CSCI 5525
Machine Learning Assignment 1
Rahul Biswas
October 7, 2018

Fisher's Linear Discriminant

Fisher's Linear Discriminant for two classes problem - we first construct the within-class covariance matrix S_W and between class covariance matrix S_B.

Let K be the number of classes, C_k be the indicator of class k, N_k be the number of observations in class k. Then we can derive S_W and S_B by:

$$S_W = \sum_{k=1}^{K} S_k$$

$$S_B = \sum_{k=1}^{K} N_k (m_k - m)(m_k - m)^T$$

where

$$S_k = \sum_{n \in C_k} (x_n - m_k)(x_n - m_k)^T$$

$$m_k = \frac{1}{N_k} \sum_{n \in C_k} x_n$$

Let W be the resulting weight matrix for Fisher's Linear Discriminant, we can derive it by maximizing the following quantity

$$J(W) = \text{Tr}\left\{(WS_W W^T)^{-1}(WS_B W^T)\right\}$$

The weight matrix are determined by those eigenvectors of $S_W^{-1} S_B$ that correspond to the D' largest eigenvalues.

Note - Sometimes calculating eigenvectors for $S_W^{-1} S_B$ results in all complex vectors, which is due to the bad condition of S_W. For this case we simply find eigenvectors for $(S_W + I)^{-1} S_B$ as a substitute.

Problem 3.a

We will first be applying this to the HomeVal50 dataset. This is a 13 feature dataset, which we would be dividing categorically, so it becomes a two class problem. Since it has 13 features, our input data is in 13 dimensions. Using LDA, we would try to project these into a lower dimensional space, maybe 1 or 2 dimensions.

So, we calculate the eigen vectors for the data set. Eigen vectors are helpful in reducing the dimensions. We usually select the maximum eigen vector because that would give us the maximum information about the dataset if we project it on that. After calculating the eigen values, we notice that we get something like,

As we can see, the first eigen value is the only one which is not 0 and everything from the second one, gets negligible. Hence, we can assume that one eigen value (which means, one dimension) is good enough to represent the data. Since, this is a 2 class problem, we can use $S_W^{-1}(m2 - m1)$ instead of $S_W^{-1} S_B$

Plotting this, gives the histogram - which shows that the data can be divided somewhat into two separate classes.
Hence, it is okay to say that a 1 dimensional linear discriminant does a good job.

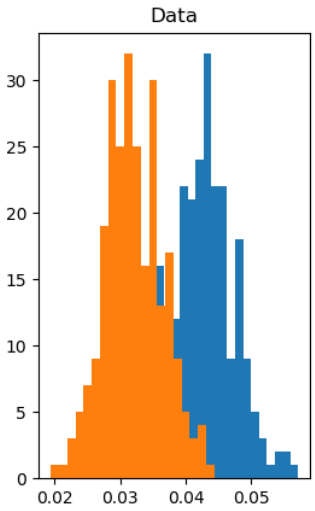

Figure 1: Histogram of 2 classes

Problem 3.b

Note that S_B is composed from K rank 1 matrices, and only $(K-1)$ of these matrices are independent, thus S_B has rank at most equal to (K - 1) and so there are at most (K - 1) nonzero eigenvalues. Hence we are unable to find more than $(K-1)$ linear "features" by this means. So, for the HomeVal50 dataset, we can not project the data into 2 dimensions.
If one tries to code this, we can see that it does not do a good job.

Problem 3.c

We would be using the Digits dataset for this problem. The Digits dataset has 10 output classes, so it's a multi-class problem.

Multivariate Gaussian Generative Model

Multivariate Gaussian Generative Model assumes that each class of observations come from a multivariate Gaussian distribution. To make prediction for single observations, it applies the following Bayes rule

$$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{p(x)}, \quad k = 1, \ldots, K$$

Note that $p(x)$ is equal for all k, hence we can ignore the denominator of the posterior probability.

For $p(x|C_k)$, since we assume C_k has a multivariate normal distribution, it's sufficient to estimate the

Table 1: Fisher + Multivariate Gaussian for Digits Data

	Mean	Std	fold1	fold2	fold4	fold6	fold8	fold10
Fisher Train Erros	10.80%	0.03%	10.61%	7.8%	8.93%	8.37%	10.61%	15.08%
Fisher Test Erros	32.56%	0.05%	27.90%	36.3%	29.6%	37.98%	38.5%	37.43%

mean μ_k and the covariance matrix Cov_k of C_k, where

$$\hat{\mu}_k = \frac{1}{N_k} \sum_{n \in C_k} x_n$$
$$\hat{\text{Cov}}_k = \frac{1}{N_K - 1} \sum_{n \in C_k} (x_n - \hat{\mu}_k)(x_n - \hat{\mu}_k)^T$$

Then it's straightforward to estimate $p(x|C_k)$.

