CIS 520, Machine Learning, Fall 2011: Assignment 3
Due: Wednesday, October 12th, 4pm, in Levine 459
[60 points]

Instructions. Please write up your responses to the following problems clearly and concisely. We encourage you to write up your responses using LaTeX; we will provide a short LaTeX template, available on Blackboard, to make this easier. **Hand in your assignment to Charity Payne in Levine 459. Write your name, the time at which you are handing in your homework, and the name(s) of any collaborator(s) on the front page of your problem set. If the office is closed, please slide the homeworks under the door.**

Collaboration. You are allowed and encouraged to work together. You may discuss the homework to understand the problem and reach a solution in groups up to size **four students.** However, *each student must write down the solution independently, and without referring to written notes from the joint session.* **In addition, each student must write on the problem set the set of people with whom s/he collaborated.** You must understand the solution well enough in order to reconstruct it by yourself. (This is for your own benefit: you have to take the exams alone.)

1 Linear Regression and LOOCV [17 points]

In the last homework, you learned about using cross validation as a way to estimate the true error of a learning algorithm. A solution that provides an almost unbiased estimate of this true error is *Leave-One-Out Cross Validation* (LOOCV), but it can take a really long time to compute the LOOCV error. In this problem, you will derive an algorithm for efficiently computing the LOOCV error for linear regression using the *Hat Matrix.* [1]

Assume that there are n given training examples, $(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)$, where each input data point X_i, has m real-valued features. The goal of regression is to learn to predict Y from X. The *linear* regression model assumes that the output Y is a weighted *linear* combination of the input features with weights given by $\mathbf{w}$, plus some Gaussian noise.

We can write this in matrix form by stacking the data points as the rows of a matrix X so that x_{ij} is the j-th feature of the i-th data point. Then writing Y, $\mathbf{w}$ and ϵ as column vectors, we can express the linear regression model in matrix form as follows:

$$Y = X\mathbf{w} + \epsilon$$

where:

$$Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix}, X = \begin{bmatrix} x_{11} & x_{12} & \ldots & x_{1m} \\ x_{21} & x_{22} & \ldots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \ldots & x_{nm} \end{bmatrix}, \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix}, \quad \text{and } \epsilon = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

[1] Unfortunately, such an efficient algorithm may not be easily found for other learning methods.

Assume that ϵ_i is normally distributed with variance σ^2. We saw in class that the maximum likelihood estimate of the model parameters $\mathbf{w}$ (which also happens to minimize the sum of squared prediction errors) is given by the *Normal equation*:

$$\hat{\mathbf{w}} = (X^T X)^{-1} X^T Y$$

Define $\hat{Y}$ to be the vector of predictions using $\hat{\mathbf{w}}$ if we were to plug in the original training set X:

$$\begin{aligned} \hat{Y} &= X\hat{\mathbf{w}} \\ &= X(X^T X)^{-1} X^T Y \\ &= HY \end{aligned}$$

where we define $H = X(X^T X)^{-1} X^T$ (H is often called the *Hat Matrix*).
As mentioned above, $\hat{\mathbf{w}}$, also minimizes the sum of squared errors:

$$\text{SSE} = \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2$$

Now recall that the Leave-One-Out Cross Validation score is defined to be:

$$\text{LOOCV} = \sum_{i=1}^{n} (Y_i - \hat{Y}_i^{(-i)})^2$$

where $\hat{Y}^{(-i)}$ is the estimator of Y after removing the i-th observation (i.e., it minimizes $\sum_{j \neq i} (Y_j - \hat{Y}_j^{(-i)})^2$).

1. **[2 points]** What is the complexity of computing the LOOCV score naively? (The naive algorithm is to loop through each point, performing a regression on the $n-1$ remaining points at each iteration.)

   Hint: The complexity of matrix inversion for a $k \times k$ matrix is $O(k^3)$. (There are faster algorithms out there but for simplicity we'll assume that we are using the naive $O(k^3)$ algorithm.)

2. **[2 points]** Write $\hat{Y}_i$ in terms of the elements of H and Y. You may find it useful to use shorthand such as H_{ab} to denote the entry in row a, column b of H.

