CS 380 Machine Learning - Spring 2011
Homework Assignment 3

Due at the start of class on March 14^{th}

1. (Regularization – 10pts) Skim Sections 4.8 (pg 80 only) and 11.10 on the concept of regularization. Explain briefly how SVMs employ regularization to control the model complexity (hint: look at the SVM optimization problem). How might we incorporate regularization into logistic regression? Write the modified equation for the log-likelihood that incorporates regularization.

2. (Probability decision boundary – 10pts) Consider a case where we have learned a conditional probability distribution $P(y|\mathbf{x})$. Suppose there are only two classes, and let $p_0 = P(y = 0|\mathbf{x})$ and let $p_1 = P(y = 1|\mathbf{x})$. Consider the following loss matrix:

	$y = 0$ (true)	$y = 1$ (true)
$\hat{y} = 0$ (predicted)	0	10
$\hat{y} = 1$ (predicted)	5	0

   Show that the decision $\hat{y}$ that minimizes the expected loss is equivalent to setting a probability threshold θ and predicting $\hat{y} = 0$ if $p_1 < \theta$ and $\hat{y} = 1$ if $p_1 \geq \theta$. What is this threshold for this loss matrix?

3. (Reject option – 15pts) In many applications, the classifier is allowed to "reject" a test example rather than classifying it into one of the classes. Consider, for example, a case in which the cost of misclassification is \$10 but the cost of having a human manually make the decision is only \$3. We can formulate this as the following loss matrix:

	$y = 0$ (true)	$y = 1$ (true)
$\hat{y} = 0$ (predicted)	0	10
$\hat{y} = 1$ (predicted)	10	0
reject	3	3

   Suppose $p(y = 1|\mathbf{x}) = 0.2$. Which decision minimizes the expected loss? Now suppose $p(y = 1|\mathbf{x}) = 0.4$. Now which decision minimizes the expected loss? Show that in cases such as this there will be two thresholds, θ_0 and θ_1, such that the optimal decision is to predict 0 if $p_1 < \theta_0$, reject if $\theta_0 \leq p_1 \leq \theta_1$, and predict 1 if $p_1 > \theta_1$.

   What are the values of these thresholds for the following loss matrix?

	$y = 0$ (true)	$y = 1$ (true)
$\hat{y} = 0$ (predicted)	0	10
$\hat{y} = 1$ (predicted)	5	0
reject	3	3

4. (Double counting the evidence – 30pts) Consider a problem in which the class label $y \in \{T, F\}$ and each training example X has 2 binary attributes $X_1, X_2 \in \{T, F\}$.

   Let the class prior be $p(Y = T) = 0.5$ and $p(X_1 = T|Y = T) = 0.8$ and $p(X_2 = T|Y = T) = 0.5$. Likewise, $p(X_1 = F|Y = F) = 0.7$ and $p(X_2 = F|Y = F) = 0.9$. Attribute X_1 provides slightly stronger evidence about the class label than X_2.

- Assume X_1 and X_2 are truly independent given Y. Write down the naive Bayes decision rule.
- What is the expected error rate of naive Bayes if it uses only attribute X_1? What if it uses only X_2?

  The expected error rate is the probability that each class generates an observation where the decision rule is incorrect. If Y is the true class label, let $\hat{Y}(X_1, X_2)$ be the predicted class label. Then the expected error rate is $p(X_1, X_2, Y | Y \neq \hat{Y}(X_1, X_2))$.
- Show that if naive Bayes uses both attributes, X_1 and X_2, the error rate is 0.235, which is better than if using only a single attribute (X_1 or X_2).
- Now suppose that we create new attribute X_3 which is an exact copy of X_2. So for every training example, attributes X_2 and X_3 have the same value. What is the expected error of naive Bayes now?
- Explain what is happening with naive Bayes? Does logistic regression suffer from the same problem? Explain why.

5. (Logistic regression – 35pts) Implement 2-class logistic regression in any language of your choice (Matlab, Java, Python, etc.). Run your code on any data set chosen from the UC Irvine Machine Learning Repository which can be found here:

   `http://archive.ics.uci.edu/ml/`

   To test your implementation, load the data set (either write throwaway code to do this, i.e., don't worry about making it generic, or "embed" the data into your program), randomize the order of the instances, and run 10-fold cross-validation over the data set (each time using one of the folds for testing and the rest for training). Turn in your code, the name of the data set you ran it on, the weight vector found when trained on the *entire* data set, and the classification accuracy on the test data averaged over the 10 folds.

   While you may use Weka to test your implementation (i.e., run it on the same data to confirm accuracy), the answers you submit must come from your implementation.

   For 5 points extra credit, generate a learning curve of your classifier for various sizes of the training set and plot it (using gnuplot, matlab, excel, etc). To generate the various training set sizes, choose 9 folds as the training data, randomize the order of the instances, and then train on the first $\{5, 10, 15, 20, \ldots\}$ instances, each time evaluating on the test fold. Then repeat nine more times, using each fold in turn as the test fold.