CSCI567 MACHINE LEARNING

Assignment 2

Suraj R
rajasekh@usc.edu
4979-2190-97

Question 1.A)

Priors	Conditionals
$p(X = 240) = \sum_i p(X = 240, y = y_i) = \frac{4}{9}$	$p(X = 240\ y = yes) = 0.01$ $p(X = 240\ y = no) = \frac{1}{2}$
$p(X = 343) = \sum_i p(X = 343, y = y_i) = \frac{4}{9}$	$p(X = 343\ y = yes) = 0.01$ $p(X = 343\ y = no) = \frac{1}{2}$
$p(X = 422) = \sum_i p(X = 422, y = y_i) = \frac{1}{18}$	$p(X = 422\ y = yes) = \frac{1}{2}$ $p(X = 422\ y = no) = 0.01$
$p(X = 031) = \sum_i p(X = 031, y = y_i) = \frac{1}{18}$	$p(X = 031\ y = yes) = \frac{1}{2}$ $p(X = 240\ y = no) = 0.01$
$p(y = "NO") = {}^4/_9 \times {}^2/_1 = \frac{8}{9}$	
$p(y = "YES") = {}^1/_{18} \times {}^2/_1 = \frac{1}{9}$	

Question 1.B)

Yes, the password 031 will open the gate.

Using Bayes theorem,

$$p(y = Yes|X = 031) = \frac{p(y = yes)p(X = 031|y = yes)}{p(x = 031)}$$

$$p(y = Yes|X = 031) = \frac{\frac{1}{9} \times \frac{1}{2}}{\frac{1}{18}} = 1.0$$

Question 1.C)

Logistic regression Estimates the probability ($\frac{y}{x}$) directly from the training data by minimizing error. Hence this is a Discriminative model where as for the given features (x) and the label y, Naïve Bayes estimates a joint probability from the training data. Hence this is a Generative model

Suppose we are given a dataset of pairs (x, c) where c is a class variable and x is a vector of features. Given a new x, if we want to predict its class, the Naïve Bayes or generative $i.i.d$ approach to this problem points to,

$$p(x, c|\theta) = p(x\ |c, \lambda)\ p(c|\pi)$$

And chooses the best parameter $\theta = \{\lambda, \pi\}$ by maximizing the joint distributions.

$$p(D, \theta) = p(\theta) \prod_i p(x_i, c_i \,|\theta) = p(\theta) \prod_i p(\, x_i \mid c_i \,, \lambda) p(\, c_i \,|\pi)$$

Whereas the discriminative approach chooses the θ by maximizing the conditional distributions:

$$p(C, \theta | X) = p(\theta) \prod_i p(c_i \,|x_i, \theta)$$

Where

$$p(c|x, \theta) = \frac{p(x,c|\theta)}{\sum_c p(x,c|\theta)}$$

Question 1.D)

Naïve Bayes assumes all the features are conditionally independent. So, if some of the features are dependent on each other (in case of a large feature space), the prediction might be poor. In case of *Logistic regression,* splits the feature space linearly and it works OK even if some of the variables are correlated.

There are some limitations also. *Naïve Bayes w*orks well even with less training data, as the estimates are based on the joint density function. But *Logistic regression, even* with the small training data, model estimates may over fit the data.

Question 2. A

As mentioned, please find the below 5 MLE parameters.

$$p(D|\pi, \mu_0, \mu_1, S_0, S_1) = \sum_{i=1}^{N} p(x_i|y_i, S_0, S_1) p(y_i, \pi)$$

$$\pi = \frac{1}{N} N_{y=1}$$

$$\mu_0 = \frac{\sum_{i=1}^{N} t_{y=0} x_i}{N_{y=0}}$$

$$\mu_1 = \frac{\sum_{i=1}^{N} t_{y=0} x_i}{N_{y=1}}$$

$$S_D = \frac{1}{N} \sum_{t=1}^{N} t_{i,y=0} (x_i - \mu_0)(x_i - \mu_0)^{\,T}$$

$$S_D = \frac{1}{N} \sum_{t=1}^{N} t_{i,y=1} (x_i - \mu_1)(x_i - \mu_1)^{\,T}$$

Question 2.B)