3. **[3 points]** Show that $\hat{Y}^{(-i)}$ is also the estimator which minimizes SSE for Z where

   $$Z_j = \begin{cases} Y_j, & j \neq i \\ \hat{Y}_i^{(-i)}, & j = i \end{cases}$$

   Hint: Try to start by writing an expression for the SSE of Z; it should look very similar to the definition of SSE for Y that was given in the introduction section of this question. Then, manipulate terms until you can argue that substituting $\hat{Y}^{(-i)}$ for $\hat{Z}$ would minimize this expression.

4. **[2 points]** Write $\hat{Y}_i^{(-i)}$ in terms of H and Z. By definition, $\hat{Y}_i^{(-i)} = Z_i$, but give an answer that includes both H and Z.

5. **[4 points]** Show that $\hat{Y}_i - \hat{Y}_i^{(-i)} = H_{ii} Y_i - H_{ii} \hat{Y}_i^{(-i)}$, where H_{ii} denotes the i-th element along the diagonal of H.

6. **[4 points]** Show that

   $$LOOCV = \sum_{i=1}^{n} \left(\frac{Y_i - \hat{Y}_i}{1 - H_{ii}} \right)^2$$

   What is the algorithmic complexity of computing the LOOCV score using this formula?

   Note: We see from this formula that the diagonal elements of H somehow indicate the impact that each particular observation has on the result of the regression.

2 Logistic regression and Naive Bayes [8 points]

A common debate in machine learning has been over generative versus discriminative models for classification. In this question we will explore this issue by considering Naive Bayes and logistic regression.

1. [**2 points**] For input $\mathcal{X}$ and output Y, which of the following is the **objective function** optimized by (i) Naive Bayes, and (ii) logistic regression?
   (a) $\mathbf{Pr}(Y)/\mathbf{Pr}(\mathcal{X})$
   (b) $\mathbf{Pr}(\mathcal{X})/\mathbf{Pr}(Y)$
   (c) $\mathbf{Pr}(Y \mid \mathcal{X})$
   (d) $\mathbf{Pr}(Y)$
   (e) $\mathbf{Pr}(\mathcal{X})$
   (f) $\mathbf{Pr}(Y)\mathbf{Pr}(\mathcal{X})$
   (g) $\mathbf{Pr}(\mathcal{X}, Y)$
   (h) None of the above (provide the correct formula in this case)
2. [**6 points**] Recall from the suggested reading that "the discriminative analog of Naive Bayes is logistic regression." This means that the parametric form of $P(Y \mid X)$ used by logistic regression is implied by the assumptions of a Naive Bayes classifier, for some specific class-conditional densities. In class you will see how to prove this for a Gaussian Naive Bayes classifier for continuous input values. Can you prove the same for binary inputs? Assume X_i and Y are both binary. Assume that $X_i \mid Y = j$ is $\mathsf{Bernoulli}(\theta_{ij})$, where $j \in \{0, 1\}$, and Y is $\mathsf{Bernoulli}(\pi)$.

3 Double-counting the evidence [12 points]

1. [**1 points**] Consider the two class problem where class label $y \in \{T, F\}$ and each training example X has 2 binary attributes $X_1, X_2 \in \{T, F\}$. How many parameters will you *need* to know/evaluate if you are to classify an example using the Naive Bayes classifier? Keep in mind that since the probability of all possible events has to sum to 1, knowing the probabilities of all except one event implies knowledge of the final event's probability already. (Don't include such final events in your count.)
2. [**1 points**] Let the class prior be $\mathbf{Pr}(Y = T) = 0.5$ and also let $\mathbf{Pr}(X_1 = T \mid Y = T) = 0.8$, $\mathbf{Pr}(X_1 = F \mid Y = F) = 0.7$, $\mathbf{Pr}(X_2 = T \mid Y = T) = 0.5$, and $\mathbf{Pr}(X_2 = F \mid Y = F) = 0.9$. So, 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** given $X_1 = x_1$ and $X_2 = x_2$. Write your answer as a table listing the value of the decision, call it $f(X_1, X_2)$, for each of the 4 settings for X_1, X_2.
3. [**5 points**] For the Naive Bayes decision function $f(X_1, X_2)$, the error rate is:

$$\sum_{X_1, X_2, Y} \mathbf{1}\left(Y \neq f(X_1, X_2)\right) P(X_1, X_2, Y).$$

   For this question, we will assume that the true data distribution is exactly the same as the Naive Bayes distribution, so we can write $P(X_1, X_2, Y)$ as $P(Y)P(X_1 \mid Y)P(X_2 \mid Y)$.
   (a) [**2 points**] Show that if Naive Bayes uses both attributes, X_1 and X_2, the error rate is 0.235.
   (b) [**1 points**] What is the error rate using only X_1?
   (c) [**1 points**] What is the error rate using only X_2?
   (d) [**1 points**] Why is the error rate (choose one) lower/higher using X_1 and X_2 together as opposed to using only a single attribute?

