CIS 520, Machine Learning, Fall 2015: Assignment 3
Solutions

Instructions. Please write up your responses to the following problems clearly and concisely. We encourage you to write up your responses using LaTeX; we have provided a LaTeX template, available on Canvas, to make this easier. **Submit your answers in PDF form to Canvas. We will not accept paper copies of the homework.**

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 **two 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 names of the people with whom you 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 [30 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:

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

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*:

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

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

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

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

where we define $H = X(X^TX)^{-1}X^T$ (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]** To begin with, we should consider when it is possible to compute $\hat{\mathbf{w}}$ in this framework.

   (a) **[1 point]** Suppose $m > n$. Is $\hat{\mathbf{w}}$ well-defined? Why or why not?
   Hint: Recall that the rank of a matrix is equal to the number of linearly independent rows, which is also equal to the number of linearly independent columns. Use the fact that for two matrices A and B which can be multiplied to form the product AB, it must be the case that $\text{rank}(AB) \leq \min(\text{rank}(A), \text{rank}(B))$. Furthermore, recall that a square matrix is invertible if and only if it is full-rank.

   (b) **[1 point]** Suppose $m \leq n$. Give a condition on X which guarantees that $\hat{\mathbf{w}}$ will **not** be well-defined and explain why not. (Don't assume X is a square matrix.)

   For the rest of question 1, assume $\hat{\mathbf{w}}$ is well-defined.

   ★ **SOLUTION:** a) No. We use the rule from linear algebra that

   $$\mathsf{rank}(AB) \leq \min(\mathsf{rank}(A), \mathsf{rank}(B)).$$

   The rank of a matrix is not greater than its smallest dimension, in this case n. X^T has the same rank as X. Therefore,

   $$\mathsf{rank}(X^TX) \leq \mathsf{rank}(X) \leq n < m.$$

   Therefore, X^TX is not full-rank, and so it is not invertible. It follows that $\hat{\mathbf{w}}$ is not well-defined.

   b) Two or more columns of X are linearly dependent, OR the rank of X is less than m, OR similar. The key relation is again

   $$\mathsf{rank}(X^TX) \leq \mathsf{rank}(X),$$

   plus the fact that X^TX can be inverted only if it is full-rank. It is not correct to just say that X is singular, since this assumes X is square. It is not correct to give the condition that two or more *rows* of X are linearly dependent, since this is guaranteed for the case $m < n$.

2. **[3 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.)

★ **SOLUTION:** Let $X^{(-i)}$ denote X without the i-th example (row), and let $Y^{(-i)}$ be Y without the i-th element. For each point, we need to compute a $\hat{\mathbf{w}}^{(-i)}$ ($\hat{\mathbf{w}}$ computed from $X^{(-i)}$ and $Y^{(-i)}$) and dot it with a single point X_i. Note that it is not necessary to compute the hat matrix H each time, since H is designed to give predictions for all examples and we only need the prediction for X_i.

The expression for $\hat{\mathbf{w}}^{(-i)}$ is:

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

Recall that multiplying a $n \times m$ matrix by a $m \times p$ matrix costs $O(nmp)$ flops (operations, in this case multiplication and addition). Forming $X^T X$ takes $m^2 n$ flops. Inverting it takes m^3 flops. Multiplying by X^T is $m^2 n$ flops, and multiplying by Y is nm flops.

We need to multiply this by n since LOOCV loops through each training point - doing so gives $O(m^2 n^2 + nm^3)$ complexity.

1 point for using $\hat{w}$, 2 point for the correct expression.

3. **[3 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.

★ **SOLUTION:** $\hat{Y}_i = \sum_{j=1}^{n} H_{ij} Y_j$

4. **[4 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.

★ **SOLUTION:** We want to show that the estimator $\hat{Z}_j$ that minimizes SSE for Z is equal to $\hat{Y}^{(-i)}$.

$$\begin{aligned} SSE(Z) &= \sum_{j=1}^{n} (Z_j - \hat{Z}_j)^2 \\ &= (Z_i - \hat{Z}_i)^2 + \sum_{j \neq i} (Z_j - \hat{Z}_j)^2 \\ &= (\hat{Y}_i^{(-i)} - \hat{Z}_i)^2 + \sum_{j \neq i} (Y_j - \hat{Z}_j)^2 \end{aligned}$$

Consider minimizing each of the two terms separately. Using that $\hat{Y}^{(-i)}$ minimizes $\sum_{j \neq i} (Y_j - \hat{Y}_j^{(-i)})^2$, we see that $\hat{Y}^{(-i)}$ minimizes the right hand term. And furthermore, the choice $Z_i = \hat{Y}_i^{(-i)}$ makes the left hand term zero and therefore minimizes it.

