EE 5239 Nonlinear Optimization Homework 4 Cover Sheet

Instructor name: Mingyi Hong

Student name: Graham Droge

- Date assigned: Thursday 10/1/2019
- Date due: Tuesday 10/13/2019, at midnight
- This cover sheet must be signed and submitted along with the homework answers on additional sheets.
- By submitting this homework with my name affixed above,
  - I understand that late homework will not be accepted,
  - I acknowledge that I am aware of the University's policy concerning academic misconduct (appended below),
  - I attest that the work I am submitting for this homework assignment is solely my own, and
  - I understand that suspiciously similar homework submitted by multiple individuals will be reported to the Dean of Students Office for investigation.
- Academic Misconduct in any form is in violation of the University's Disciplinary Regulations and will not be tolerated. This includes, but is not limited to: copying or sharing answers on tests or assignments, plagiarism, having someone else do your academic work or working with someone on homework when not permitted to do so by the instructor. Depending on the act, a student could receive an F grade on the test/assignment, F grade for the course, and could be suspended or expelled from the University.

Problem 3.1.6

For this problem we will use the theorem of Lagrange Multipliers, so we will start by writing the Lagrangian equation

$$f(x) = \prod_{i=1}^{n} x_i^{a_i} \quad g(x) = 0 = \sum_{i=1}^{n} x_i - 1$$

$$L(x, \lambda) = f - \lambda(\sum_{i=1}^{n} x_i - 1)$$

Now using the fact that $\boldsymbol{\nabla} L = 0$ we get on an element wise basis

$$a_i x_i^{a_i - 1} \prod_{j \neq i}^{n} x_j^{a_j} - \lambda = 0$$

$$a_i \frac{f}{x_i} - \lambda = 0 \rightarrow \lambda = a_i \frac{f}{x_i} \rightarrow x_i = a_i \frac{f}{\lambda}$$

Now we can plug the equation for x_i into $g(x)$ and then rearrange to get put it as an expression for λ

$$\sum_{i=1}^{n} a_i \frac{f}{\lambda} - 1 = \sum_{i=1}^{n} a_i - \frac{\lambda}{f} = 0$$

$$\lambda = f \sum_{i=1}^{n} a_i$$

Now using the expression for λ we can plug it back in to the expression above to get an expression for x_i

$$f \sum_{i=1}^{n} a_i = a_i \frac{f}{x_i} \quad \rightarrow \quad x_i = \frac{a_i}{\sum_{i=1}^{n} a_i}$$

3.3.1

For this problem we need to show that the minimizing x of

$$f(x)_1 = argmin_{x \in X}(\boldsymbol{\nabla} f(x^k)'(x - x^k) + \frac{1}{2s^k}(x - x^k)' H^k (x - x^k))$$

is the same one that minimizes

$$f(x)_2 = \frac{1}{2} \left\| x - (x^k - s^k (H^k)^{-1} \boldsymbol{\nabla} f(x^k)) \right\|_{H^k}^2$$

To do this we can find the the minimizing x for each function and compare if they are the same. So we compute the gradients of both functions with respect to the vector x

$$\boldsymbol{\nabla}_x f_1 = \boldsymbol{\nabla}_x \{\boldsymbol{\nabla} f(x^k)' x - \boldsymbol{\nabla} f(x^k) x^k + \frac{1}{2s^k}(x - x^k) H^k (x - x^k)\}$$

$$\boldsymbol{\nabla}_x f_1 = \boldsymbol{\nabla} f(x^k) + \frac{1}{s^k} H^k (x - x^k)$$

Now setting to 0 and solving for x we obtain

$$x^* = x^k - (H^k)^{-1} s^k \nabla f(x^k) \quad (1)$$

Now we will move on to the other equation

$$\nabla_x f_2 = \frac{\partial f_2}{\partial z} \frac{\partial z}{\partial x}$$

$$\frac{\partial f_2}{\partial z} = \frac{\partial}{\partial z} \{ \frac{1}{2} z^{'} H^k z \} = H^k z$$

$$\frac{\partial z}{\partial x} = \frac{\partial}{\partial x} \{ x - x^k + s^k (H^k)^{-1} \nabla f(x^k) \} = 1$$

$$\nabla_x f_2 = \frac{\partial f_2}{\partial z} \frac{\partial z}{\partial x} = H^k z = H^k (x - x^k + s^k (H^k)^{-1} \nabla f(x^k))$$

Finally distributing H^k through, setting the equation equal to 0 and rearranging we obtain (2) below. As we can see (2) and (1) are equal implying that $\bar{x}^k = x^*$ which equals the minimizer of $f(x)_2$. Since the constraints are are consistent in both we don't have to worry about that in the final answer.

