EECE5644 Spring 2020 – Take Home Exam 2

Chenglong Lin
lin.chengl@husky.neu.edu
NU ID: 001024390

February 24, 2020

1 Question 1

1. **Part 1**
The classifier that achieves minimum probability of error using the knowledge of the true pdf is the following:

$$\frac{P_1(x)}{P_0(x)} \lessgtr \frac{q_0(\lambda_{10}-\lambda_{00})}{q_1(\lambda_{01}-\lambda_{11})} \quad <: D=0 \quad >: D=1 \quad (eq.1)$$

Where q_0 is prior for label 0, q_1 is prior for label 1, λ is loss values.

$$q_0 = 0.9 \;\; q_1 = 0.1 \;\; \lambda_{10} = \lambda_{01} = 1 \;\; \lambda_{00} = \lambda_{11} = 0 \;\; (0-1 \;\; loss)$$

On the left side, $P_0(x)$ is multi-variate class conditional Gaussian pdf, $g(x|\mu_0, \Sigma_0)$, $P_1(x) = g(x|\mu_1, \Sigma_1)$. Multi-variate Gaussian pdf can be expressed as following:

$$g(x|\mu, \Sigma) = (2\pi)^{\frac{-k}{2}} det(\Sigma)^{\frac{-1}{2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$$

In this problem, our Gaussian pdfs parameters are:

$$\mu_0 = \begin{bmatrix} -2 \\ 0 \end{bmatrix} \quad \mathbf{\Sigma}_0 = \begin{bmatrix} 1 & -0.9 \\ -0.9 & 2 \end{bmatrix}$$
$$\mu_1 = \begin{bmatrix} 2 \\ 0 \end{bmatrix} \quad \mathbf{\Sigma}_1 = \begin{bmatrix} 2 & 0.9 \\ 0.9 & 1 \end{bmatrix}$$

We can plug in these values into eq.1 to get the classifier that achieves minimum probability of error.
I used these parameters to draw samples for these 4 different datasets, $D^{10}_{train}, D^{100}_{train}, D^{1000}_{train}, D^{10k}_{validate}$
I applied this classifier on $D^{10k}_{validate}$ dataset and drew its ROC curve, which

I also marked the minimum p(error) point in green:

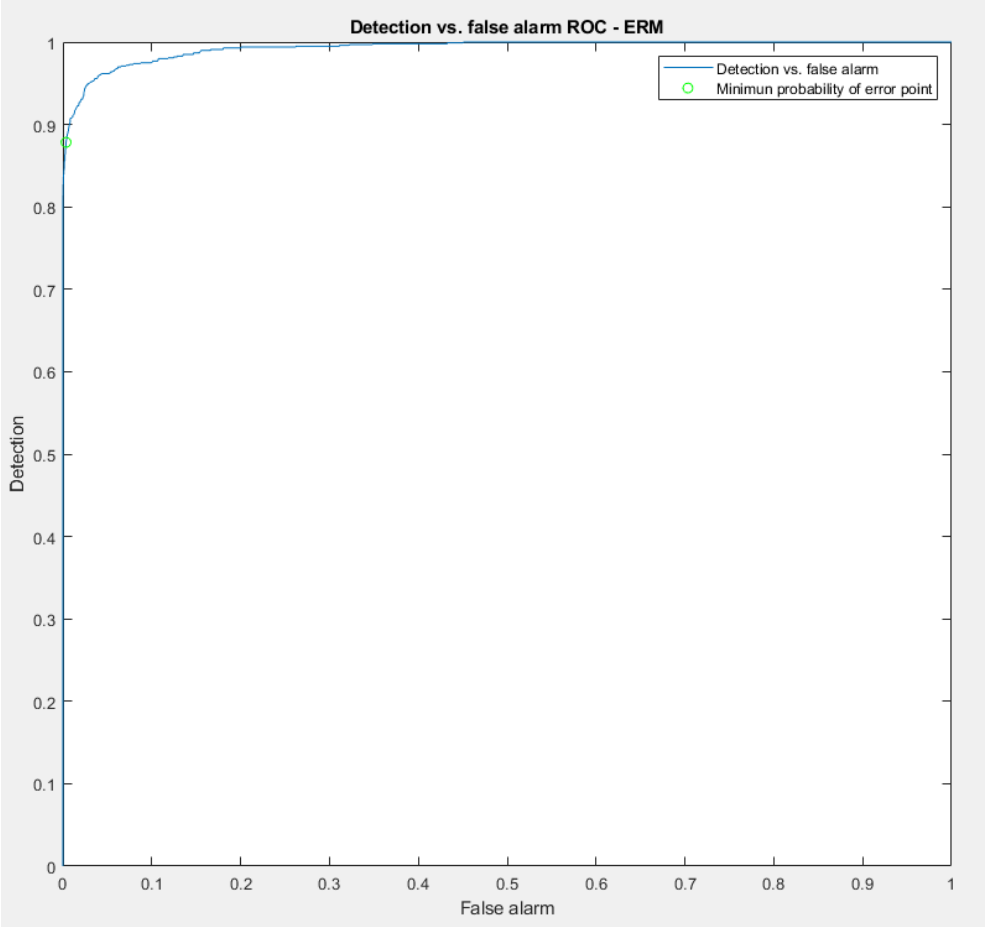

Figure 1: ROC curve on $D^{10k}_{validate}$

The decision boundary of this classification rule overlaid on $D^{10k}_{validate}$:

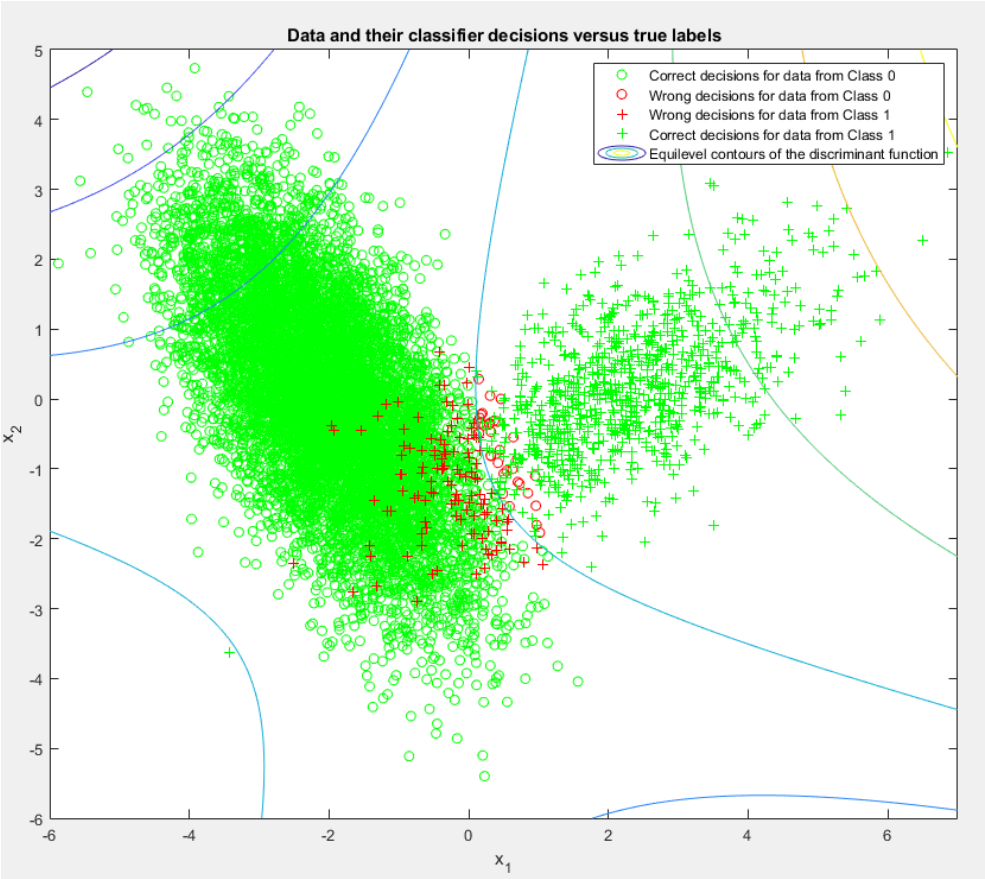

Figure 2: Decision boundary on $D^{10k}_{validate}$

By using this classifier, I was able to obtain min-P(error) of 1.39%. The min-P(error) is calculated using following formula:

$$\frac{(p_{10} * n_0 + p_{01} * n_1)}{N}$$

Where p_{10} is probability of false positive; p_{01} is probability of false negative; n_0 is the number of label 0; n_1 is the number of label 1; N is the number of total samples/labels.