So $\hat{Y}^{(-i)}$ is also the estimator that minimizes $SSE(Z)$. This is certainly a valid estimator, i.e. we can obtain it with linear regression, because the model $w^{(-i)}$ which produces the $n-1$ terms of $\hat{Y}^{(-i)}$ in the sum is the same model which predicts $\hat{Y}_i^{(-i)}$ from the remaining observation X_i.

5. **[6 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.

★ **SOLUTION:** Since $\hat{Y}^{(-i)}$ is the min-SSE estimator for Z as $\hat{Y}$ is for Y, we can write it (see 1.3) as

$$\hat{Y}_i^{(-i)} = \sum_{j=1}^{n} H_{ij} Z_j.$$

Big picture: how would you think to come up with this? Possible answer: we have two expressions for the LOOCV, one given in the intro to the question, and one which we derive finally in 1.7 (the last part of this question). To get from the original formula to the 1.7 formula, we remove the estimates for each observation predicted by each of the n regressions with $n-1$ observations, and we insert estimates predicted from just one regression and values from one hat matrix computation. So, it turns out that the predictions from each of the n regressions are completely encoded in the hat matrix and the single regression with n observations, and it turns out that the trade-off in computational complexity is favorable.

1.5 is sort of the breakthrough moment where we can write the predictions from each of the n smaller regressions completely in terms of things we know without doing those n regressions. So if you were trying to figure out how to do this from the beginning, you might be trying to find exactly this.

6. **[6 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.
Hint: Use the results from part 2 and 4. Substitute Z_i with Y_i and $\hat{Y}_i^{-i}$ by using its definition in part 3.

★ **SOLUTION:** From parts (2) and (4), we can write

$$\begin{aligned}
\hat{Y}_i - \hat{Y}_i^{(-i)} &= \sum_{j=1}^{n} H_{ij} Y_j - \sum_{j=1}^{n} H_{ij} Z_j \\
&= \sum_{j \neq i} H_{ij} Y_j + H_{ii} Y_i - \sum_{j \neq i} H_{ij} Z_j - H_{ii} Z_i \\
&= \sum_{j \neq i} H_{ij} Y_j + H_{ii} Y_i - \sum_{j \neq i} H_{ij} Y_j - H_{ii} \hat{Y}_i^{(-i)} \\
&= H_{ii} Y_i - H_{ii} \hat{Y}_i^{(-i)}
\end{aligned}$$

7. **[6 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.

★ **SOLUTION:** Part (**??**) says that $\hat{Y}_i - \hat{Y}_i^{(-i)} = H_{ii} Y_i - H_{ii} \hat{Y}_i^{(-i)}$. We can rearrange this to

$$\begin{aligned}
\hat{Y}_i^{(-i)} - H_{ii} \hat{Y}_i^{(-i)} &= \hat{Y}_i - H_{ii} Y_i \\
\hat{Y}_i^{(-i)} &= \frac{\hat{Y}_i - H_{ii} Y_i}{1 - H_{ii}}
\end{aligned}$$

Now the LOOCV error becomes

$$\begin{aligned} \mathit{LOOCV} &= \sum_{i=1}^{n} (Y_i - \hat{Y}_i^{(-i)})^2 \\ &= \sum_i \left(Y_i - \frac{\hat{Y}_i - H_{ii} Y_i}{1 - H_{ii}} \right)^2 \\ &= \sum_i \left(\frac{(1 - H_{ii}) Y_i - \hat{Y}_i + H_{ii} Y_i}{1 - H_{ii}} \right)^2 \\ &= \sum_i \left(\frac{Y_i - \hat{Y}_i}{1 - H_{ii}} \right)^2 \end{aligned}$$

Evaluating LOOCV using this formula can be done in $O(m^2 n + m^3)$ time since it only requires one regression (and we don't need to compute all of H to compute H_{ii}).

3 points for showing the LOOCV equation, 3 points for the correct complexity.