Now, to calculate $p(C_k|x)$, we would be using the formula,

$$p(C_k|x) = \sigma(w_k^T.x + w_k 0)$$
$$w_k = \sum^{-1} \mu_k$$
$$w_{k0} = -0.5 \mu_k^T \sum^{-1} \mu_k + logp(C_k)$$

We would be taking the following steps to solve the question.

1. We would be calculating $S_W^{-1} S_B$ to obtain the linear discriminant.
2. Choosing the top 2 eigen vectors, because we are projecting it onto two dimensions.
3. Transform the data into a new subspace, $X_{new} = X.W$ where X is the original data and W is the eigen vector.
4. Once we have the new data, we send that to our Gaussian function, find priors, covariance, mean of every single class by itself and then apply the formula written above. Then, we use argmax() to get our answer and then compare with the testing data.

One instance of output of the program -
Train error-rate for fold-1: 0.08379888268156421
Train error-rate for fold-2: 0.1005586592178771
Train error-rate for fold-3: 0.0949720670391061
Train error-rate for fold-4: 0.11731843575418999
Train error-rate for fold-5: 0.11731843575418999
Train error-rate for fold-6: 0.07262569832402233
Train error-rate for fold-7: 0.11173184357541899
Train error-rate for fold-8: 0.1005586592178771
Train error-rate for fold-9: 0.12849162011173187
Train error-rate for fold-10: 0.11731843575418999
Train error-rate for fold-11: 0.1428571428571429

Mean train-error 10.80 percent
Std train-error 0.02 percent
Test error-rate for fold-1: 0.2793296089385475
Test error-rate for fold-2: 0.36312849162011174
Test error-rate for fold-3: 0.26815642458100564
Test error-rate for fold-4: 0.2960893854748603
Test error-rate for fold-5: 0.3240223463687151
Test error-rate for fold-6: 0.3798882681564246
Test error-rate for fold-7: 0.23463687150837986
Test error-rate for fold-8: 0.3854748603351955
Test error-rate for fold-9: 0.3910614525139665
Test error-rate for fold-10: 0.3743016759776536
Test error-rate for fold-11: 0.2857142857142857

Mean test-error 32.56 percent
Std test-error 0.05 percent

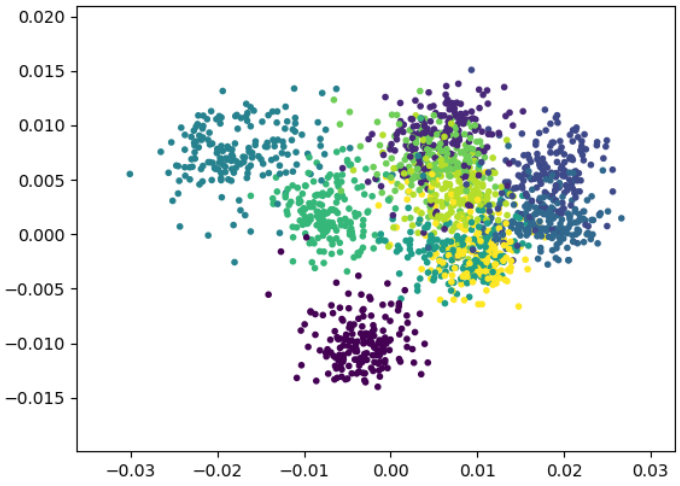

Figure 2: Projection

Problem 4.a

Logistic Regression

Logistic Regression assumes

$$p(C_1|x) = \sigma(w^T x)$$

where $\sigma(\cdot)$ is the logistic sigmoid function

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

The sigmoid function is used when we have a binary classification and we use the softmax function when we have more than two classes.
Note - You can use softmax function for binary classification as well, which is what my code does.

It is a discriminant model, where we take $p(y|x)$ to solve the problem.
We approach the problem by finding the extremum of the log-likelihood function,

$$\sum_{n=1}^{N} y_n w^T x_n - \log(1 + exp(w^T x_n)$$

We would solve this equation in "w" by the method mentioned in the question, Iterately Reweighted Least Sqaures Method. We would be updating the weight by solving the weighted linear least square problem.

$$W^{k+1} = (X^T R X)^{-1}.X^T.R.y$$

where R is the diagonal matrix of weights, with initialization to 1.