Given the posterior probability in the question,

$$p(y=1|x,\pi,\mu_0,\mu_1,S_0,S_1)=\frac{p(y=1|\pi)p(x|y,\mu_1,S_1)}{p(y=1|\pi)p(x|\mu_1,S_1)+p(y=0|\pi)p(x|\mu_0,S_0)}$$

Multiply and divide by numerator,

$$p(y=1|x,\pi,\mu_0,\mu_1,S_0,S_1)=\frac{1}{1+\frac{p(y=0|\pi)p(x|\mu_0,S_0)}{p(y=1|\pi)p(x|\mu_1,S_1)}}$$

$$=\frac{1}{1+\exp\left(ln\frac{p(y=0|\pi)p(x|\mu_0,S_0)}{p(y=1|\pi)p(x|\mu_1,S_1)}\right)}$$

$$=\frac{1}{1+exp\left(ln\frac{p(y=0|\pi)}{p(y=1|\pi)}+\sum_{i=1}^{N}ln\frac{p(x_i|\mu_0,S_0}{p(x_i|\mu_1,S_1}\right)}$$

$$\frac{1}{1+exp\left(ln\frac{1-\pi}{\pi}+\sum_{i=1}^{N}ln\frac{p(x_i|\mu_0,S_0}{p(x_i|\mu_1,S_1}\right)}$$

Considering the second part of denominator,

$$\sum_{i=1}^{N}ln\frac{P(X_i|Y=0)}{P(X_i|Y=1)}=\sum_{i=1}^{N}ln\frac{\frac{1}{2\pi^{\frac{n}{2}}|S|^{\frac{1}{2}}}\exp\left[-\frac{1}{2}(X_i-\mu_{i0})^{T}S^{-1}(X_i-\mu_{i0})\right]}{\frac{1}{2\pi^{\frac{n}{2}}|S|^{\frac{1}{2}}}\exp\left[-\frac{1}{2}(X_i-\mu_{i1})^{T}S^{-1}(X_i-\mu_{i1})\right]}$$

$$=\sum_{i=1}^{N}\frac{S^{-1}}{2}[(X_i-\mu_{i1})^{2}-(X_i-\mu_{i0})^{2}]$$

$$=\sum_{i=1}^{N}S^{-1}\left[X_i(\mu_{i0}-\mu_{i1})+\frac{(\mu_{i1}{}^{2}-\mu_{i0}{}^{2})}{2}\right]$$

Using this in A,

$$p(y=1|x,\pi,\mu_0,\mu_1,S_0,S_1)=\frac{1}{1+exp\left(ln\frac{1-\pi}{\pi}+\sum_{i=1}^{N}S^{-1}\left[X_i(\mu_{i0}-\mu_{i1})+\frac{(\mu_{i1}{}^{2}-\mu_{i0}{}^{2})}{2}\right]\right)}$$

$$=\frac{1}{1+\exp(\omega_0+\sum_i^N\omega_i x_i)}$$

Where

$$\omega_i=S^{-1}(\mu_{i0}-\mu_{i1})$$

$$\omega_0 = ln\frac{1-\pi}{\pi} + \sum_{i=1}^{N} S^{-1}\frac{({\mu_{i1}}^2 - {\mu_{i0}}^2)}{2}$$

$$p(y = 1|x, \pi, \mu_0, \mu_1, S_0, S_1) = \frac{1}{1 + \exp(-\theta^T x)}$$

Question 2.C)

We know that from Poisson distribution,

$$x|y = 0 \sim Poisson(\lambda_0) = p(X = k) = \frac{{\lambda_0}^k e^{-\lambda_0}}{K!}$$
$$= \exp(K\ ln\ \lambda_0 - \lambda_0 - ln\Gamma(K+1))$$

Similarly,

$$x|y = 1 \sim Poisson(\lambda_1) = p(X = k) = \frac{{\lambda_1}^k e^{-\lambda_1}}{K!}$$
$$= \exp(K\ ln\ \lambda_1 - \lambda_1 - ln\Gamma(K+1))$$

Let's substitute this in the above equation, and we have proved that posterior distribution can be simplified into generalized linear model.