4. **[3 points]** Now, suppose that we create a 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, $X_2 = X_3$.

   (a) **[1 points]** Are X_2 and X_3 conditionally independent given Y?

   (b) **[2 points]** What is the error rate of Naive Bayes now, using X_1, X_2, and X_3?

5. **[1 points]** Why does Naive Bayes perform worse with the addition of X_3? (*Hint*: What assumption does Naive Bayes make about the inputs?)

6. **[1 points]** Does logistic regression suffer from the same problem? Explain why or why not.

7. **[3 points] Extra credit**: In spite of the above fact we will see that in some examples Naive Bayes doesn't do too badly. Consider the above example i.e. your features are X_1, X_2 which are truely independent given Y and a third feature $X_3 = X_2$. Suppose you are now given an example with $X_1 = T$ and $X_2 = F$. You are also given the probabilities $\mathbf{Pr}(Y = T \mid X_1 = T) = p$ and $Pr(Y = T \mid X_2 = F) = q$, and $P(Y = T) = 0.5$.

   (a) Prove that the decision rule is $p \geq \frac{(1-q)^2}{q^2+(1-q)^2}$ by applying Bayes rule again.

   (b) What is the true decision rule?

   (c) Plot the two decision boundaries (vary q between 0 and 1) and highlight the region where Naive Bayes makes mistakes.

4 Boosting [23 points]

The details of Adaboost are in *Robert E. Schapire. The boosting approach to machine learning: An overview.* In Nonlinear Estimation and Classification. Springer, 2003. http://www.cs.princeton.edu/~schapire/uncompress-papers.cgi/msri.ps (The AdaBoost notation in Bishop's *Pattern Recognition* is slightly different and might be confusing when solving this question, so you should use the Schapire paper as your main reference here.)

4.1 Analyzing the training error of boosting [18 points]

Consider the AdaBoost algorithm you saw in class. In this question we will try to analyze the training error of boosting.

1. **[2 points]** Given a set of m examples, (x_i, y_i) (y_i is the class label of x_i), $i = 1, \ldots, m$, let $h_t(x)$ be the weak classifier obtained at step t, and let α_t be its weight. Recall that the final classifier is

$$H(x) = \text{sign}(f(x)), \text{ where } f(x) = \sum_{t=1}^{T} \alpha_t h_t(x).$$

   Show that the training error of the final classifier can be bounded from above by an exponential loss function:

$$\frac{1}{m}\sum_{i=1}^{m} I(H(x_i) \neq y_i) \leq \frac{1}{m}\sum_{i=1}^{m} \exp(-f(x_i)y_i),$$

   where $I(a = b)$ is the indicator function. It is equal to 1 if $a = b$, and 0 otherwise

   Hint: $e^{-x} \geq 1 \Leftrightarrow x \leq 0$.

2. **[6 points]** Remember that

$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Use this recursive definition to prove the following.

$$\frac{1}{m}\sum_{i=1}^{m}\exp(-f(x_i)y_i) = \prod_{t=1}^{T} Z_t, \tag{1}$$

where Z_t is the normalization factor for distribution D_{t+1}:

$$Z_t = \sum_{i=1}^{m} D_t(i)\exp(-\alpha_t y_i h_t(x_i)). \tag{2}$$

Hint: Remember that $e^{\sum_i g_i} = \prod_i e^{g_i}$,$D_1(i) = \frac{1}{m}$, and that $\sum_i D_{t+1}(i) = 1$.