2 Logistic regression and Naive Bayes [20 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. **[4 points]** For input X and output Y, which of the following is the **objective function** optimized by (i) Naive Bayes, and (ii) logistic regression?

   (a) $\mathbf{Pr}(Y)/\mathbf{Pr}(X)$
   (b) $\mathbf{Pr}(X)/\mathbf{Pr}(Y)$
   (c) $\mathbf{Pr}(Y \mid X)$
   (d) $\mathbf{Pr}(Y)$
   (e) $\mathbf{Pr}(X)$
   (f) $\mathbf{Pr}(Y)\mathbf{Pr}(X)$
   (g) $\mathbf{Pr}(X, Y)$
   (h) None of the above (provide the correct formula in this case)

   ★ **SOLUTION:** (i) **??**, (ii) **??**.

2. **[16 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 Bernoulli(θ_{ij}), where $j \in \{0, 1\}$, and Y is Bernoulli(π).
   Hint: Start by using Bayes Rule and the assumptions of Naive Bayes to express the objective function for logistic regression in terms of the given quantities θ_{ij} and π.

★ **SOLUTION:** Our goal in this problem is to transform the Naive Bayes class conditional probability given in **??** into the form of the logistic regression class conditional probability given in **??**.

$$\mathbf{Pr}_{NB}(Y=1 \mid \mathcal{X}) = \frac{\mathbf{Pr}(Y=1)\prod_{i=1}^{n}\mathbf{Pr}(X_i \mid Y=1)}{\mathbf{Pr}(Y=0)\prod_{i=1}^{n}\mathbf{Pr}(X_i \mid Y=0)+\mathbf{Pr}(Y=1)\prod_{i=1}^{n}\mathbf{Pr}(X_i \mid Y=1)} \quad (1)$$

$$\mathbf{Pr}_{LR}(Y=1 \mid \mathcal{X}) = \frac{1}{1+\exp(-w_0-\sum_{i=1}^{n}w_iX_i)} \quad (2)$$

We begin by applying a few basic algebraic transformations to **??** to obtain **??**.

$$\begin{aligned}\mathbf{Pr}_{NB}(Y=1 \mid \mathcal{X}) &= \frac{1}{1+\frac{\mathbf{Pr}(Y=0)\prod_{i=1}^{n}\mathbf{Pr}(X_i|Y=0)}{\mathbf{Pr}(Y=1)\prod_{i=1}^{n}\mathbf{Pr}(X_i|Y=1)}} \\ &= \frac{1}{1+\exp\left(\ln\left(\frac{\mathbf{Pr}(Y=0)\prod_{i=1}^{n}\mathbf{Pr}(X_i|Y=0)}{\mathbf{Pr}(Y=1)\prod_{i=1}^{n}\mathbf{Pr}(X_i|Y=1)}\right)\right)} \\ &= \frac{1}{1+\exp\left(\ln\left(\frac{\mathbf{Pr}(Y=0)}{\mathbf{Pr}(Y=1)}\right)+\sum_{i=1}^{n}\ln\left(\frac{\mathbf{Pr}(X_i|Y=0)}{\mathbf{Pr}(X_i|Y=1)}\right)\right)} \end{aligned} \quad (3)$$

The probability mass functions of the Bernoulli distribution for of Y and $(X_i \mid Y=j)$ are given by:

$$\mathbf{Pr}(Y=\alpha)=\pi^{\alpha}(1-\pi)^{1-\alpha} \qquad \text{and} \qquad Pr(X_i=\beta \mid Y=j)=\theta_{i,j}^{\beta}(1-\theta_{i,j})^{1-\beta}.$$

Therefore we can rewrite **??** as **??** by substituting the above probability mass functions.

$$\mathbf{Pr}_{NB}(Y=1 \mid \mathcal{X}) = \frac{1}{1+\exp\left(\ln\left(\frac{1-\pi}{\pi}\right)+\sum_{i=1}^{n}\ln\left(\frac{\theta_{i0}^{X_i}(1-\theta_{i0})^{(1-X_i)}}{\theta_{i1}^{X_i}(1-\theta_{i1})^{(1-X_i)}}\right)\right)} \quad (4)$$

With some more algebraic manipulation we can transform **??** into **??**.