$$p(y = 1|x, \pi, \mu_0, \mu_1, S_0, S_1) = \frac{1}{1 + \exp\left(ln\left(\frac{p(x|y, \mu_0, S_0)}{p(x|y, \mu_1, S_1)}\right)\right)}$$

$$= \frac{1}{1 + \exp\left(ln\frac{1-\pi}{\pi} + ln\left(\frac{\exp(K\ ln\ \lambda_0 - \lambda_0 - ln\Gamma(K+1))}{\exp(K\ ln\ \lambda_1 - \lambda_1 - ln\Gamma(K+1))}\right)\right)}$$

$$= \frac{1}{1 + \exp((K\ ln\ \lambda_0 - \lambda_0 - ln\Gamma(K+1)) - (K\ ln\ \lambda_1 - \lambda_1 - ln\Gamma(K+1))}$$

$$= \frac{1}{1 + \exp(k(ln\ \lambda_0 - ln\ \lambda_1) + (\lambda_1 - \lambda_0))}$$

$$= \frac{1}{1 + \exp(\sum_i x_i\ ((ln\ \lambda_0 - ln\ \lambda_1) + (\lambda_1 - \lambda_0))}$$

In general, we can write as,

$$= \frac{1}{1 + \exp(w_0 + \sum_i w_i x_i)}$$

Where

$w_0 = ln\frac{1-\pi}{\pi}(\lambda_1 - \lambda_0)\ and\ w_i = \ln\lambda_0 - \ln\lambda_1$

Question 3.

Given $l2$ regularization equation,

$$\sum_{n=1}^{N} logp(y_n|x_n,\omega) - \lambda \sum_{k=1}^{c} \|\omega_k\|_2^2$$

We know that $p(y=c|x,\omega)$ is given below

$$\sum_{n=1}^{N} log \frac{e^{\omega_{c0}+\omega_c^T x}}{\sum_{k=1}^{C} e^{\omega_{k0}+\omega_k^T x}} - \lambda \sum_{k=1}^{c} \|\omega_k\|_2^2$$

Let's substitute $w_c^T = [\omega_{c0}, \omega_c]\ and\ X^T = [1\ x]$. Applying MLE,

$$\frac{\partial p(y|x,\omega)}{\partial w_c} = \frac{\partial}{\partial w_c}\left(\sum_{n=1}^{N}\left(\log e^{w_c^t x_n} - log \sum_{k=1}^{c} e^{w_k^t x_n}\right)\right) - \lambda \sum_{k=1}^{c} \|\omega_k\|_2^2$$

Differentiate with respect to w_c in the above equation and each term is as follows.

$$\frac{\partial}{\partial w_c}\sum_{n=1}^{N}\left(\log e^{w_c^t x_n}\right) = \sum_{n:y_n=c}^{N} x_n$$

$$\frac{\partial}{\partial w_c}\sum_{n=1}^{N} log \sum_{k=1}^{c} e^{w_k^t x_n} = \sum_{n=1}^{N} \frac{x_n e^{w_c^t x_n}}{\sum_{k=1}^{c} e^{w_k^t x_n}}$$

$$\frac{\partial}{\partial w_c}\sum_{k=1}^{c} \|\omega_k\|_2^2 = \frac{\partial}{\partial w_c}\sum_{j} j\omega_{cj}^2 = 2w_c$$

Substituting these in the above equation and replacing $w_c = w$, we get,

$$\frac{\partial p(y|x,W)}{\partial w} = \sum_{n:y_n=c}^{N} x_n - \sum_{n=1}^{N} \frac{x_n e^{w_c^t x_n}}{\sum_{k=1}^{c} e^{w_k^t x_n}} - 2\lambda w = 0$$

