**CSCI 5525 - Assignment 1**

Ans 1.

$$E_{(x,y)}[l(f(x), y)]$$

$$\Rightarrow \int_x \left\{ \int_y l(f(x), y)\, p(y|x)\, dy \right\} p(x)\, dx.$$

$$\Rightarrow \int_x \left\{ \int_y [y - f(x)]^2\, p(y|x)\, dy \right\} p(x)\, dx$$

to minimize the functional $E_{(x,y)}$ w.r.t $f(x)$, we need to do

$$\frac{\delta\, \mathbb{E}_{x,y}[l(f(x), y)]}{\delta f(x)} = 0$$

Here it is better to think of $f(x)$ as a variable $C$.

$$\frac{\delta\, \mathbb{E}_{x,y}[\quad]}{\delta C} = 0$$

from the calculus of variations, we have for a functional that is defined by an integral over a function $G(y, y', x)$

$$F(y) = \int G(y(x), y'(x), x)\, dx$$

$\Rightarrow$ and $\boxed{\delta(F) = 0}$ for these functionals is equivalent to doing

$$\frac{\partial G}{\partial y} - \frac{d}{dx}\left(\frac{\partial G}{\partial y'}\right) = 0$$

if $G$ is a function only of $y$ and $x$, the 2nd term in the above equation vanishes to

$$\Rightarrow \boxed{\frac{\partial G}{\partial y} = 0}$$

So we have to cast our $E(l)$ into the $F$ above.

we see that:- $$E(f) = \int_x \left[ \int_y (y - f(x))^2 p(x,y)\, dy \right] dx$$

so $F(y) \iff E(f)$

$$G(y(x), x) \iff G(f(x), x) = \int_y (y - f(x))^2 p(x,y)\, dy$$

& $$\frac{\partial G}{\partial y} \iff \frac{\partial G}{\partial f(x)} = 0$$

$$\frac{\partial \left( \int_y (y - f(x))^2 p(x,y)\, dy \right)}{\partial f(x)} = 0.$$

$$\Rightarrow \int_y (-2)\,(y - f(x))\, p(x,y)\, dy = 0.$$

$$\Rightarrow \int_y y\, p(x,y)\, dy = \int_y f(x)\, p(x,y)\, dy$$

$$\Rightarrow \int_y y\, p(x,y)\, dy = f(x) \int_y p(x,y)\, dy$$

$$\Rightarrow$$

$$f(x)\, p(x) = \int_y y\, p(x,y)\, dy$$

$$\Rightarrow f(x)\, p(x) = \int_y y\, p(y/x)\, p(x)\, dy$$

$$\Rightarrow f(x)\, \cancel{p(x)} = \cancel{p(x)} \int_y y\, p(y/x)\, dy$$

$$\text{or} \quad \boxed{f(x) = \mathbb{E}(y/x)}$$

Hence for the squared loss, optimal solution is given by the conditional expectation of $y$.

**Q2.** I got inspiration of the following method from R.Nowak lectures on pattern classification.

Ans 2.

conditional distribution of $P(y|x)$ decides the Bayes classifier decision.

Let $f : \mathbb{R}^d \to \{-1, +1\}$ any classifier other than the Bayes (which we call as $f^*$)

so.

$$P(f(x) \neq y) = P(\text{error}) = 1 - P(\text{correct})$$

$$= 1 - P(f(x) = y)$$

$$= 1 - \left[ P(f(x) = y = 1 | x) + P(f(x) = y = 0 | x) \right]$$

Now since we are following the 0-1 loss:

$$\Rightarrow 1 - E\left(\mathbb{1}\{y=1\}\mathbb{1}\{f(x)=1\}/x\right) - E\left(\mathbb{1}\{y=0\}\mathbb{1}\{f(x)=0\}/x\right)$$

$$\Rightarrow 1 - \left[\mathbb{1}\{f(x)=1\} E[\mathbb{1}\{y=1\}|x] + \mathbb{1}\{f(x)=0\} E[\mathbb{1}\{y=0\}|x]\right]$$

$$\Rightarrow 1 - \left[\mathbb{1}\{f(x)=1\} P(y=1|x) + \mathbb{1}\{f(x)=0\} P(y=0|x)\right]$$

$$\Rightarrow 1 - \left[\mathbb{1}\{f(x)=1\}\eta(x) + \mathbb{1}\{f(x)=0\}(1-\eta(x))\right]$$

$P(\text{error})$ for any general classifier

If we are able to prove that

$$P_f(\text{error}) - P_{f^*}(\text{error}) > 0$$

then $f^*$ is optimal.