$$\begin{aligned}\mathbf{Pr}_{NB}(Y=1 \mid X) &= \frac{1}{1+\exp\left(\ln\left(\frac{1-\pi}{\pi}\right)+\sum_{i=1}^{n}X_i\ln\left(\frac{\theta_{i0}}{\theta_{i1}}\right)+(1-X_i)\ln\left(\frac{1-\theta_{i0}}{1-\theta_{i1}}\right)\right)} \\ &= \frac{1}{1+\exp\left(\ln\left(\frac{1-\pi}{\pi}\right)+\sum_{i=1}^{n}\ln\left(\frac{1-\theta_{i0}}{1-\theta_{i1}}\right)+\sum_{i=1}^{n}X_i\ln\left(\frac{\theta_{i0}(1-\theta_{i1})}{\theta_{i1}(1-\theta_{i0})}\right)\right)} \end{aligned} \quad (5)$$

We now recognize that if we define w_0 and w_i as in **??** and **??** we can write **??** as **??**.

$$w_0 = -\ln\left(\frac{1-\pi}{\pi}\right)-\sum_{i=1}^{n}\ln\left(\frac{1-\theta_{i0}}{1-\theta_{i1}}\right) \quad (6)$$

$$w_i = -\ln\left(\frac{\theta_{i0}(1-\theta_{i1})}{\theta_{i1}(1-\theta_{i0})}\right) \quad (7)$$

We may therefore conclude that the assumptions made by the Naive Bayes classifier over binary random variables imply a decision rule that has the parametric form of logistic regression.

■ **COMMON MISTAKE 1:** Incorrect expressions for the conditional probabilities $P(X|Y)$.

■ **COMMON MISTAKE 2:** Forgetting that logistic regression has the form $\frac{1}{1+exp\{-\mathbf{w}^T\mathbf{x}\}}$, so that w_0 and w_i need to be negated.

3 Double-counting the evidence [30 points]

1. **[2 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.)

   ★ **SOLUTION:** The Naive Bayes classifier learns the the conditional probabilities $\mathbf{Pr}(X_1 \mid Y)$ and $\mathbf{Pr}(X_2 \mid Y)$ as well as the class probability $\mathbf{Pr}(Y)$. To represent $\mathbf{Pr}(Y)$ we need only one parameter π because $\mathbf{Pr}(Y = T) + \mathbf{Pr}(Y = F) = 1$. The $\mathbf{Pr}(X_i \mid Y)$ can be represented using only $\mathbf{Pr}(X_i = T \mid Y = T) = \theta_{i1}$ and $\mathbf{Pr}(X_i = T \mid Y = F) = \theta_{i0}$. We do not need additional parameters to represent $\mathbf{Pr}(X_i = F \mid Y = F) = 1 - \theta_{i0}$ or $\mathbf{Pr}(X_i = F \mid Y = T) = 1 - \theta_{i1}$. Therefore we need 1 parameter for $\mathbf{Pr}(Y)$, 2 parameters for $\mathbf{Pr}(X_1|Y)$, and 2 parameters for $\mathbf{Pr}(X_2|Y)$ resulting in a total of *5* parameters.

2. **[2 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 $\mathbf{Pr}(X_2 = F \mid Y = F) = 0.9$. (*Note:* Questions 3.2 - 3.4 all use these probabilities.) 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.

   ★ **SOLUTION:** The mathematical form of the decision rule for the Naive Bayes classifier is given in **??** where $\mathbf{1}(\cdot)$ is the indicator function. Notice that in **??** the sum of log ratios is used rather than a direct comparison of $\mathbf{Pr}(Y = T \mid \mathcal{X}) > \mathbf{Pr}(Y = F \mid \mathcal{X})$. While both are equivalent the sum of logs form is often more numerically stable and easier to work with.

$$f(X_1, X_2) = \mathbf{1}\left(\ln\left(\frac{\mathbf{Pr}(Y=1)}{\mathbf{Pr}(Y=0)}\right) + \sum_{i=1}^{n} \ln\left(\frac{\mathbf{Pr}(X_i \mid Y=1)}{\mathbf{Pr}(X_i \mid Y=0)}\right) > 0\right) \tag{8}$$

X_1	X_2	Y	$\mathbf{Pr}(X_1, X_2, Y)$	$\mathbf{Pr}_{NB}(Y \mid X_1, X_2)$	$f(X_1, X_2)$
0	0	0	0.315	0.863014	0
0	0	1	0.05	0.136986	0
0	1	0	0.035	0.411765	1
0	1	1	0.05	0.588235	1
1	0	0	0.135	0.402985	1
1	0	1	0.2	0.597015	1
1	1	0	0.015	0.0697674	1
1	1	1	0.2	0.930233	1

Table 1: Decision rule for the Naive Bayes classifier. The column containing $f(X_1, X_2)$ contains the actual decision rule. For simplicity I have also included the joint and conditional probabilities as $\mathbf{Pr}(X_1, X_2, Y)$ and $\mathbf{Pr}(Y \mid X_1, X_2)$ respectively. Joint probabilties corresponding to incorrect predictions have been colored yellow.