2. **Part 2**
The negative-log-likelihood minimization problem (loss function) to be solved by gradient descend can be expressed as following:

$$argmin_\theta \quad -\frac{1}{N}(\Sigma^N_{x_i:L_i=0} ln(1 - h(x_i, \theta)) + \Sigma^N_{x_i:L_i=1} ln(h(x_i, \theta))) \quad (eq.2)$$

where N is total number of sampels, and $h(x_i, \theta)$ is sigmoid function:

$$h(x_i, \theta) = \frac{1}{1 + e^{-\theta * Z(x)}}$$

In this problem, we are doing logistic-linear approximation, so our θ is [b w1 w2] and our Z(x) is $[1 \quad x1 \quad x2]^T$
The negative-log-likelihood minimization was obtain by taking natural log on maximization of $P(D|\theta)$, and by applying the bayes theorem, and by modeling the output using the sigmoid function. The detailed derivation of this negative-log-likelihood minimization formula was shown in the class [1]. I then used gradient descend to iteratively minimize this cost function and to obtain parameter θ for the linear classifier.
The linear classifier is in form of following formula:

$$y = \theta * Z(x) = w1 * x1 + w2 * x2 + b$$

To make decision using this linear classifier, I set y to zero, because we are drawing contours with two dimensions x1, x2. By arranging some variables. We get the following.

$$x2 = \frac{-w2x1 - w1}{w3}$$

By plugging in x1's from the data samples into this formula. We obtain a set of x2's, we then compare it with the x2 true from the data samples. Depending on the sign of the coefficient $-\frac{w2}{w3}$ of this classifier (orientation of the decision line), we can either classify the left side of the line to be

label 0 or right side of the line to be label 0. In this problem, when this coefficient is positive, we decide the sample to be label 1 when calculated x2 is larger than sample x2 value, label 0 when smaller than. On the other hand, when this coefficient is negative, we decide the sample to be label 0 when calculated x2 is larger than sample x2 value, label 1 when smaller than. The following is probability of error of linear classifiers obtained from $D^{10}_{train}, D^{100}_{train}, D^{1000}_{train}$ datasets on $D^{10k}_{validate}$ dataset:

Training set	**Probability of error on** $D^{10k}_{validate}$
D^{10}_{train}	2.48%
D^{100}_{train}	1.76%
D^{1000}_{train}	1.61%

We can see that as number of sample for training increased, the probability of error decreased. This is expected. The following are supplementary visualization of decision boundaries of these trained classifiers superimposed on their respective training datasets and validation dataset.