So Lets consider the difference.

$$P_f(\text{error}) - P_{f^*}(\text{error})$$

$$= \eta(x)\left[\mathbb{1}\{f^*(x)=1\} - \mathbb{1}\{f(x)=1\}\right]$$

$$+ 1-\eta(x)\left[\mathbb{1}\{f^*(x)=0\} - \mathbb{1}\{f(x)=0\}\right]$$

$$\Rightarrow \eta(x)\left[\mathbb{1}\{f^*(x)=1\} - \mathbb{1}\{f(x)=1\}\right]$$

$$+ \left[1-\eta(x)\right]\left[\mathbb{1}\{f^*(x)=1\} - \mathbb{1}\{f(x)=1\}\right]$$

$$\Rightarrow \underbrace{(2\eta(x)-1)}_{(A)}\underbrace{\left(\mathbb{1}\{f^*(x)=1\} - \mathbb{1}\{f(x)=1\}\right)}_{(B)}$$

for $P(1/x) \geq 1/2$

$A \geq 0$ & $B \geq 0$ $\{\mathbb{1}\{f^*(x)=1\}=1$ &
$\mathbb{1}\{f(x)=1\}=0$ or $1\}$

for $P(1/x) \leq 1/2$ $A \leq 0$ and $B < 0$

$\{\mathbb{1}\{f^*(x)=1\}=0$ &
$\mathbb{1}\{f(x)=1\}=0$ or $1$

so $A \times B \Rightarrow$ always Positive.

**Q3.**

**(a)**

We are considering the MNIST dataset for 4 class classification of hand-written digits 1,3,7,8.

**Fisher Linear Discriminant:**

Fisher Linear Discriminant is a great way to reduce the dimension of the feature space from a high number to a smaller but more “useful” dimensions. So we can view this model in terms of dimensionality reduction.

Consider in the following equation, we are reducing the dimension of the input vector $\mathbf{x}$ and projecting it down to one dimension using:

$$y = \mathbf{w^T x}$$

Here we are going to transform the $\mathbf{x}$ from 784 dimensions to 3 dimensions. But this in general also leads to a considerable loss of information.

The best way to encode important data into fewer dimensions is our motive. A bit of it can be done using a Principal Component Analysis, which says essentially that:

- The class means are the simplest measure of the separation of the classes, when projected onto $\mathbf{w}$.
  This suggests that we need that $\mathbf{w}$ which maximizes the following:
  $$m_2 - m_1 = \mathbf{w^T}(\mathbf{m_2} - \mathbf{m_1})$$
  where $m_k = \mathbf{w^T m_k}$ is the mean of the projected data from any of the 4 classes of digit that we have

- There still a bit of problem here. Even after separating the means, we see that there is a considerable overlap between the classes when projected onto the line joining their means. This is an attribute that arises from the strongly non-diagonal covariances in the class distributions.
- Fisher proposes that, instead just maximizing the separation between the class means, we should also ensure that each class’s variance is minimized...thus minimizing the overlap we had been talking about.

Fisher criterion is defined to be the ratio of the between class variance to the within class variance, thus taking care of the above mentioned point.

$$J(w) = \frac{\mathbf{w^T S}_B \mathbf{w}}{\mathbf{w^T S}_W \mathbf{w}}$$

Here SB is the between class covariance matrix given by:

$$\mathbf{S_B} = \frac{(\mathbf{m_1} - \mathbf{m})(\mathbf{m_1} - \mathbf{m})^\mathbf{T} + (\mathbf{m_3} - \mathbf{m})(\mathbf{m_3} - \mathbf{m})^\mathbf{T} + (\mathbf{m_7} - \mathbf{m})(\mathbf{m_7} - \mathbf{m})^\mathbf{T} + (\mathbf{m_8} - \mathbf{m})(\mathbf{m_8} - \mathbf{m})^\mathbf{T}}{4}$$