An alternative and for this problem more informative representation of the decision rule for binary variables is given in **??**.

3. **[8 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.

★ **SOLUTION:** To compute the error rate using X_1 and X_2 we sum $\mathbf{Pr}(X_1, X_2, Y)$ for all the rows where $f(X_1, X_2) \neq Y$ in **??**, obtaining $0.05 + 0.035 + 0.135 + 0.015 = 0.235$.

(b) **[2 points]** What is the error rate using only X_1?

★ **SOLUTION:** If we consider only X_1 we get the following decision table:

X_1	Y	$\sum_{x_2=0}^{1} \mathbf{Pr}(X_1, x_2, Y)$	$\mathbf{Pr}_{NB}(Y \mid X_1)$	$f(X_1)$
0	0	0.35	0.777778	0
0	1	0.1	0.222222	0
1	0	0.15	0.272727	1
1	1	0.4	0.727273	1

and an error rate of $0.1 + 0.15 = 0.25$ which is greater than the original classifier $f(X_1, X_2)$.

(c) **[2 points]** What is the error rate using only X_2?

★ **SOLUTION:** If we use only X_2 we obtain the decision table:

X_2	Y	$\sum_{x_1=0}^{1} \mathbf{Pr}(x_1, X_2, Y)$	$\mathbf{Pr}_{NB}(Y \mid X_2)$	$f(X_2)$
0	0	0.45	0.642857	0
0	1	0.25	0.357143	0
1	0	0.05	0.166667	1
1	1	0.25	0.833333	1

and an error rate of $0.25 + 0.05 = 0.30$ which is greater than the original classifier $f(X_1, X_2)$.

(d) **[2 points]** Give a conceptual explanation for why the error rate is (choose one) lower/higher using X_1 and X_2 together as opposed to using only a single attribute.

★ **SOLUTION:** We see that a Naive Bayes classifier that uses both variables performs better than a Naive Bayes classifier that uses only one of the two. This implies that there is unique (uncorrelated) information in each variable.

4. **[5 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) **[2 points]** Are X_2 and X_3 conditionally independent given Y?

★ **SOLUTION:** First lets consider whether X_2 and X_3 are conditionally independent given Y. However this is quite simple as knowing the value of X_2 completely determines the value of X_3. Therefore X_2 and X_3 are clear *NOT* conditionally independent given Y.

(b) **[3 points]** What is the error rate of Naive Bayes now, using X_1, X_2, and X_3? The predicted Y should be computed using the (possibly incorrect) assumption of conditional independence, and the error rate should be computed using the true probabilities.

★ **SOLUTION:** To work out the error rate of the Naive Bayes classifier we construct another table:

X_1	X_2	X_3	Y	$\mathbf{Pr}(X_1, X_2, X_3, Y)$	$\mathbf{Pr}_{NB}(Y \mid X_1, X_2, X_3)$	$f(X_1, X_2, X_3)$
0	0	0	0	0.315	0.918963	0
0	0	0	1	0.05	0.0810373	0
0	0	1	0	0	0.557522	0
0	0	1	1	0	0.442478	0
0	1	0	0	0	0.557522	0
0	1	0	1	0	0.442478	0
0	1	1	0	0.035	0.122807	1
0	1	1	1	0.05	0.877193	1
1	0	0	0	0.135	0.548533	0
1	0	0	1	0.2	0.451467	0
1	0	1	0	0	0.118943	1
1	0	1	1	0	0.881057	1
1	1	0	0	0	0.118943	1
1	1	0	1	0	0.881057	1
1	1	1	0	0.015	0.0147783	1
1	1	1	1	0.2	0.985222	1

The error rate is obtained by summing the probability of all events in which the classifier predicts the wrong value for Y which is $0.05 + 0.035 + 0.20 + 0.015 = 0.3$. The astute reader will notice that all events where $X_2 \neq X_3$ occur with zero probability and could have be skipped in the table. The only thing that has changed by introducing X_3 is that for $X_1 = 1$ and $X_2 = 0$ the new Naive Bayes classifier using three variables predicts 0 rather than 1. Consequently, the error sum contains 0.20 rather than 0.135 resulting in a higher error rate. We see that the Naive Bayes classifier performs worse when we add highly correlated (uninformative) features. Naturally, we would not expect Naive Bayes to perform better by simply replicating existing features.