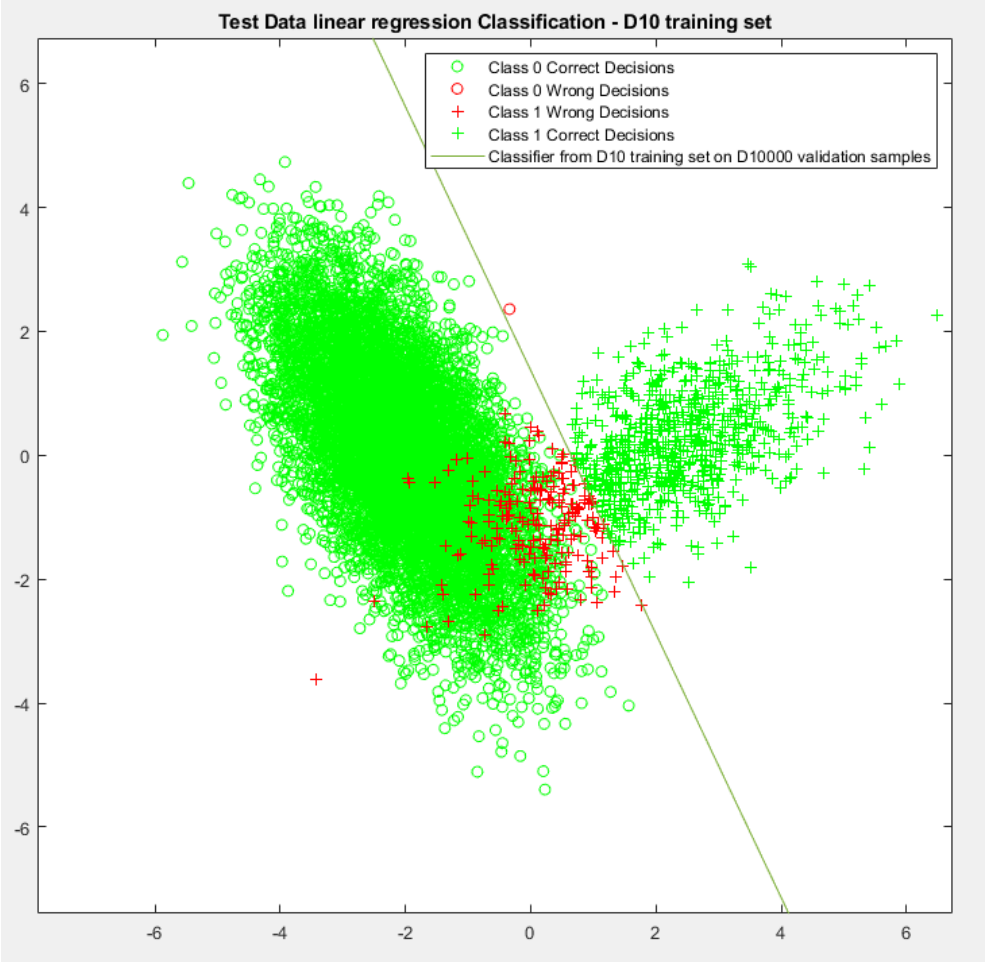

Figure 3: D^{10}_{train} linear classifier on $D^{10k}_{validation}$, x2 vs x1

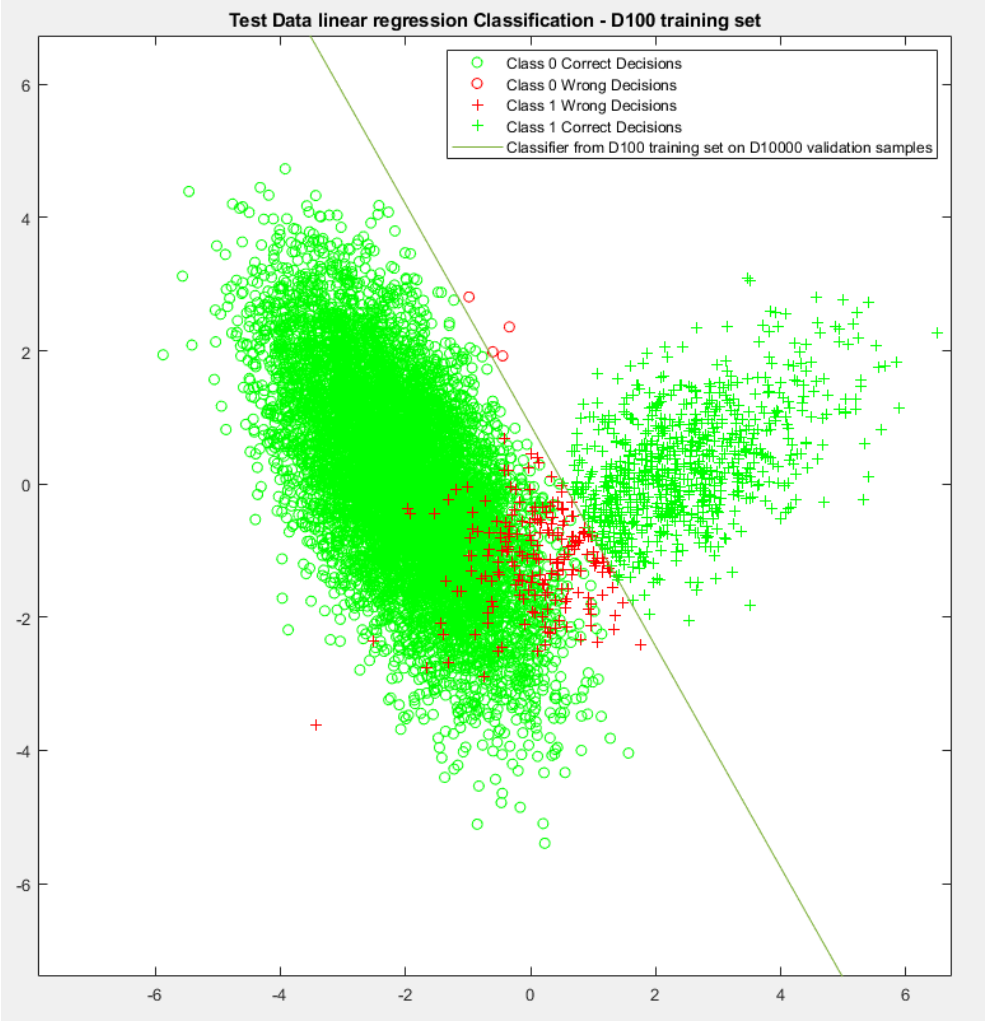

Figure 4: D^{100}_{train} linear classifier on $D^{10k}_{validation}$, x2 vs x1

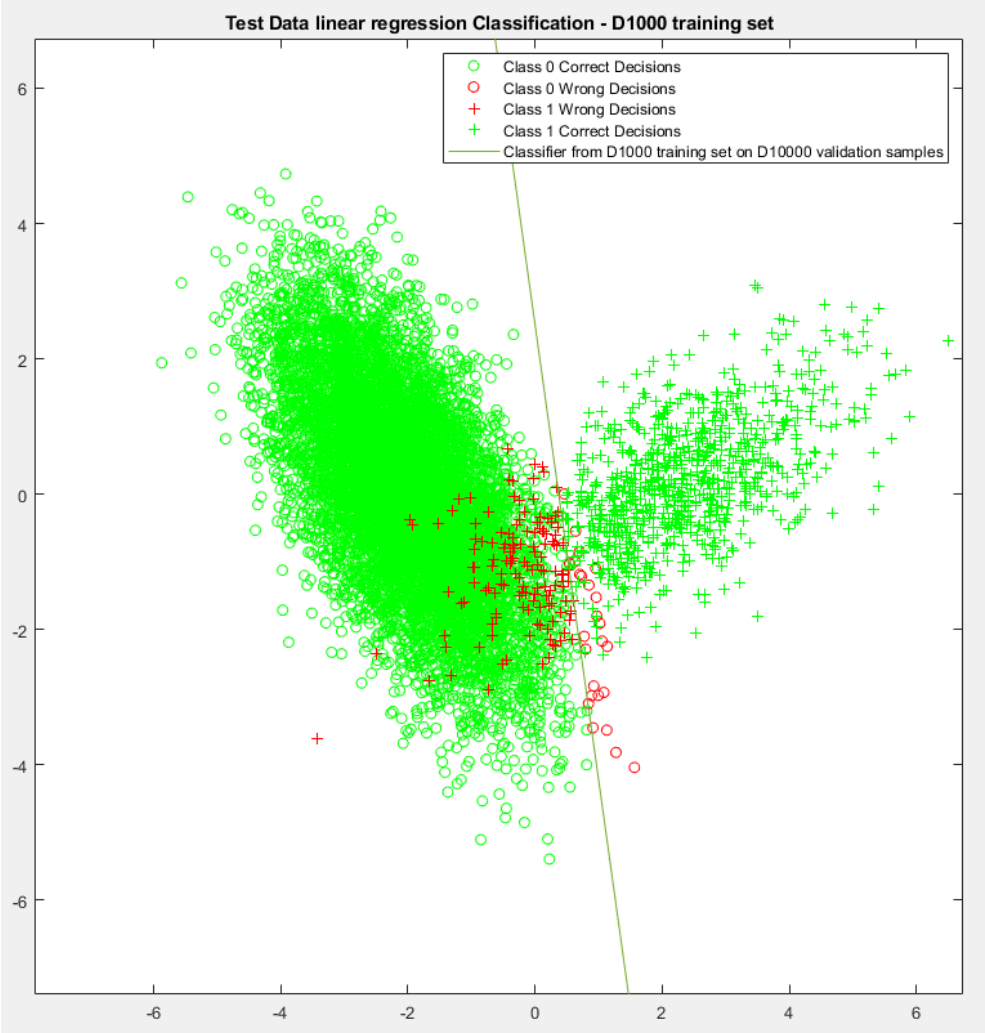

Figure 5: D^{1000}_{train} linear classifier on $D^{10k}_{validation}$, x2 vs x1

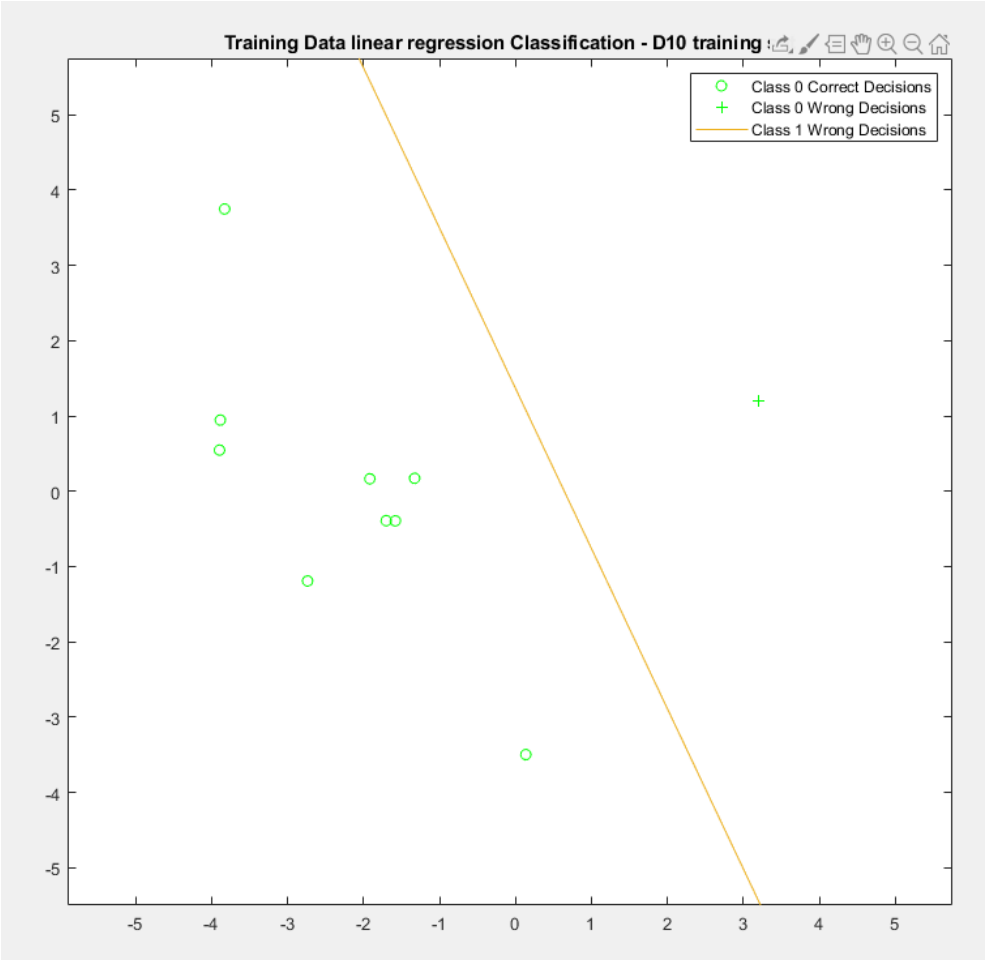

Figure 6: D^{10}_{train} linear classifier on D^{10}_{train}, x2 vs x1

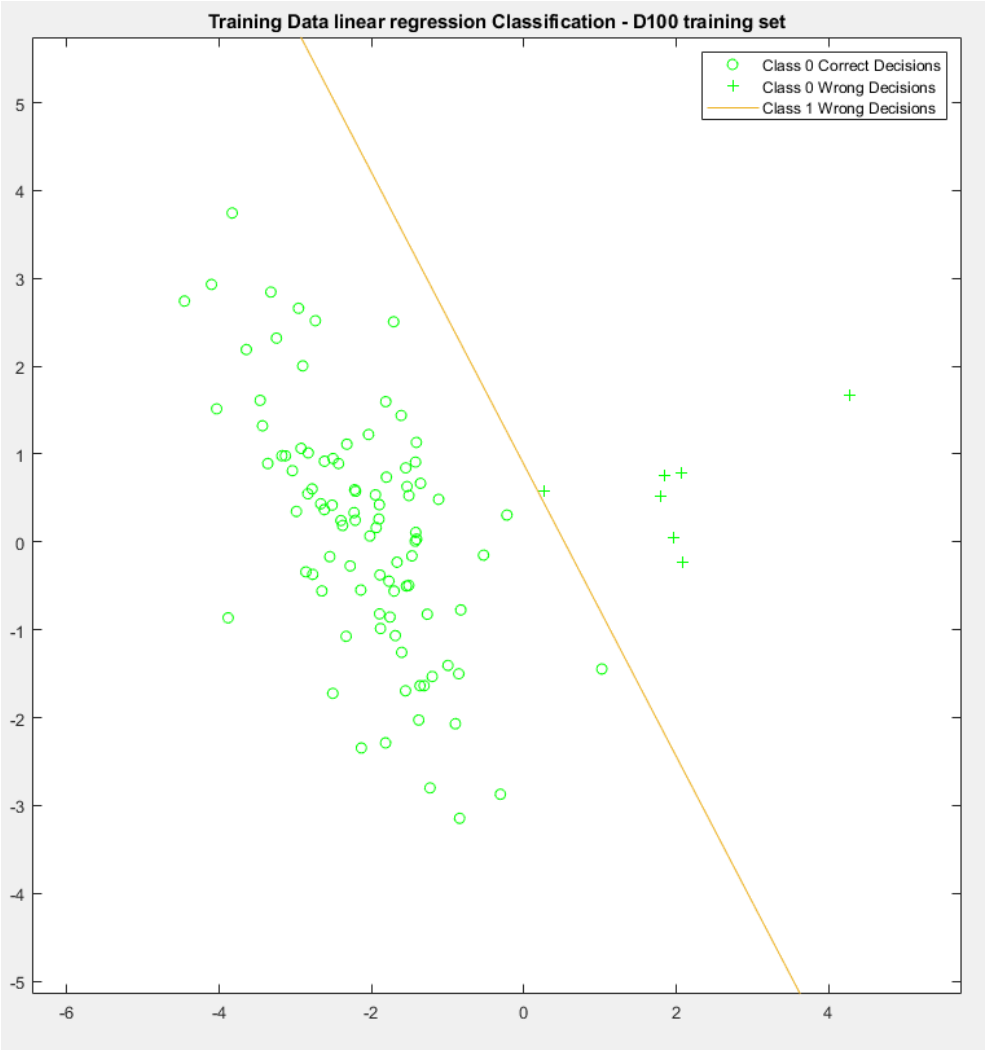

Figure 7: D_{train}^{100} linear classifier on D_{train}^{100}, x2 vs x1

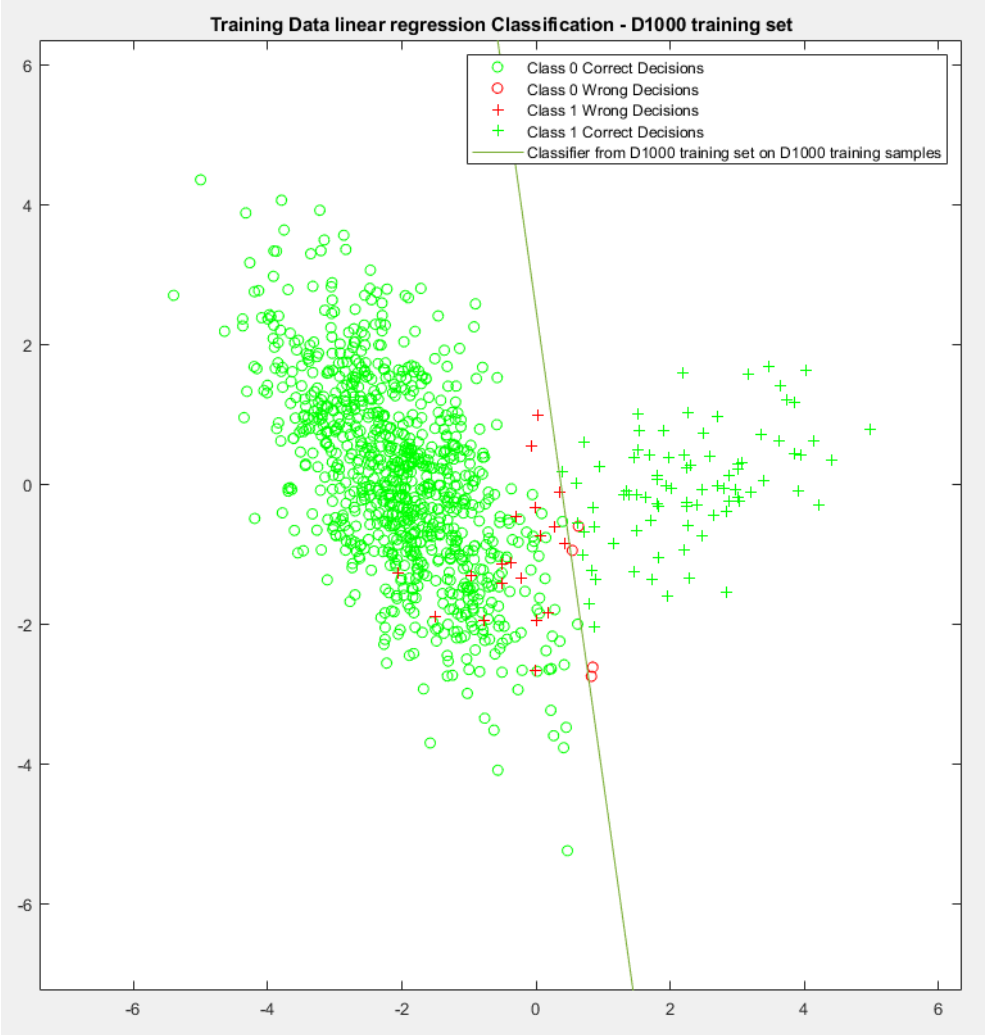

Figure 8: D_{train}^{1000} linear classifier on D_{train}^{1000}, x2 vs x1

3. **Part 3**

The negative-log-likelihood minimization problem (loss function) for logistic quadratic approximation is same as eq. 2 above. However, because we are approximating a quadratic boundary, we need to add quadratic terms into our Z(x) and θ. After adding quadratic terms, we have $Z(x) = [1 \quad x1 \quad x2 \quad x1^2 \quad x1x2 \quad x2^2]^T$ and $\theta = [b \quad w1 \quad w2 \quad w3 \quad w4 \quad w5]$. Therefore, the quadratic classifier is in form of following formula:

$$y = \theta * Z(x) = w5 * x2^2 + w4 * x1x2 + w3 * x1^2 + w2 * x2 + w1 * x1 + b$$

To make decision using this quadratic classifier, I plugged in x1's and x2's from the data samples into this formula. After which we will obtain a set of y's. For data samples that generates positive y value, we classify it as label 1. On the other hand, for data samples that generates negative y value, we classify it as label 0.

$$w5 * x2^2 + w4 * x1x2 + w3 * x1^2 + w2 * x2 + w1 * x1 + b \lessgtr 0$$
$$>: D = 1 \quad <: D = 0$$

The following is probability of error of quadratic classifier obtained from $D^{10}_{train}, D^{100}_{train}, D^{1000}_{train}$ datasets on $D^{10k}_{validate}$ dataset:

Training set	**Probability of error on** $D^{10k}_{validate}$
D^{10}_{train}	2.65%
D^{100}_{train}	1.69%
D^{1000}_{train}	1.41%

We can see the same trend, as the the number of sample for training increased, the probability of error decreased. On the other note, we can see that by using a quadratic classifier, we obtained better accuracy compared to the linear classifier while using the same training dataset. This is expected as part of the Gaussian overlapped, forming a oval shape boundary. Therefore, a quadratic classifier should perform better than a linear one. The following are supplementary visualization of quadratic decision boundaries of these trained classifiers superimposed on their respective training

datasets and validation dataset.