Similarly for all C classes $w_1, \dots\ w_c$ and calculating $\frac{\partial p(y|x,W)}{\partial w_c}$

$$\sum_{k=1}^{c}\sum_{n:y_n=c}^{N} x_n - \sum_{k=1}^{c}\sum_{n=1}^{N} \frac{x_n e^{w_c^t x_n}}{\sum_{k=1}^{c} e^{w_k^t x_n}} - \sum_{k=1}^{c} 2\lambda w_k = 0$$

$$\sum_{n=1}^{N} x_n - \sum_{k=1}^{c} \sum_{n=1}^{N} \frac{x_n e^{w_c^t x_n}}{\sum_{k=1}^{c} e^{w_k^t x_n}} - \sum_{k=1}^{c} 2\lambda w_k = 0$$

$$\sum_{k=1}^{c} w_k = \frac{\sum_{k=1}^{c} \sum_{n=1}^{N} \frac{x_n e^{w_c^t x_n}}{\sum_{k=1}^{c} e^{w_k^t x_n}}}{2\lambda}$$

$$\sum_{k=1}^{c} w_k = \sum_{n=1}^{N} x_n - \sum_{n=1}^{N} x_n \ /2\lambda$$

$$\sum_{k=1}^{c} w_k = 0$$

A zero vector on left hand side must have all components 0, thus

$$\sum_{k=1}^{c} \omega_k = 0$$

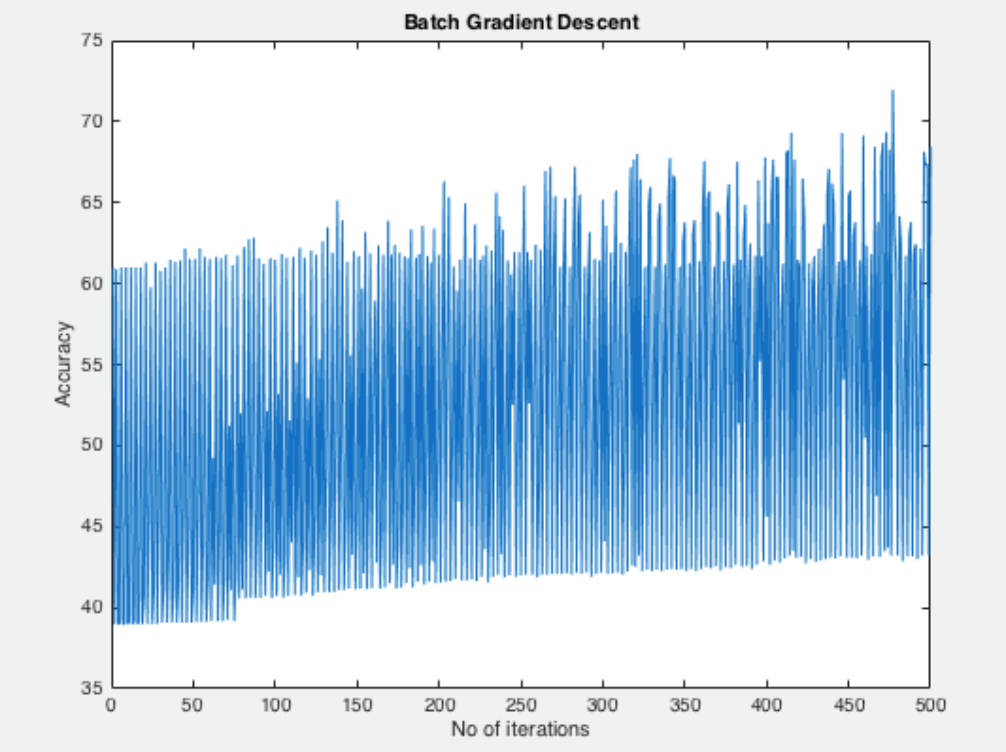
Programming 4.1

B.

Accuracy	Raw data	Normalized data
Batch Gradient	Test: 67.22% Train: 68.43%	Test: 87.82% Train: 89.78%
Newton	Test: 65.22% Train: 64.09%	Test: 87.15% Train: 89.35%
glmfit	Test: 89.65% Train: 87.56%	Test: 87.95% Train: 89.52%