Another, simpler way of looking at the solution is as follows. We can imagine sampling (X_1, X_2, X_3, Y) in a 2 step process: first, (X_1, X_2, Y) is sampled according to P. Then X_3 is deterministically assigned the value of X_2. Thus, $P(X_1, X_2, X_3, Y \mid X_2 = X_3) = P(X_1, X_2, Y)$. In other words, introducing X_3 does *not* change the true underlying distribution of the data when considering X_1 and X_2. What does change? The decision rule changes, so that effectively X_2 is the only variable used. We already computed the error rate of X_2, which is 0.3.

■ **COMMON MISTAKE 1:** Many did not see that the "true" distribution no longer follows the conditional independence assumption following the introduction of X_3, and found that the error rate decreased.

5. **[2 points]** Why does Naive Bayes perform worse with the addition of X_3? (*Hint*: What assumption does Naive Bayes make about the inputs?)

★ **SOLUTION:** Naive Bayes performs worse because the addition of X_3 breaks the conditional independence assumption while not introducing any additional information. As a consequence the classifier over counts X_2 partially ignoring X_1.

6. **[3 points]** Does logistic regression suffer from the same problem? Explain why or why not.

★ **SOLUTION:** Logistic regression does not use the same conditional independence assumptions and therefore does not suffer when highly correlated features. An optimal logistic regression classifier would set $w_2' = w_3' = w_2/2$ where w_2 is the coefficient of X_2 obtained by training only on X_1 and X_2 and w_2' and w_3' are the coefficients obtained by training on X_1, X_2, and X_3.

7. **[8 points]** : 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 $P(Y = T) = 0.5$. (*Note:* You should **not** use the probabilities from 3.2-3.4 in your solutions to the following.)

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

   ★ **SOLUTION:** The decision rule for Naive Bayes is: Predict $Y = T$ if

$$\mathbf{Pr}(Y = T) \prod_i \mathbf{Pr}(X_i = x_i \mid Y = T) \geq \mathbf{Pr}(Y = F) \prod_i \mathbf{Pr}(X_i = x_i \mid Y = F)$$

   But we know that $\mathbf{Pr}(Y = T) = \mathbf{Pr}(Y = F)$

$$\prod_i \mathbf{Pr}(X_i = x_i \mid Y = T) \geq \prod_i \mathbf{Pr}(X_i = x_i \mid Y = F)$$

   Using Bayes rule again we get,

$$\prod_i \frac{\mathbf{Pr}(Y = T|X_i = x_i)\mathbf{Pr}(Y = T)}{\mathbf{Pr}(X_i = x_i)} \geq \prod_i \frac{\mathbf{Pr}(Y = F|X_i = x_i)\mathbf{Pr}(Y = F)}{\mathbf{Pr}(X_i = x_i)}$$

   Now, the $\prod_i \mathbf{Pr}(X_i = x_i)$ parts cancel from both sides. Also, we again note $\mathbf{Pr}(Y = T) = \mathbf{Pr}(Y = F)$ and therefore remove $\mathbf{Pr}(Y = T)$ from both side. Hence we now have

$$\prod_i \mathbf{Pr}(Y = T|X_i = x_i) \geq \prod_i \mathbf{Pr}(Y = F|X_i = x_i)$$

   Setting $x_1 = T$ and $x_3 = x_2 = F$, we find :

$$\begin{aligned} &\mathbf{Pr}(Y = T \mid X_1 = T)\mathbf{Pr}(Y = T \mid X_2 = F)\mathbf{Pr}(Y = T \mid X_3 = F) \\ \geq\ &\mathbf{Pr}(Y = F \mid X_1 = T)\mathbf{Pr}(Y = F \mid X_2 = F)\mathbf{Pr}(Y = F \mid X_3 = F) \end{aligned}$$

   Plugging in the values given in question we now obtain:

$$pq^2 \geq (1-p)(1-q)^2$$

   Thus the "Naive Bayes" decision rule is given by,

$$\text{Predict Y=T} \iff p \geq \frac{(1-q)^2}{q^2 + (1-q)^2}$$

   (b) What is the true decision rule?