and $\mathbf{S_w}$ is the total within class covariance matrix, given by

$$\mathbf{S_w} = \sum_{n \in digit_1} (\mathbf{x_n} - \mathbf{m_1})(\mathbf{x_n} - \mathbf{m_1})^T + \sum_{n \in digit_4} (\mathbf{x_n} - \mathbf{m_4})(\mathbf{x_n} - \mathbf{m_4})^T + \sum_{n \in digit_7} (\mathbf{x_n} - \mathbf{m_7})(\mathbf{x_n} - \mathbf{m_7})^T + \sum_{n \in digit_8} (\mathbf{x_n} - \mathbf{m_8})(\mathbf{x_n} - \mathbf{m_8})^T$$

After that, we find the **eigenvalues and eigenvectors of Sb/Sw.**
Then arrange the top k-1 (here 4-1=3) eigenvalues as the projection directions.

W=[eigenvector_1, eigenvector_2, eigenvector_3];

Now we reduce the dimensions of each of the labels from 784 to 3. (by $y = \mathbf{w^T x}$ where WT is the mapping matrix)
This gives me:

1. 3 dimensional training data for label 1.
2. 3 dimensional training data for label 4.
3. 3 dimensional training data for label 7.
4. 3 dimensional training data for label 8.

We try to fit a 3 dimensional gaussian distribution on each of the training data for each label separately.

Then, whenever a new data is fed for testing, we try to see the probability of that point lying in all the 4 gaussian spaces. The one in which we get the highest value is the right class where the new data has to go.
**Imp**: (So while doing this, we still need to reduce the test data's dimensions from 784 to 3 and then proceed with the above instructions)

Results:

net training data error of FLDA : **0.046538 (or 4.65%)**
standard deviation of the training data error of FLDA : **0.76 %**

net testing data error of FLDA : **0.053972 (or 5.39%)**
standard deviation of the testing data error of FLDA : **5.36 %**

**PS:**
I have produced plots in the matlab script where i have shown actual 3d Gaussian fitting the data. (It looks like a elliptical sphere. You may have to zoom in the data to see them)

**Q3 (b)**
**Least squares linear discriminant.**

Least squares linear discriminant approximates the conditional expectation of the target values when the input vector is provided. This is the basic theme of the least squares linear regression.

To get the above idea working for us in terms of classification (using the binary coding scheme), this conditional expectation is given by the vector of posterior class probabilities. The class having the maximum value of such probability is said to be the required class.

Since this is a multi-class classification problem, each class is to be described by its own linear model so that

$y_k(x) = \mathbf{w}^{\mathbf{T}}{}_k\mathbf{x} + w_{k0}$ where k = 1, 4, 7 and 8.

In terms of vector notations:

$$y(x) = \mathbf{W}^{\mathbf{T}}\mathbf{x}$$

where W's kth column comprises of the 784+1(augmented) dimensional vector

$$\mathbf{w_k} = (w_{k0}, \mathbf{w}_k^T)^{\mathbf{T}}$$

We then need to determine the parameter matrix W. This we can achieve by minimizing a sum-of-squares error function.

$$min(\frac{1}{2}\sum_{n=1}^{N}(t_n - \mathbf{W}^{\mathbf{T}}\mathbf{x}))$$

The W at which the above comes out to be:

$$\mathbf{W} = pinv(\mathbf{X})\mathbf{T}$$

where pinv() is the pseudo-inverse of the given function. This was the essential training part. We have determined a W that fits the linear regression model.

To find the testing error rate, we just multiply this W with the testing data. The output is a 1 by 4 vector. The column corresponding to the maximum value gives us the correct label of the test data.

Results:

net training data error of Least Squares : **5.343706%**
standard deviation of the training data error of Least Squares : **0.91%**

net testing data error of Least Squares : **6.097114%**
standard deviation of the testing data error of Least Squares : **6.58%**

**Q4(a)**

Will have to describe in brief the approaches in Logistic Regression and Naive Bayes along with the iterative equations used there.