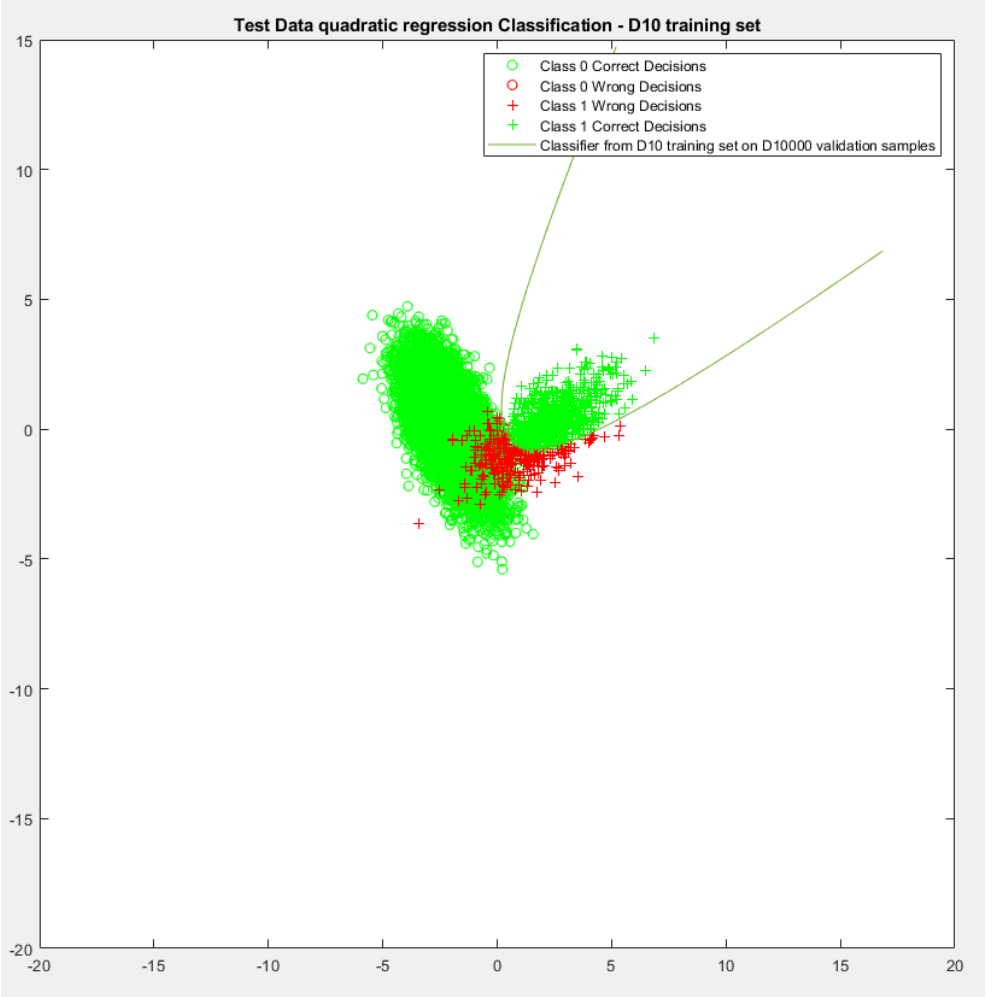

Figure 9: D^{10}_{train} quadratic classifier on $D^{10k}_{validation}$, x2 vs x1

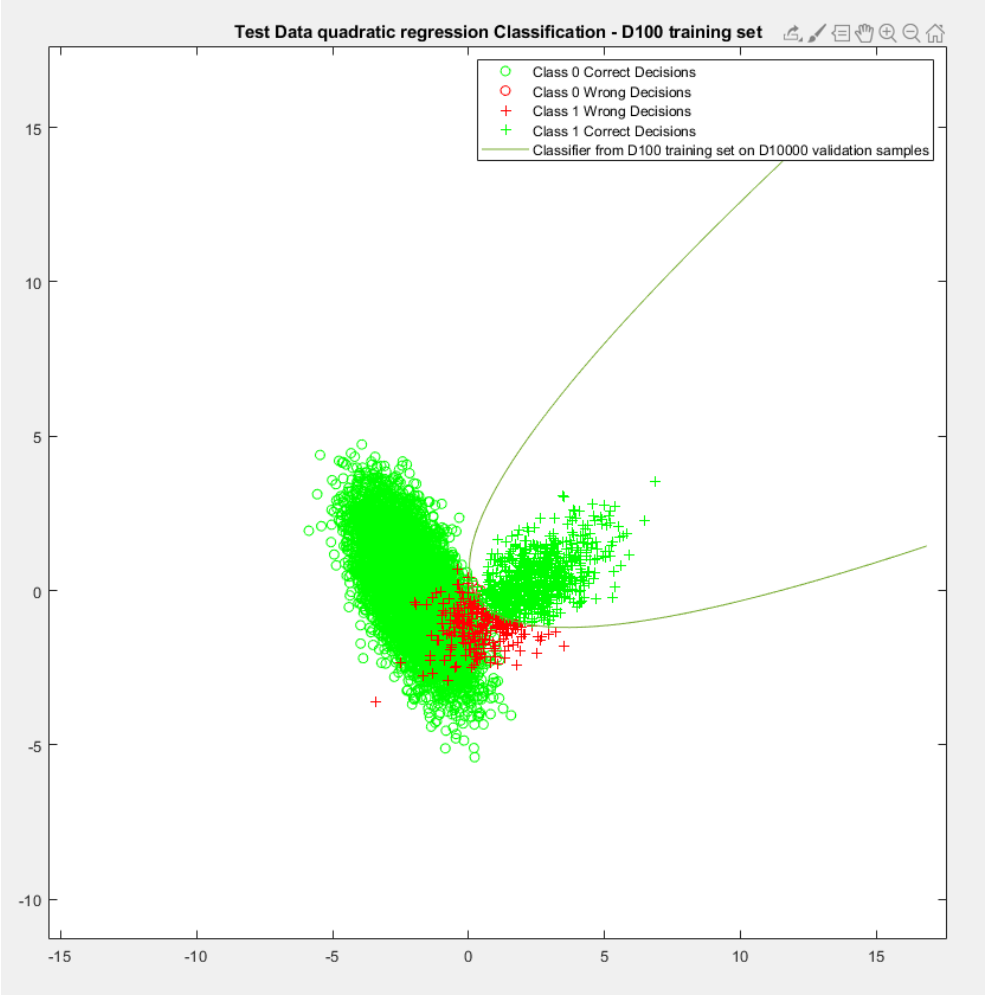

Figure 10: D^{100}_{train} quadratic classifier on $D^{10k}_{validation}$, x2 vs x1