$$H^k x = H^k x^k - s^k \nabla f(x^k)$$

$$x^* = x^k - (H^k)^{-1} s^k \nabla f(x^k) \quad (2)$$

3.1.12

For this problem we need to develop an algorithm for find the projection of a vector z on a simplex. We can start by using Lagrange Multipliers.

$$L(x, \lambda) = \frac{1}{2} \|x - z\|^2 + \lambda (\mathbf{1} x - 1)$$

$$L(x, \lambda) = \sum (\frac{1}{2} (x_i - z_i)^2 + \lambda x_i) - \lambda$$

Using the KKT conditions we can solve for λ and x_i. Taking the partial and setting it to zeros gives $x_i^* = (z_i - \lambda)$ now we can plug that into the equality constraints to obtain

$$\sum (z_i - \lambda) - 1 = 0$$

What this tells us is that we need to find a λ that satisfies the above equation while also satisfying the constraints that $z_i > 0$. We can simply perform a bisection method to find the λ that satisfies the above equation and once we have that the optimal x vector is just the subtraction of λ from z.

4.1.1

For the following problems we use the theorem of Lagrange Multipliers. So we will set up the Lagrangian and calculate partial derivatives and set them to zero. Then solving for the given x vectors

(a)

$$f(x) = \|x\|^2 \quad \sum_{i=1}^{n} x_i - 1$$

$$L(x,\lambda) = \|x\|^2 - \lambda(\sum_{i=1}^{n} x_i - 1)$$

$$\nabla_x L = 2x - \lambda \qquad \nabla_\lambda L = \sum_{i=1}^{n} x_i - 1$$

Put the left equation in terms of x_i and plug into the right equation

$$x = \frac{\lambda}{2}\mathbf{1} \rightarrow \sum_{i=1}^{n} \frac{\lambda}{2}\mathbf{1} - 1 = 0 \rightarrow \lambda = \frac{2}{n}$$

Lastly plug λ back in to find the minimizing x vector

$$x = \frac{1}{n}\mathbf{1} \rightarrow x_i = \frac{1}{n}$$

(b)

$$f(x) = \sum_{i=1}^{n} x_i \qquad h(x) = \|x\|^2 - 1$$

$$L(x,\lambda) = \sum_{i=1}^{n} x_i - \lambda\{\|x\|^2 - 1\}$$

$$\nabla_x L = 0 = \mathbf{1} - 2\lambda x \qquad \nabla_\lambda = 0 = \|x\|^2 - 1$$

Now we solve the simultaneous equations like before

$$x = \frac{1}{2\lambda}\mathbf{1} \rightarrow x_i = \frac{1}{2\lambda}$$

$$\nabla_\lambda L = 0 = \left\| \frac{1}{2\lambda}\mathbf{1} \right\|^2 - 1 = n * \{\frac{1}{4\lambda^2}\} - 1 \rightarrow \lambda = \frac{\sqrt{n}}{2}$$

Lastly use λ to solve for x

$$x = \frac{1}{\sqrt{n}}\mathbf{1} \rightarrow x_i = \frac{1}{\sqrt{n}}$$

(c)

$$f(x) = \|x\|^2 \qquad h(x) = x^{'}Qx - 1$$

$$L(x,\lambda) = \|x\|^2 + \lambda\{x^{'}Qx - 1\}$$

Same procedure as before

$$\nabla_x L = 0 = 2x + 2\lambda Qx \qquad \nabla_\lambda = 0 = x^{'}Qx - 1$$

multiply both sides of the left term by $x^{'}$ to obtain

$$x^{'}x + \lambda x^{'}Qx = x^{'}x + \lambda = 0 \rightarrow \lambda = -x^{'}x$$

Honestly I got stuck here on how to use the other equation to resolve the λ.

Option 1

We start with the equation below to minimize with $(z)_+^2 = (max(0,z))^2$

$$f(x) = -x_1x_2 - x_2x_3 - x_3x_1 + (x_1-1)_+^2 + (-x_1-1)_+^2 + (x_2-1)_+^2 + (-x_2-1)_+^2 + (x_3-1)_+^2 + (-x_3-1)_+^2$$

$$\frac{\partial}{\partial z} z_+^2 = \begin{cases} 2z, & \text{if } z > 0 \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