We update R in every iteration, in effect, updating W as well.

$$r_i^t = \frac{1}{max(\delta, |y - X_i.W^T|)}$$

By iteratively update w till convergence (or maximum number of iteration is reached), we can obtain an estimate for w.
Note - I have set the toleration constant manually. (as 0.001)

Naive Bayes with Gaussian Approximation

Naive Bayes assumes features are independent with each other, which suggests that

$$p(x_1, \ldots, x_p|C_k) = \prod_{i=1}^{p} p(x_i|C_k)$$

By univariate Gaussian approximation, we assume each $p(x_i|C_k),\ i = 1, \ldots, N,\ k = 1, \ldots, K$ has a Gaussian distribution, whose parameters can be estimated from sample.

For prediction, simply apply Bayes rule

$$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{p(x)} = \frac{\prod_{i=1}^{p} p(x_i|C_k)p(C_k)}{p(x)}$$

$$p(C_k|x) = \sigma(w_k^T.x + w_k 0)$$
$$w_k = \sum^{-1} \mu_k$$
$$w_{k0} = -0.5\mu_k^T \sum^{-1} \mu_k + logp(C_k)$$

Then we can predict new observation as class k if the prediction score for class k has the largest value among others.

Results and Outputs

We have conducted the experiment of two datasets, the Boston 50 percentile dataset and the Digits dataset. The Boston data is a 2 class problem whereas the Digits dataset is a multi-class problem.

For each iteration, we first split the whole dataset into 80% for training and 20% for test. Then, within training set, we randomly select 10%, 25%, 50%, 75%, 100% for training the classifier, and calculate the errors for both training and testing each of the inner-splits. The whole procedure will be repeated 10 (user-input) times. We apply this validation procedure to evaluate the prediction performance of the following two models:

1. Logistic Regression with Iterately Reweighted Least Squares
2. Naive Bayes with Marginal Univariate Gaussian Approximation

The simulation results are summarized in tables below.

As we can see from the plots and tables, logistic regression beats Naive Bayes pretty consistently(the first point, at 10 % training percent is a little ambiguous)
We also notice that logistic regression gets much better and has an improved test error performance with higher training percentages.
On the other hand, Naive Bayes is pretty consistent and stays insensitive to training percentages.

Case 1 - Dataset Boston.csv

Table 2: Logistic VS Naive Bayes for Boston.csv

	10 %	25 %	50 %	75 %	100 %
Logistic Test Error Mean	20.19%	16.96%	16.76%	15.78%	15.09%
Logistic Test Error Std	4.96%	1.38%	1.98%	2.7%	1.8%
Naive Bayes Test Error Mean	19.05 %	20.19 %	20.07 %	20.07%	20.15%
Naive Bayes Test Error Std	2.32%	2.07%	2.47%	2.75%	2.71%