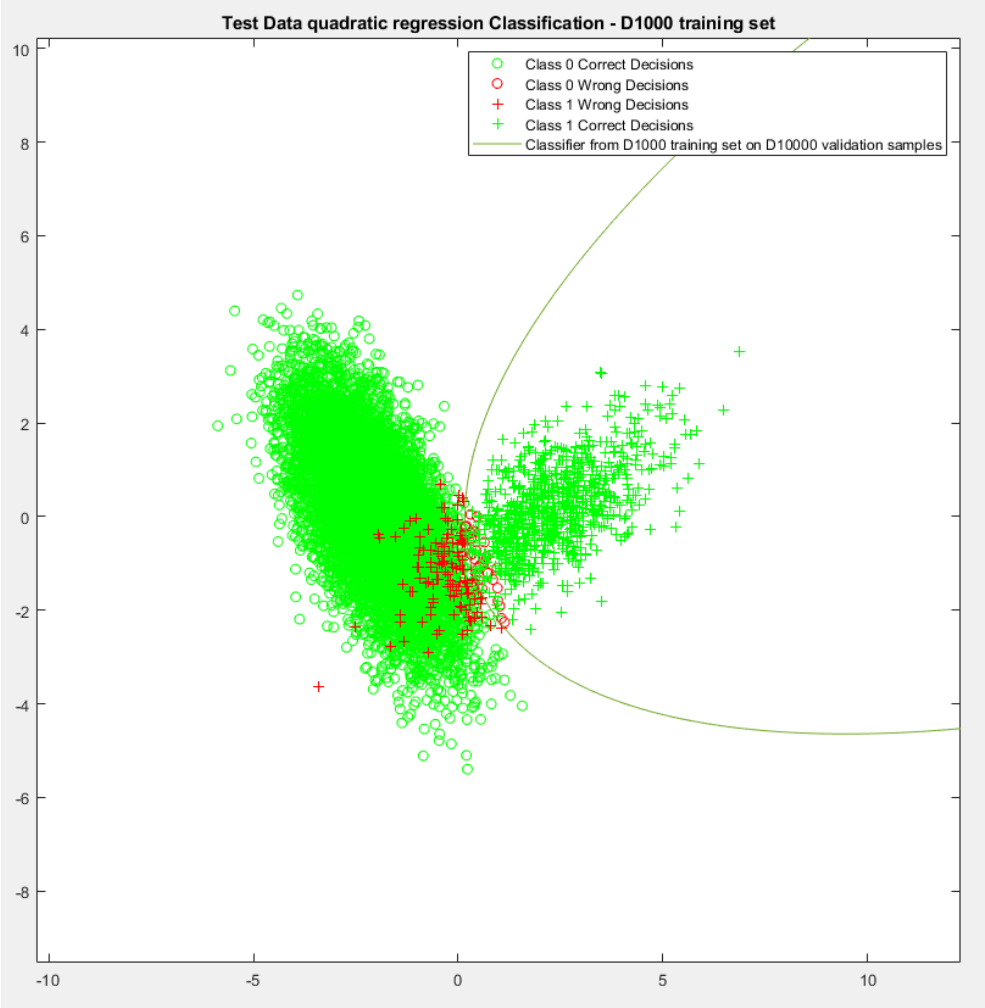

Figure 11: D^{1000}_{train} quadratic classifier on $D^{10k}_{validation}$, x2 vs x1