3. [**3 points**] Equation 1 suggests that the training error can be reduced rapidly by greedily optimizing Z_t at each step. You have shown that the error is bounded from above:

$$\epsilon_{training} \leq \prod_{t=1}^{T} Z_t.$$

Observe that $Z_1, \ldots, Z_{t-1}$ are determined by the first $(t-1)$ rounds of boosting, and we cannot change them on round t. A greedy step we can take to minimize the training error bound on round t is to minimize Z_t.

In this question, you will prove that for binary weak classifiers, Z_t from Equation 2 is minimized by picking α_t as:

$$\alpha_t^* = \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right), \tag{3}$$

where ϵ_t is the training error of weak classifier h_t for the weighted dataset:

$$\epsilon_t = \sum_{i=1}^{m} D_t(i) I(h_t(x_i) \neq y_i).$$

where I is the indicator function. For this proof, only consider the simplest case of binary classifiers, i.e. the output of $h_t(x)$ is binary, $\{-1, +1\}$.

For this special class of classifiers, first show that the normalizer Z_t can be written as:

$$Z_t = (1-\epsilon_t)\exp(-\alpha_t) + \epsilon_t \exp(\alpha_t).$$

Hint: Consider the sums over correctly and incorrectly classified examples separately.

4. [**2 points**] Now, prove that the value of α_t that minimizes this definition of Z_t is given by Equation 3.

5. [**1 points**] Prove that for the above value of α_t we have $Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$.

6. [**2 points**] Furthermore, let $\epsilon_t = \frac{1}{2} - \gamma_t$ and prove that $Z_t \leq \exp(-2\gamma_t^2)$. Therefore, we will know

$$\epsilon_{training} \leq \prod_t Z_t \leq \exp(-2\sum_t \gamma_t^2).$$

Hint: $\log(1-x) \leq -x$ for $0 < x \leq 1$.

7. [**1 points**] If each weak classifier is slightly better than random, so that $\gamma_t \geq \gamma$, for some $\gamma > 0$, then the training error drops exponentially fast in T, i.e.

$$\epsilon_{training} \leq \exp(-2T\gamma^2).$$

Argue that in each round of boosting, there always exists a weak classifier h_t such that its training error on the weighted dataset $\epsilon_t \leq 0.5$.

8. [**1 points**] Show that for $\epsilon_t = 0.5$ the training error can get "stuck" above zero. *Hint*: $D_t(i)$s may get stuck.

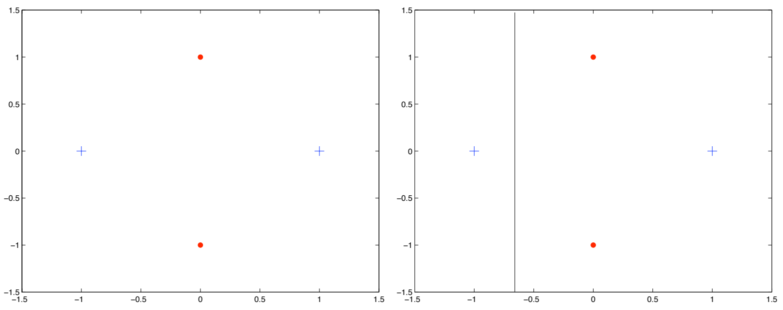

Figure 1: a) Toy data in Question 4. b) h_1 in Question 4

4.2 Adaboost on a toy dataset [5 points]

Now we will apply Adaboost to classify a toy dataset. Consider the dataset shown in Figure 1a. The dataset consists of 4 points: $(X_1 : 0, -1, -)$, $(X_2 : 1, 0, +)$, $(X_3 : -1, 0, +)$ and $(X_4 : 0, 1, -)$. For this part, you may find it helpful to use MATLAB as a calculator rather than doing the computations by hand. (You do not need to submit any code though.)

1. **[2 points]** For $T = 4$, show how Adaboost works for this dataset, using simple decision stumps (depth-1 decision trees that simply split on a single variable once) as weak classifiers. For each timestep compute the following:

$$\epsilon_t, \alpha_t, Z_t, D_t(i) \ \forall i,$$

   Also for each timestep draw your weak classifier. For example h_1 can be as shown in 1b).

2. **[1 points]** What is the training error of Adaboost for this toy dataset?

3. **[2 points]** Is the above dataset linearly separable? Explain why Adaboost does better than a decision stump on the above dataset.