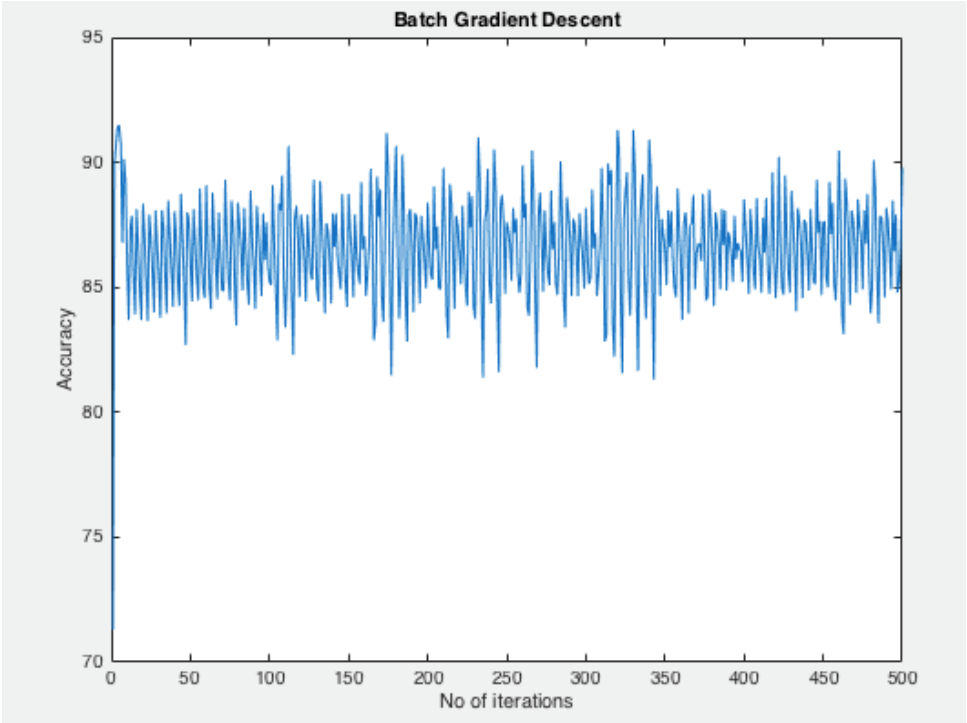
Batch Gradient for raw data.

Number of Iterations to meet glmfit accuracy: 873



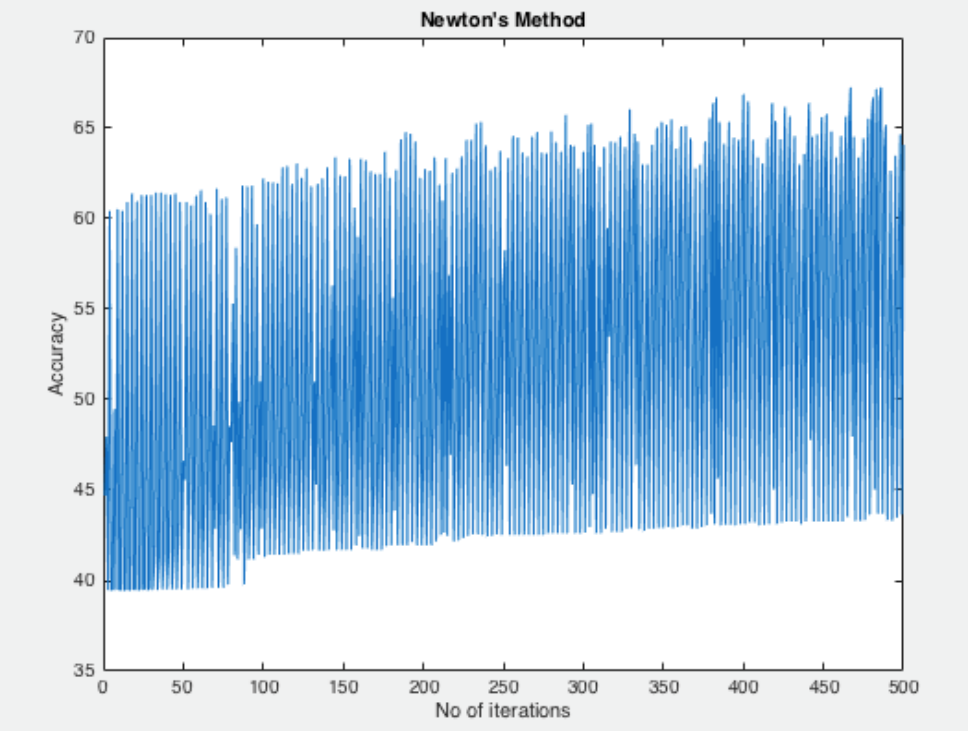
Batch Gradient for Normalized data.