★ **SOLUTION:** The actual decision rule doesn't take into consideration X_3, as we know that X_1 and X_2 are truly independent given Y. Therefore, we predict true if and only if $\mathbf{Pr}(Y = T \mid X_1 = T, X_2 = F) \geq \mathbf{Pr}(Y = F \mid X1_T, X2_F)$.
Following similar calculations as before, we have

$$pq \geq (1-p)(1-q)$$
$$p \geq 1 - q$$

Thus the "real" decision rule is given by,

$$\text{Predict Y=T} \iff p \geq 1 - q$$

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

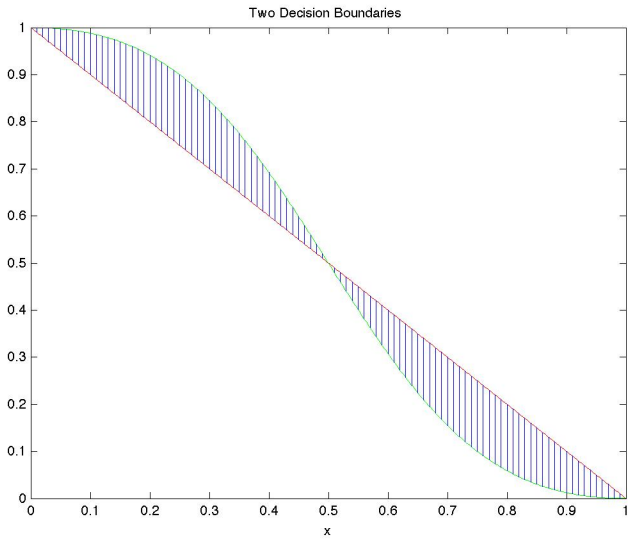

Figure 1: True and Naive Bayes decision boundary

★ **SOLUTION:** The curved line is for Naive Bayes, and the straight line shows the true decision boundary. The shaded part in **??** shows the region where Naive Bayes decision differs from the true decision.

4 Feature Selection [20 points]

We saw in class that one can use a variety of regularization penalties in linear regression.

$$\hat{w} = \arg\min_{w} \quad \|Y - Xw\|_2^2 + \lambda \|w\|_p^p$$

Consider the three cases, $p = 0$, 1, and 2. We want to know what effect these different penalties have on estimates of w.

Let's see this using a simple problem. Use the provided data (data.mat). Assume the constant term in the regression is zero, and assume $\lambda = 1$, except, of course, for question (1). You don't need to write code

that solves these problems in their full generality; instead, feel free to use matlab functions to do the main calculations, and then just do a primitive search over parameter space by plugging in a few different values. Matlab function fminsearch will be helpful. (*Note:* If you are not familiar with function handles, please review the code from HW 2 or see Matlab documentation.)

1. **[3 points]** If we assume that the response variable y is distributed according to $y \sim N(w \cdot x, \sigma^2)$, then what is the MLE estimate $\hat{w}_{MLE}$ of w?

   ★ **SOLUTION:** Let $r = (Y - Xw)$.

   $$\begin{aligned} \frac{\partial r^T r}{\partial w} &= -X^T(Y - Xw) = 0 \\ w &= (X^T X)^{-1} X^T Y \\ w &= [0.8891, -0.8260, 4.1902] \end{aligned}$$

2. **[2 points]** Given $\lambda = 1$, what is $\hat{w}$ for $p = 2$?

   ★ **SOLUTION:** The closed form solution is $w = (X^T X + \lambda I)^{-1} X^T Y$
   We use *fminsearch* in MATLAB to solve for the solution.
   $w = [0.8646, -0.8210, 4.1219]$

3. **[2 points]** Given $\lambda = 1$, what is $\hat{w}$ for $p = 1$?

   ★ **SOLUTION:** We can use either *lasso* or *fminsearch* in MATLAB.
   Note that *lasso* in Matlab doesn't minimize the same objective function.