**Logistic Regression:**

Herein, the posterior probability of of the data being spam can be written as a logistic sigmoid of the the feature vector times the design vector W.

$$\sigma(\mathbf{W^T}X) = \frac{1}{1+e^{(-\mathbf{W^T}X)}}$$

To fit this model (or to say that to fit the parameters of matrix W) to our data, we have to used maximum likelihood.
To write the above function in a simplified manner, we can write it as:

Cost Function:

$$\left(\frac{1}{1+e^{(-\mathbf{W^T}X)}}\right)^{t_n}\left(1-\frac{1}{1+e^{(-\mathbf{W^T}X)}}\right)^{1-t_n}$$

We are able to write the above function function since there are only 2 classes available to us. i.e spam and non-spam.

So taking the negative logarithm of the likelihood we get the cross entropy error function. Again differentiating it with respect to W and equating it to 0, we are able to form the closed form solution for the design matrix W.

$$\nabla(-\log(CostFunction)) = \nabla(Error) = 0$$

that is :

$$\nabla(Error) = \sum_{n=1}^{N}(\sigma(W^T X_n) - t_n)X = 0$$

We here note that this looks very much similar to the gradient of the sum-of-squares error function for the linear regression model (except for the sigmoid function which is not there).

Now In **my approach** I am making use of this result to give a sequential algorithm called **the gradient descent**. Starting with any initial value for the parameters in the design matrix W, I follow the **Stochastic gradient descent rule** herein:

Initialize W_i's =1;

for (all observations X_i)
    current_result_i = sigmoid(<W, X_i>)

    % Calculate the gradient of the error for the ith result. (We want this to be 0)
    gradient = (current_result_i - actual_label_i)X_i;

    % Tweak the parameters of the W matrix because of the gradient.
    **W = W - (gradient) * alpha;**  % Most important step here.
end

Check the **W** on the training set data. **Keep on repeating** the above loop until we get approximately the minimum error!

Note:
After running the code for 900 times(in my coding), I assume that I have substantially reached the best that LR could give me.

To generate plots, above is to be repeated 100 times.
My results are as follows:

| **% of training data taken:** | **5%** | **10%** | **15%** | **20%** | **25%** | **30%** |
|---|---|---|---|---|---|---|
| **Mean error of logistic R:** | **0.2528** | **0.2353** | **0.2234** | **0.1979** | **0.1936** | **0.1858** |
| **std of the logistic R:** | **0.0953** | **0.0986** | **0.1320** | **0.0945** | **0.0894** | **0.1043** |

**Q4 (b)**

**Naive Bayes**

Naive Bayes is a simple probabilistic classifier based on Bayes theorem and the conditional Independence assumption.

According to Bayes Theorem:

$$p(C_k|\mathbf{x}) = \frac{p(C_k)p(\mathbf{x}|C_k)}{p(\mathbf{x})}$$
$$C_1 \rightarrow spam, C_0 \rightarrow non-spam$$

Denominator is just a normalizer, so we can ignore that for seeking whether the p(C_1/x) is greater or p(C_0/x).
Finding the posterior and following the “conditional Independence” assumptions, we see that we can break the above to the following:

$$p(C_0|\mathbf{x}) = p(C_0)p(x_1|C_0)p(x_2|C_0)p(x_3|C_0)p(x_4|C_0)...p(x_{57}|C_0)$$

similarly for the other class, we have the following:

$$p(C_1|\mathbf{x}) = p(C_1)p(x_1|C_1)p(x_2|C_1)p(x_3|C_1)p(x_4|C_1)...p(x_{57}|C_1)$$

So we basically try to fit the univariate gaussian curve 57* 2 times(Using M.L.Estimate) on each of the probabilities above.
Rest all is simple multiplication.