Number of Iterations to meet glmfit accuracy: 2



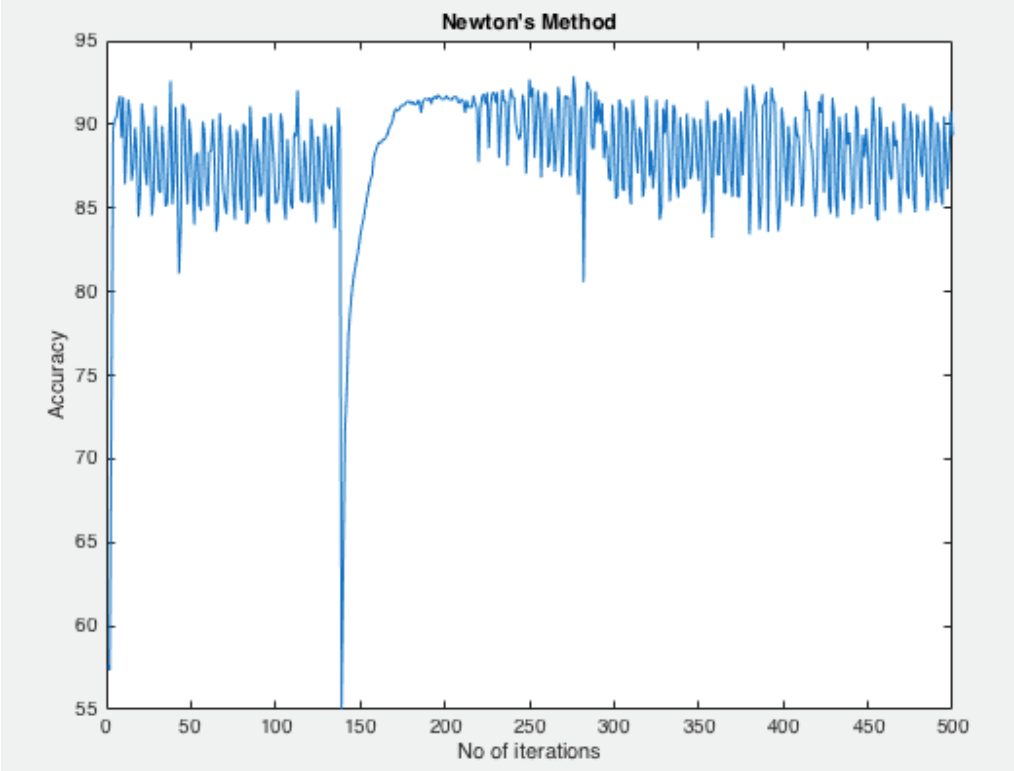
Newton's method for raw data

Number of Iterations to meet glmfit accuracy: 2454

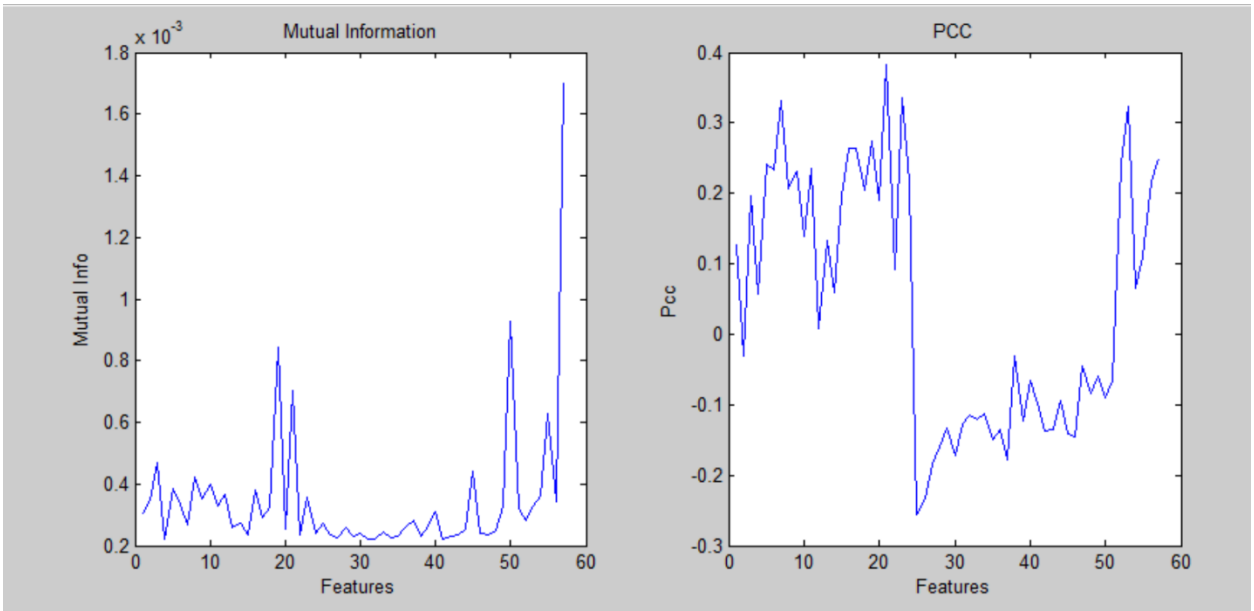


Newton's method for normalized data

Number of Iterations to meet glmfit accuracy: 1



C. Compare MI and PCC



D.

The 20 features with highest Mutual information with their ID's are
57, 50, 19, 21, 55, 3, 45, 8, 10, 5, 16, 12, 54, 23, 9, 2, 56, 6, 53, and 49.

Accuracy of training Set is 84.44% and test set is 82.82%

E.

the 3 features with lowest PCC are ID: 55, 23, and 3.

