decision about whether to read it ($y = +1$ for "read", $y = -1$ for "discard").

| $x_1$ know author? | $x_2$ is long? | $x_3$ has 'research' | $x_4$ has 'grade' | $x_5$ has 'lottery' | $y$ $\Rightarrow$ read? |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | -1 |
| 1 | 1 | 0 | 1 | 0 | -1 |
| 0 | 1 | 1 | 1 | 1 | -1 |
| 1 | 1 | 1 | 1 | 0 | -1 |
| 0 | 1 | 0 | 0 | 0 | -1 |
| 1 | 0 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | -1 |

(a) Calculate the entropy of the class variable $y$

(b) Calculate the information gain for each feature $x_i$. Which feature should I split on first?

(c) Draw the complete decision tree that will be learned from these data.

## Problem 2: VC Dimension and Decision Trees

(a) Consider a decision tree with *one* feature $x_1$, and depth less than $D$. What is the VC dimension of this learner?

(b) Now consider a decision tree with *two* features. For $D = 1$, how many parameters does this decision tree have? Show by example that for $D = 1$, the learner can shatter three points, but argue that it cannot shatter four. How does this compare to your answer in the previous part?

(c) Now consider a decision tree with *four* features, still with $D = 1$. In this case, how many parameters does the model have? Modify your previous example to show that this learner *can* shatter four points. (If you're feeling ambitious, show that you can also do it with *three* features instead.) What does this suggest about the VC dimension of a depth-$D$ decision tree?

## Problem 3: Decision trees; bagging

Download the SpamBase data set from UCI's machine learning repository:
`http://archive.ics.uci.edu/ml/machine-learning-databases/spambase/spambase.data`

These data are pre-processed features of emails, along with a class indicating whether the associated email was spam or not. They are a bit out of date; modern spam prediction data sets would have hundreds of thousands of features, rather than a few dozen.

Pre-process and split the data into training and validation sets:

```
rand('state',0); randn('state',0);
data = load('data/spambase.data');
X = data(:,1:57); Y=data(:,end);                    % extract features & class labels
[X,Y] = reorderData(X,Y);                           % permute out of class-based order
[Xt,Xv,Yt,Yv] = splitData(X,Y,.6);                  % divide into training & test
[Xt,S] = rescaleData(Xt); Xv = rescaleData(Xv,S); % pre-process data scaling
```

(a) Use either Matlab's or my own provided decision tree code (`treeClassify`) to learn a decision tree classifier for the data, and report its training and validation accuracy. Report the settings you used to learn the tree (i.e., leaf criteria, split impurity score function, etc.)

(b) Now search over values for the maximum depth cutoff for the tree, and score both training and test. Plot the results and report what value of depth you would select and why.

(c) Now use *no* maximum depth, but use the `minParent` option to control complexity. Again report training and validation errors for a range of values, what value you prefer, and why.

(d) Now, learn a bagged ensemble of decision trees. (See the pseudocode from lecture slides.) For your individual learners, don't use any complexity control (no depth cutoff or minParent). For the bootstrap process, draw the same number of data as in the original data set ($N' = N$). Plot the training and validation error as a function of the number of learners you include in the ensemble.

(e) Now, choose and fix a number of learners to include in the ensemble (say, 25). Re-run your bagged ensemble learning algorithm, but now sample fewer data points per bootstrap set. (I suggest spacing values of $N'$ logarithmically between say $0.1\,N$ and $N$.) Again, plot training and validation error as a function of $N'$. Explain briefly what is happening and why.

## Problem 4: Boosting

Consider the following classification problem. You wish to use boosting to learn a classifier consisting of a decision stump – using a single feature, a threshold on the value of $x_1$ (horizontal axis) or $x_2$ (vertical axis) at each level. Find the first three classifiers learned by AdaBoost, where your weak learning algorithm at each stage directly minimizes the weighted classification error over all possible decision stumps (there are only 5 viable classifiers to check). At each step, compute the weighted error of your newly selected classifier, the resulting classifier weight alpha, and the new example weights to be used at the next stage. Show how you computed these values.

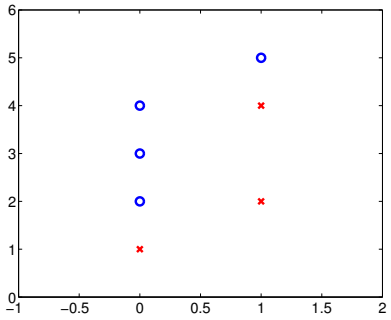


Figure 1: Data for AdaBoost problem.

If you compute the error of the overall (ensemble) classifier (not required), you will notice that it does not improve between the first and second weak learners. (It does improve on the 3rd.) Why is this the case?