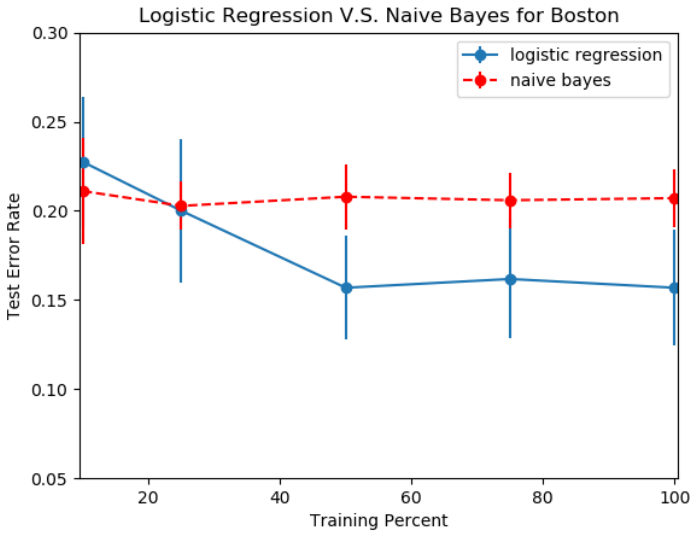

Figure 3: Boston dataset example 1

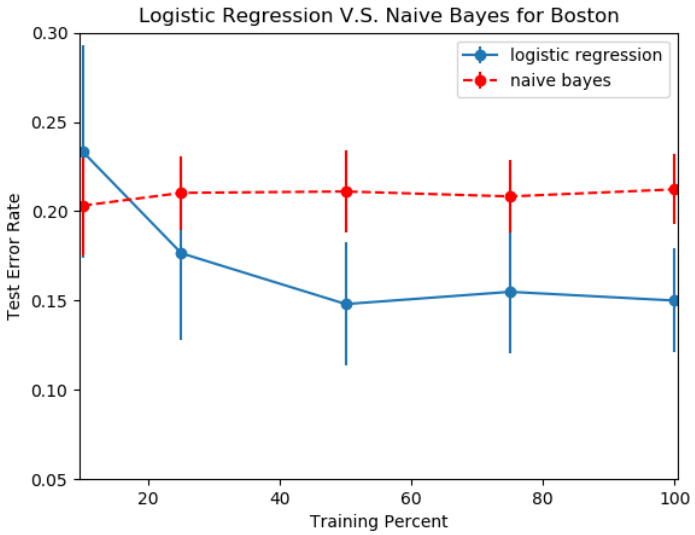

Figure 4: Boston dataset example 2

Case 2 - Dataset Digits.csv

Table 3: Logistic VS Naive Bayes for Digits.csv

	10 %	25 %	50 %	75 %	100 %
Logistic Test Error Mean	20.55%	10.41%	7.97%	7.47%	7.30%
Logistic Test Error Std	2.37%	1.4393 %	1.19%	1.36 %	1.39%
Naive Bayes Test Error Mean	14.88%	12.46%	11.20%	11.04%	10.81%
Naive Bayes Test Error Std	1.68%	1.38%	0.78%	0.749%	0.83%

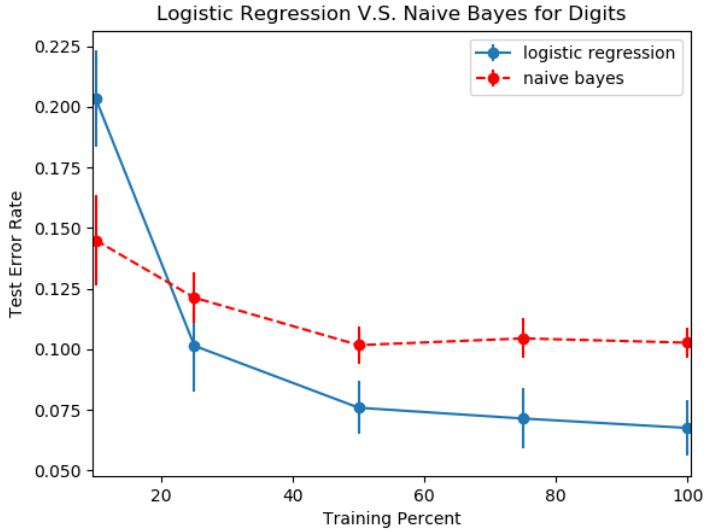

Figure 5: Digits dataset example 1

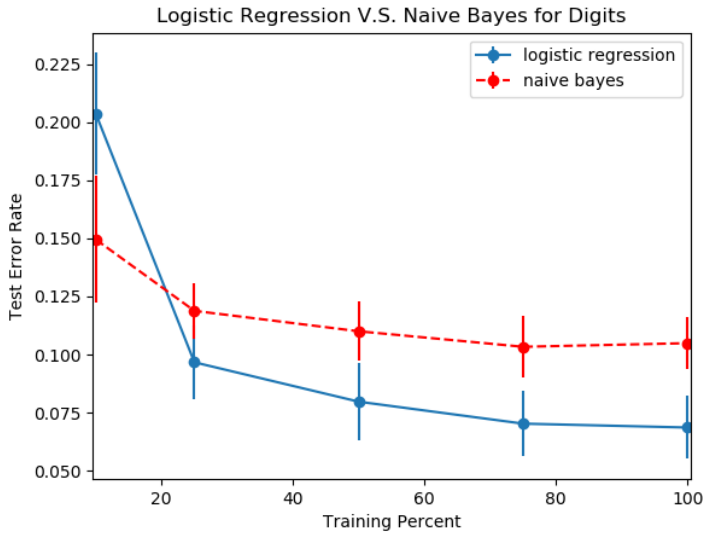

Figure 6: Digits dataset example 2

ASSIGNMENT 1

MACHINE LEARNING

RAHUL BISWAS

Question 1

(a)

$$E[l(f(x), y)] = \int_x \left\{ \int_y l(f(x), y)\, p(y/x) \right\}$$

$$l(f(x), y) = (f(x) - y)^2$$

There are 2 methods to solve this:

Method 1:

$$E[l(f(x), y)] = \int_x \int_y (f(x) - y)^2\, p(y/x)\, dy\, p(x)\, dx$$

We minimize the expected loss,

$$\min_f E_{(x,y)} \Rightarrow \min_f \int_y (f(x) - y)^2\, p(y/x)\, dy\, p(x)\, dx$$

Take the derivative & set it to 0

$$\frac{\partial}{\partial f} \int_y \int (f(x) - y)^2 p(y/x)\, dy\, p(x)\, dx = 0$$

Since we know that $p(x, y) = p(y/x) dy$ and since we are integrating wrt only $\int_y$, we can ignore $p(x)\, dx$.

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

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

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

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

$f(x)$ on LHS has no y component,

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

$$f(x) = \frac{\int y\,p(y/x)\,dy}{\int p(y/x)\,dy} = E[y/x]$$

$\Rightarrow$ Optimal $f(x)$ is when it is $E[y/x]$

Method 2

We will add $E[y/x]$ in this one.

$$E[l(f(x),y)] = \int_x\int_y (f(x)-y)^2\,p(y/x)\,dy\,p(x)\,dx$$

$\Rightarrow$ Consider only the $(f(x)-y)^2$ part:

$$(f(x)-y)^2 = (f(x) - E[y/x] + E[y/x] - y)^2$$

$$= (f(x) - E[y/x])^2 + 2(f(x)-E[y/x])(E[y/x]-y) + (E[y/x]-y)^2$$

The cross term vanishes when we take the integral over $f(x)$.

$$E[L] = \int \left(f(x) - E[y/x]\right)^2 p(x)\,dx + \int \text{variance}[y/x]\, p(x)\,dx$$

We see the term $f(x)$ only in the first term which will be minimized when

$$f(x) = E[y/x]$$

(b) $l(f(x), y) = |f(x) - y|$

$$E[L] = \int_x \int_y |f(x) - y|\, p(y/x)\,dy\; p(x)\,dx$$

Derivative of the inner function will give us the following →

$$\frac{\partial}{\partial f} \int_y |f(x) - y| \, p(y/x) \, dy = 0$$

$$\frac{\partial}{\partial f} \left[\int_{-\infty}^{f(x)} (y - f(x)) \, p(y/x) \, dy + \int_{f(x)}^{\infty} (f(x) - y) \, p(y/x) \, dy = 0 \right.$$

$$\int_{-\infty}^{f(x)} p(y/x) \, dy = \int_{f(x)}^{\infty} p(y/x) \, dy$$

Hence, the optimum solution is the median.

Method 2

$$\int |f(x) - y| \, p(y/x) \, dy$$

$$= \int \text{signum}(f(x) - y)$$

and since we are deriving and setting it to 0, the only value of signum where it is 0 is the median.

Question 2

We will start with the approach that

$$E[C_j] = \sum_{i=1:M} \int_{x \in R_i} P(C_j/x)\, p(x)\, dx = 1$$

overall error where $y == C_j$ and $y != C_j$

We also know that we can break down in

$$\underbrace{\sum_{\substack{i=1:M \\ \text{and } i \neq j}} \int_{x \in R_i} p(C_j/x)\, p(x)\, dx}_{\text{Region where it is wrongly classified}} + \underbrace{\int_{\substack{x \in R_i \\ i=j}} p(C_j/x)\, p(x)\, dx}_{\text{Region where it is correctly classified}}$$

We know that $E[C_j]$ is the overall error, so.

$$E[C_j] = E[C_{j,\, y \neq j}] + E[C_{j,\, y=j}]$$

$$\sum_{i=1:M} \int p(C_j/x)\,p(x)\,dx = \underbrace{\sum_{\substack{i=1:M \\ i \neq j}} \int p(C_j/x)\,p(x)\,dx}_{E[y = C_j]} + \int_{x \in R_i} p(C_j/x)\,p(x)\,dx$$

$$E[y = C_j] = \sum_{i=1:M} \int p(C_j/x)\,p(x)\,dx - \int_{x \in R_j} p(C_j/x)\,p(x)\,dx$$

Hence, proved.