We have discretized the data using 3 equally sized bins using matlab linspace function.

Accuracy: 86.445%

Programming 4.2

A.

Model1.m1 = [1.5820 0.0186]

Model2.m2 = [-1.5299 1.5264]

Model3.m3 = [-1.4949 -1.4464]

Model1.S1 =

0.9506 0.0513

0.0513 4.0563

Model2.S2 =

1.0671 -0.5028

-0.5028 1.0630

Model3.S3 =

1.0393 0.5084

0.5084 1.0719

Model1.pi1 = 0.2857

Model2.pi2 = 0.2857

Model3.pi3 = 0.4286

Accuracy = 90.2286%

B.

Model1.m1 = [1.5820 0.0186]

Model2.m2 = [-1.5299 1.5264]

Model3.m3 = [-1.4949 -1.4464]

S1=S2=S3 =

[1.0219 0.0889

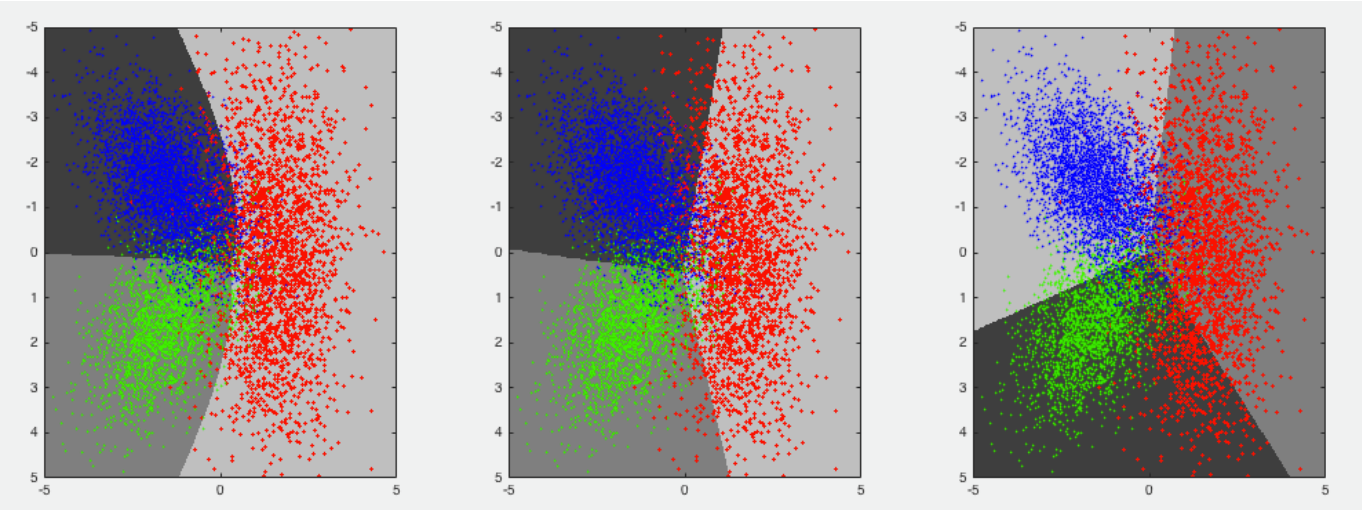
0.0889 1.9221]