**Testing**

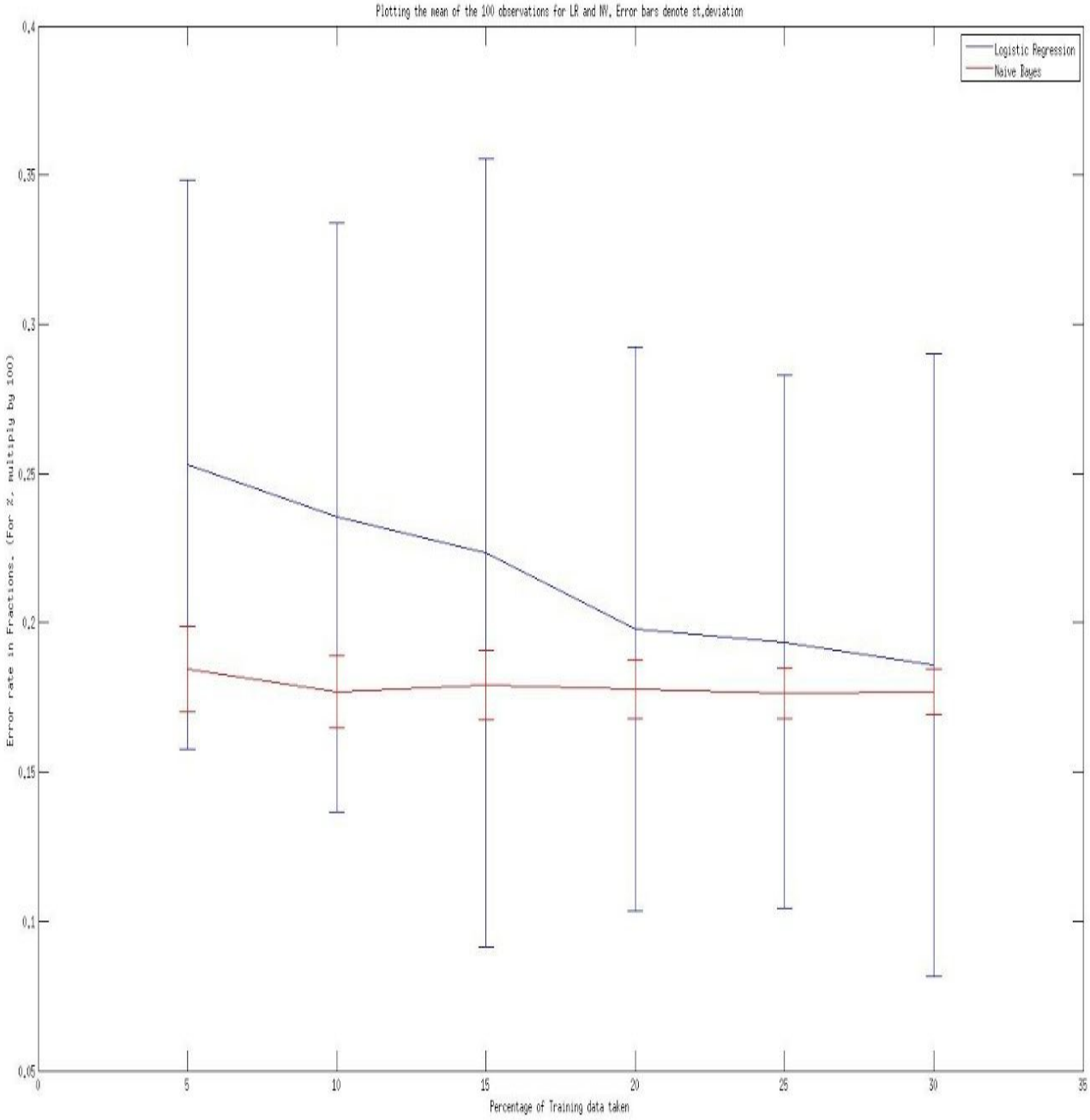
If p(C_0|x_i) >p(C_1|x_i) for a test data x_i then we predict that it qualifies for Non-Spam.
If Otherwise, then it qualifies under Spam.

To generate plots, above is to be repeated 100 times.
The results are as follows:

| **% of training data taken:** | **5%** | **10%** | **15%** | **20%** | **25%** | **30%** |
|---|---|---|---|---|---|---|

**mean error of Naive Bayes:** **0.1844 0.1770 0.1790 0.1778 0.1766 0.1768**
**std error of the Naive Bayes:** **0.0143 0.0121 0.0117 0.0097 0.0085 0.0077**

**Plot is in the next page (I have also attached the image in the zip if it is not clear here)**


Plotting the mean of the 100 observations for LR and NV. Error bars denote st.deviation
Logistic Regression
Naive Bayes
Error rate in Fractions. (For %, multiply by 100)
Percentage of Training data taken
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
5
10
15
20
25
30
35