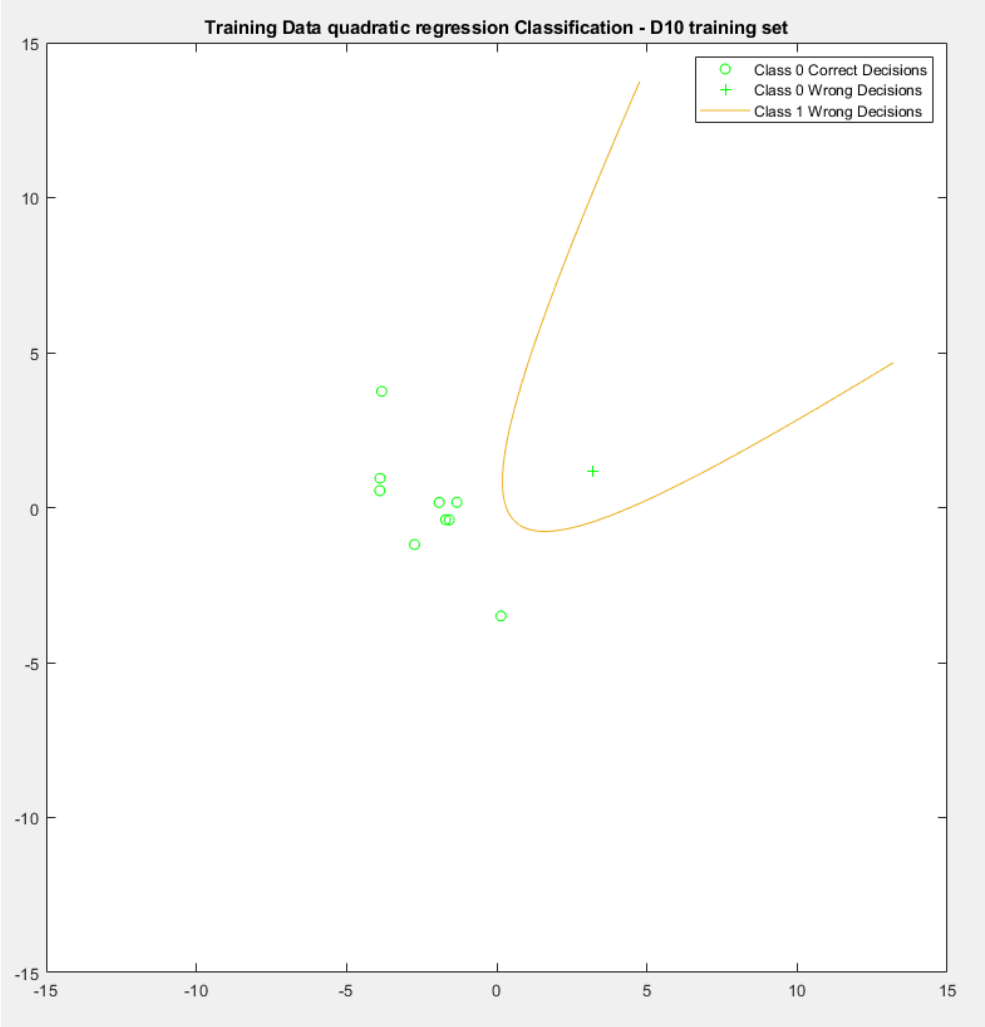

Figure 12: D^{10}_{train} quadratic classifier on D^{10}_{train}, x2 vs x1

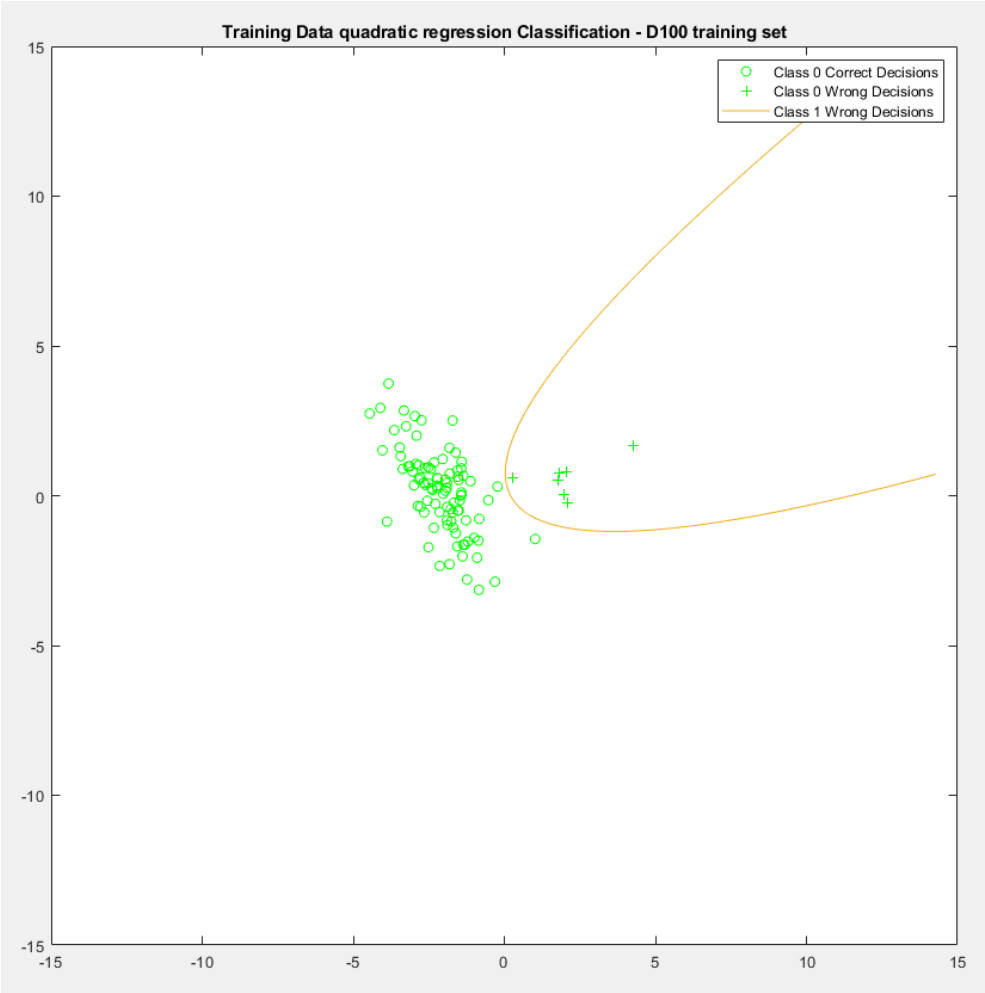

Figure 13: D_{train}^{100} quadratic classifier on D_{train}^{100}, x2 vs x1

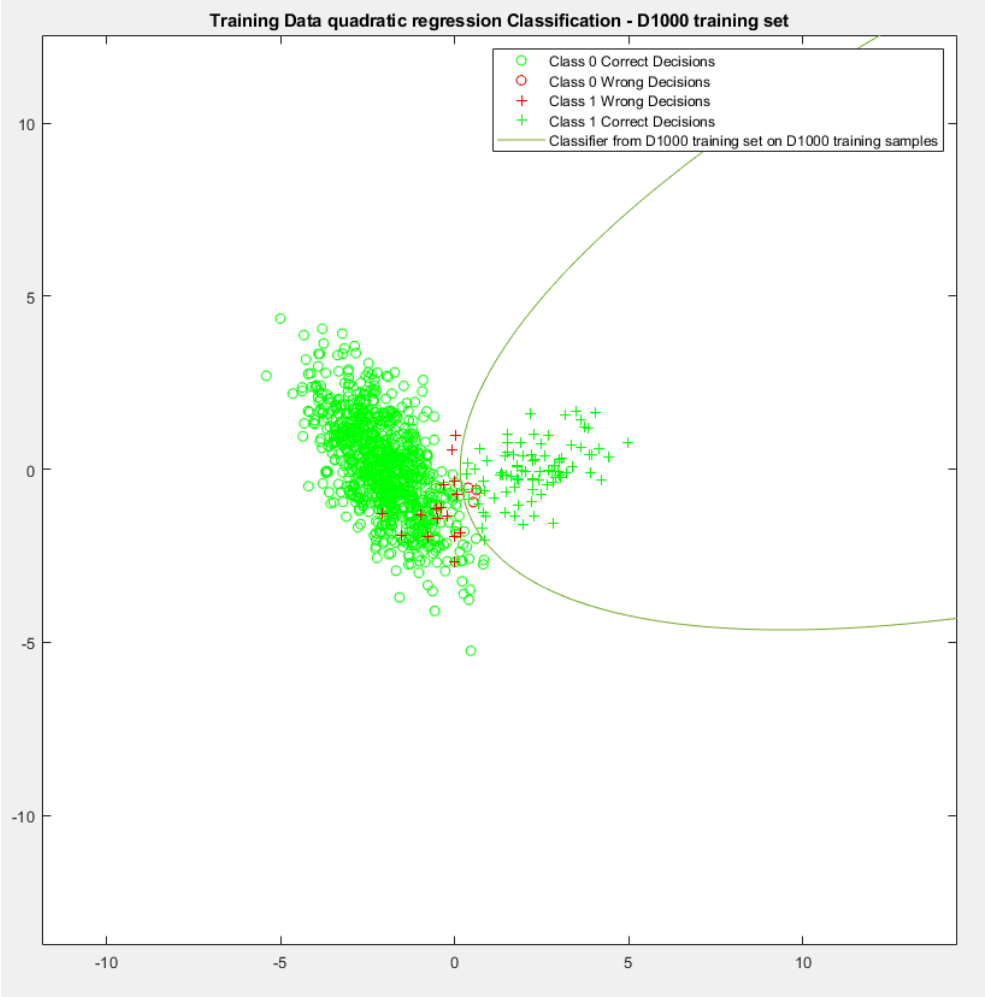

Figure 14: D^{1000}_{train} quadratic classifier on D^{1000}_{train}, x2 vs x1

2 Problem 2

In this problem, we are asked to pick true parameters w for the cubic polynomial with roots in the interval [-1,1]. i.e.

$$y = a(x - r1)(x - r2)(x - r3)$$

where a is a nonzero real value, and r1, r2, r3 are real values between -1 and +1. I choose r1=-1, r2=1, r3=1, a=1.5. After some basic algebra. I obtained:

$$y = 1.5x^3 - 1.5x^2 - 1.5x + 1.5$$

Therefore:

$$\theta_{true} = [1.5 \quad -1.5 \quad -1.5 \quad 1.5]^T$$

My Gaussian noise is drawn from

$$v \in N(0, \sigma_v^2) \quad \sigma_v = 0.5$$

In this problem I choose B=5. To generate my γ's logarithmically, I used logspace(-B,B) function from Matlab. This function generates logarithmically spaced vector from 10^{-B} to 10^B.
Our iid samples x is drawn from an uniform distribution:

$$x \in Uniform[-1, 1]$$

The MAP estimator formula for estimating parameter θ_{MAP} is as following:

$$\theta_{MAP} = (\sum_{i=1}^{N} y_i Z_i))(\sum_{i=1}^{N} Z_i Z_i^T + \frac{\sigma_v^2 I}{\gamma^2}) \quad (eq.3)$$

Which y is the y value corrupted by the Gaussian noise, $Z = [1 \;\; x \;\; x^2 \;\; x^3]^T$, and I is identity matrix, N is number of samples. This MAP estimator formula is derived by adding prior terms to the ML estimator. The detailed derivation can be found in [2].
For every γ value, I did 100 experiments generating θ_{MAP} using eq. 3 from above, and taking squared L_2 distance between the true parameters θ_{true} and θ_{MAP}. These squared-error values can be calculated by using following formula:

$$squared - error = \frac{\sum_i^N (\theta_{true,i} - \theta_{MAP,i})^2}{N}$$

The following is the plot for these squared-error values vs gamma:

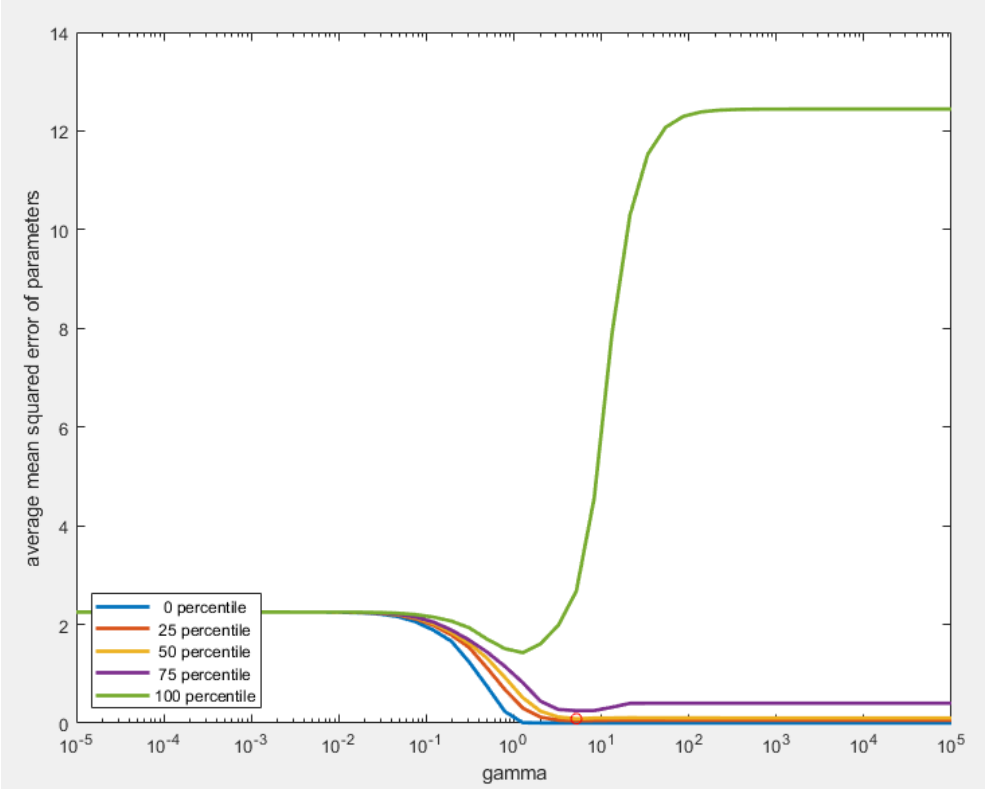

Figure 15: Squared-error values vs γ

From this plot, you can see that when γ is small, all MAP estimation is consistent, and have same square error. At medium gamma value, the square error of MAP estimation starts to diverge. As the value of gamma increases to infinity, all curves converge to their own values as the MAP estimator reduced to ML estimator because the prior term becomes zero.