Accuracy = 88.4457%

C.

Training the logistic regression using the standard Matlab function and the accuracy is 88.4571%.

D.

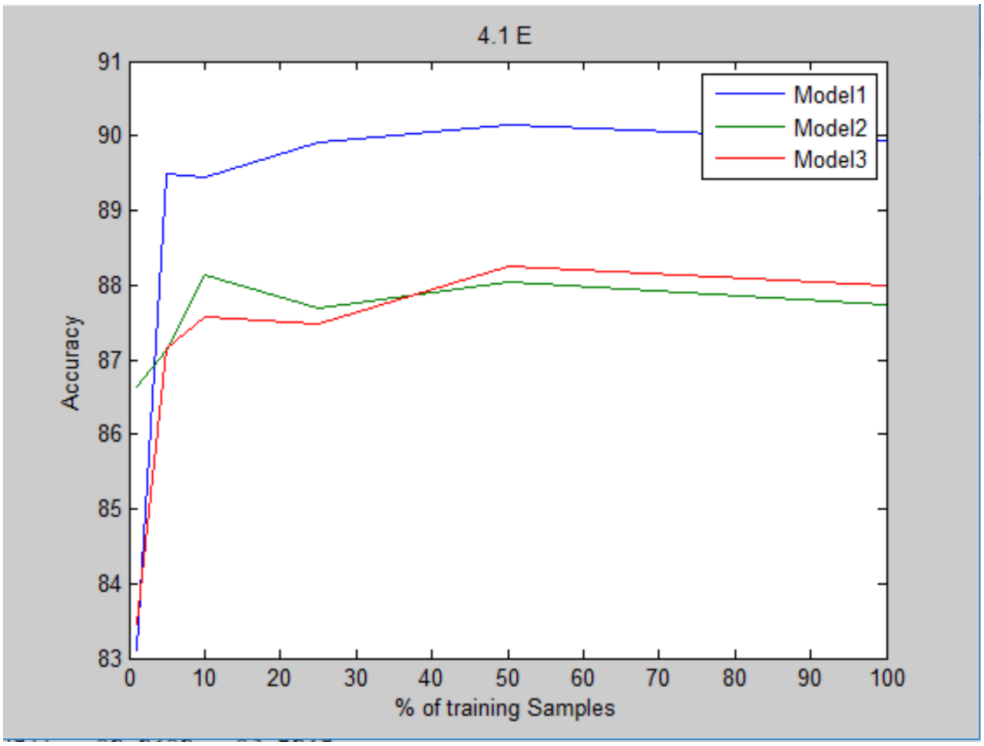


Model1 that is having independent S1, S2 and S3 is having highest accuracy. From the above experiment, we can clearly make the decision boundary between red, green and blue. With equal covariance, the posterior distribution of GMM has same form to logistic regression. Thus, model 1 with independent covariance allows better classification of data samples unlike model 2 and model 3.

E.

Accuracies for 1%, 5%, 10%, 25%, 50% and 100% for all the three models.

[83.0971 86.6286 83.4629
89.4857 87.1200 87.1543
89.4400 88.1371 87.5657
89.9200 87.6914 87.4743
90.1371 88.0457 88.2629
89.9429 87.7371 88.0000]



Collaboration: Ankhush