	OBJ1 (fminsearch)	OBJ1 (lasso)
dataset	$w = [0.8749, -0.8182, 4.1829]$	$w = [0, -0.2755, 4.1902]$

4. **[4 points]** Given $\lambda = 1$, what is $\hat{w}$ for $p = 0$? Note that since L0 norm is not a "real" norm, the penalty expression is a little different:

   $$\hat{w} = \arg\min_w \quad \|Y - Xw\|_2^2 + \lambda \|w\|_0$$

   Also for L0 norm, you have to solve all combinatorial cases separately where some certain components of w are set to zero, then add L0 accordingly. There are 8 cases for 3 unknown w_i.

   ★ **SOLUTION:** For L0 norm, we have to solve all combinatorial cases separately where some certain components of w are set to zero, then add L0 accordingly. There are 8 cases for 3 unknown w_i.

   $$w = [0.8891, -0.8260, 4.1902]$$

5. **[4 points]** Write a paragraph describing the relation between the estimates of w in the four cases, explaining why that makes sense given the different penalties.

★ **SOLUTION:** The MLE estimate simply tries to minimize the residual error without enforcing any prior beliefs about w. Regularizing w under different norms corresponds to different prior beliefs on how we expect the w to be. The $L2$ norm corresponds to the belief that parameters w follow a Gaussian distribution with mean 0; this will tend to shrink all the weights by a constant factor. None will be zeroed out. Penalizing the $L1$ norm decreases all the parameters w_1, w_2, w_3 toward zero by a constant additive amount. If the parameters would be pushed beyond zero,they are zeroed out. In this problem, two were set to zero. Finally, the $L0$ norm encourages a sparse solution, but does not shrink those parameters that are not zeroed out. Thus the nonzero coefficients are larger than under $L1$ or $L2$.

6. **[5 points]** When $\lambda > 0$, we make a trade-off between minimizing the sum of squared errors and the magnitude of $\hat{w}$. In the following questions, we will explore this trade-off further. For the following, use the same data from data.mat.

   (a) **[1 point]** For the MLE estimate of w (as in 4.1), write down the value of the ratio

   $$||\hat{w}_{MLE}||_2^2 \; / \; ||Y - X\hat{w}_{MLE}||_2^2.$$

   ★ **SOLUTION:** $||\hat{w}_{MLE}||_2^2 \; / \; ||Y - X\hat{w}_{MLE}||_2^2 \approx 19.03/3104.8 \approx 0.0061$.

   (b) i. **[1 point]** Suppose the assumptions of linear regression are satisfied. Let's say that with N training samples (assume $N >> P$, where P is the number of features), you compute $\hat{w}_{MLE}$. Then let's say you do the same, this time with $2N$ training samples. How do you expect $||Y - X\hat{w}_{MLE}||_2^2$ to change when going from N to $2N$ samples? When $N >> P$, does this sum of squared errors for linear regression directly depend on the number of training samples?

   ★ **SOLUTION:** The SSE will approximately double when N is doubled. Since the assumptions of linear regression are satisfied, doubling the amount of training data will not dramatically change the model when $N >> P$, so we expect approximately twice the SSE with twice the number of summands. Yes, SSE depends directly on N since SSE is a sum over N squared error terms (one for each training sample).

   ii. **[1 point]** Likewise, if you double the number of training samples, how do you expect $||\hat{w}_{MLE}||_2^2$ to change? Does $||\hat{w}_{MLE}||_2^2$ for linear regression directly depend on the number of training samples in the large-N limit?

   ★ **SOLUTION:** In the large N limit, we expect $||\hat{w}_{MLE}||_2^2$ to change barely at all when N is doubled. No, $||\hat{w}_{MLE}||_2^2$ does not depend directly on N since it is a sum over the P elements of the vector $\hat{w}_{MLE}$.

   (c) **[1 point]** Using any method (e.g. trial and error, random search, etc.), find a value of λ for which the estimate $\hat{w}$ satisfies

   $$0.8 < ||\hat{w}||_2^2 \; / \; ||\hat{w}_{MLE}||_2^2 < 0.9.$$

   ★ **SOLUTION:** Any λ between 3.25 and 7.10.

   (d) **[1 point]** Using any method (e.g. trial and error, random search, etc.), find a value of λ for which the estimate $\hat{w}$ satisfies

   $$0.4 < ||\hat{w}||_2^2 \; / \; ||\hat{w}_{MLE}||_2^2 < 0.5.$$

★ **SOLUTION:** Any λ between 25.1 and 35.3.