For the sake of experiment, I also drew the original, max, min, median error polynomials.

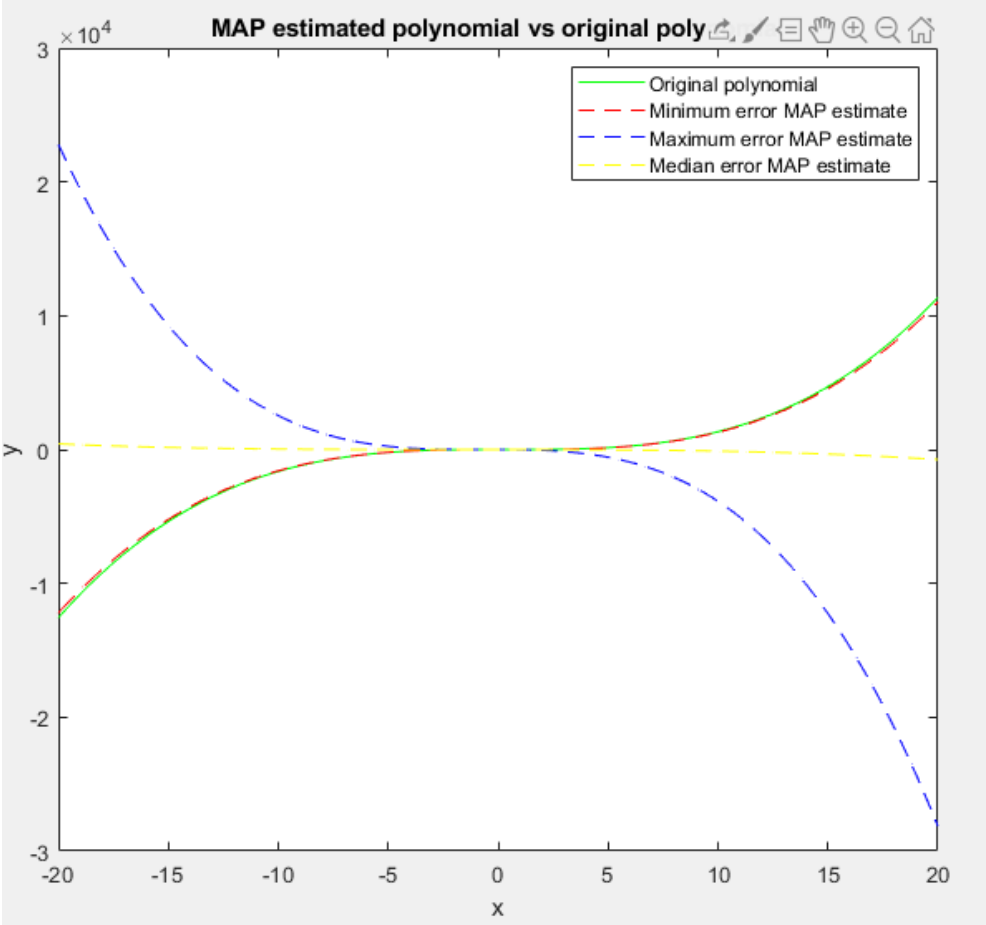

Figure 16: Polynomials

3 Problem 3

In this problem, we need to initialize 4 Gaussian components. I picked following α's, μ's and γ's for these components. α is prior of the component. μ is mean, and γ is covariance of the component.

$$\alpha_{true} = [0.24 \quad 0.23 \quad 0.26 \quad 0.27] \quad \mu_{true} = \begin{bmatrix} -5 & 2 & 6 & 0 \\ 0 & -5 & 3 & 8 \end{bmatrix}$$

$$\mathbf{\Sigma}_{true,1} = \begin{bmatrix} 3 & 1 \\ 1 & 5 \end{bmatrix} \quad \mathbf{\Sigma}_{true,1} = \begin{bmatrix} 5 & 1 \\ 1 & 4 \end{bmatrix} \quad \mathbf{\Sigma}_{true,1} = \begin{bmatrix} 4 & 1 \\ 1 & 5 \end{bmatrix} \quad \mathbf{\Sigma}_{true,1} = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}$$

I am choosing similar prior for each components because I would like to make the EM easier to figure out correct component numbers. If one component's prior is small, it's less likely to appear in the data sample, so it's less likely to be identified. My choice of μ's and Σ's would make Gaussian components somewhat close, but not overlaping too much. If the Gaussian components overlaps too much, EM would more likely identify them as a single component. On the other hand, if the Gaussian components are too far away and covariance of the component is large, EM would more likely to identify it as multiple components.

Bootstrapping

Since we are estimating maximum 6 Gaussian components, we are estimating 12 parameters from μ's and 24 parameters from Σ's. In total, we are estimating 36 parameters. I decided to draw 20 samples for each parameters when bootstraping. Therefore, I am drawing 720 samples for my training and validation set.

Gaussian noise

In this problem, we are trying to estimate number of Gaussian components from 3 different sample data set, D10, D100, D1000. When bootstraping from these datasets, I introduced a small Gaussian noise ($v \in N(0, 1e^{-4})$) to the drawn sample, reducing repetition and prevent it being exactly replicated.

EM reinitialization

When initializing the EM, I randomly choose samples from the training set as our initial mean estimates and get covariance based on data samples near these initial mean estimates. This could cause the algorithm to stall when the mean of each components are poorly selected. In my design, I wrote code to detect stalled EM process and reinitialize it to find another way to the global optimum. The following is table of average initialization needed for each attempt to seek for the global optimum in each dataset.

Training set	**Average initializations needed**
D_{train}^{10}	5.17
D_{train}^{100}	4.09
D_{train}^{1000}	3.38

We can see that D1000 dataset needs less initialization. It's expected because dataset with more data samples has more dynamic range, and is less likely to get bad initializations.

Threshold

The original delta, EM stopping criterion, could causing the EM to stall when set too small. I adjusted mine to 0.3, it works best in case of my GMM example. I also adjusted the covariance regularization parameter to $1e^{-5}$.

Log-likelihood performance

After the EM converges, I used Log-likelihood performance formula to determine the performance of the estimated GMM model. The formula is the following:

$$\sum_{i}^{N} ln(\sum_{i}^{N} \alpha_i * g(x|\mu_i, \Sigma_i))$$

Where α is prior, $g(x|\mu_i, \Sigma_i)$ is Gaussian PDF, and N is number of Gaussian components. This formula came from Professor's EMforGMM code [3].

Result

After loop through 100 experiments x 10 bootstrapped training/validation set x 6 candidates x 3 different datasets, I got the result for each dataset's EM decisions. The result was obtain by averaging all log-likelihood performance value for each candidate component, and find the index of the max average performance. The index number is the winner components.

Training set	**Decision**
D_{train}^{10}	6
D_{train}^{100}	6
D_{train}^{1000}	4

Training set	**Winner count [1 2 3 4 5 6]**
D_{train}^{10}	[0 0 0 0 0 100]
D_{train}^{100}	[0 0 0 0 1 99]
D_{train}^{1000}	[0 0 16 44 40 0]

For this EM on GMM model, I was able to get correct GMM components using D1000 training set. D10 and D100 were incorrect due to limited samples.

4 References, Citations, and Acknowledgment

[1] Panopto video, EECE5644 Thu, 6th Feb at 1:08:00 to 1:30:00. [Online]. Boston, MA, 2020.
[2] Panopto video, EECE5644 Thu, 6th Feb at 53:00 to 1:05:00. [Online]. Boston, MA, 2020.
[3] EECE56442020sprSharedFolder/Code/EMforGMM, Northeastern University, Feb 24, 2020.
Accessed on: Feb 24, 2020. [Online]. Available: https://drive .google.com/drive/folders/1BJidoDDb6ifRs48MjPyGAj1RD3VxXvJi?usp=sharing
[4] EECE56442020sprSharedFolder/Code/VolunteerExamples/ParameterEstimation, ClassificationExamplev2, Northeastern University, Feb 24, 2020. Accessed on: Feb 24, 2020. [Online]. Available: https://drive.google.com/drive/folders/1BJidoDDb6ifRs48MjPyGAj1RD3VxXvJi?usp=sharing

5 Git repo

https://github.com/MicroRAsus/Machine-Learning-Pattern-Recognition-HW2