(1) can be written as

$$\frac{\partial}{\partial z} z_+^2 = 2z(sign(z))_+$$

Now we can compute the partial derivatives

$$\frac{\partial f}{\partial x_1} = -x_2 - x_3 + 2(x_1-1)(sign(x_1-1))_+ + 2(x_1+1)(sign(-x_1-1))_+$$

$$\frac{\partial f}{\partial x_2} = -x_1 - x_3 + 2(x_2-1)(sign(x_2-1))_+ + 2(x_2+1)(sign(-x_2-1))_+$$

$$\frac{\partial f}{\partial x_3} = -x_2 - x_1 + 2(x_3-1)(sign(x_3-1))_+ + 2(x_3+2)(sign(-x_3-1))_+$$

Now fixing (x_2, x_3) we can optimize over x_1 by setting the partial with respect to x_1 equal to 0.

$$\frac{1}{2}(x_2+x_3) = (x_1-1)(sign(x_1-1))_+ + (x_1+1)(sign(-x_1-1))_+$$

Looking at this equation we can see that if $(x_2+x_3) = 0$ the only time this occurs on the right hand side is if $x_1 = [1,-1]$. Now if $(x_2+x_3) \neq 0$ the sign is determined by the sign of x_1 where if x_1 is negative we get the right term and the opposite if x_1 is positive. In either case if we drop the sign we get the equation $x_1 = 1 + \frac{1}{2}(x_2+x_3)$ so this all can be put into a more formal equation considering the sign as show below.

$$x_1 = \begin{cases} (1 + \frac{1}{2}|x_2+x_3|)sign(x_2+x_3), & \text{if } x_2+x_3 \neq 0 \\ [-1,1], & \text{otherwise} \end{cases} \tag{2}$$

For the six points given to be stationary points the gradient needs to be equal to zero. We can test by plugging the points into the expressions for the partial derivatives and see if they equate to 0. The quicker way would be to use the previous equation derived for the minimizer over one variable. We see that if $x_2 + x_3 = 0$ we wont get zero. Also if we minimized over the other two variables they would have a similar equation but with respect to the other two variables. Thus we can conclude based on these six points that cycle the sign around that at least one of the partial derivatives will equate to $[1,-1]$ so they are not stationary points.

Based on proposition 3.7.1 we can explain why the coordinate descent method diverges for this problem. That proposition assumes that the function f is continuously differentiable over the set X but this problem implements the ReLU function which is not differentiable at 0. This non-linearity of the ReLU function can produce cycles in the coordinate descent points from its non-uniqueness when under-flowing the ReLU function.

Programming Assignment

Linear Regression

This first model that was built and analyzed is a linear least squares regression model. The data that we are building this model around is a 2 feature vector that corresponds to a given handwritten digit. The features and symmetry and intensity. The specific task that we are trying to solve is to classify the handwritten digits 1 and 9. So the task is to produce a decision boundary that best separates digit 1 from digit 9. For this model the data needs to be processed into a form where the target labels take the form $t_i \in [-1, 1]$, which is done because our prediction function will take an input matrix and make a prediction on $sign(Ax)$. Along with that modification we fit the model with an intercept value so the training instances A will be extended to include a column vector that will consist of just ones, which correspond to the bias term in the weight vector. With these processing tasks out of the way we can move on to the model formulation. The model consists of an objective value that we are trying to minimize, which in this model is the least squares function $\frac{1}{2}\|Ax - b\|^2$. b is the target variables, A is the training instances and x is the weight vector that we are trying to find that minimizes the objective value. To iteratively obtain better weight vector estimations we need an algorithm, which i chose the gradient descent method. With this method the next weight vector in the sequence of iterates is chosen by $x^{k+1} = x^k - \alpha \nabla f(x)$ where α is the learning rate. So we will need the gradient of the objective function to compute the next iterate, which is given by $\nabla f(x) = (A^T A)x - A^T b$. To decide on an appropriate learning rate I tested a number of them and found that $\alpha = 0.0001$ produced accurate and fast results. Now lets get into the results obtained after running this algorithm of the training and testing data given to us. Figure 1 shows the training data with the proper classifications and the decision boundary that was obtained through the fitting of my linear model. Figure 3 shows the decision boundary for the test data. Since the final weight vector will correspond to $w^T x = 0$, we can rearrange the equation to put in a $y = mx + b$ form to plot the decision boundary (i.e $x2 = -\frac{w1}{w2}x1 - \frac{w0}{w2}$). We can see from the figures below that the model produces a decision boundary that appears to conform well to the data and separates the data nicely. We can get a more mathematical view how how accurate this model is by testing the accuracy on the test data after fitting on the train data. After prediction is run with the trained weight vector I obtain an accuracy of around 95.46%. This a fairly good accuracy for a linear model. Along with the decision boundary plots, figure 3 shows the objective values as the iterates progress through the algorithm during training. This plot shows that the gradient descent quickly finds the optimal value.

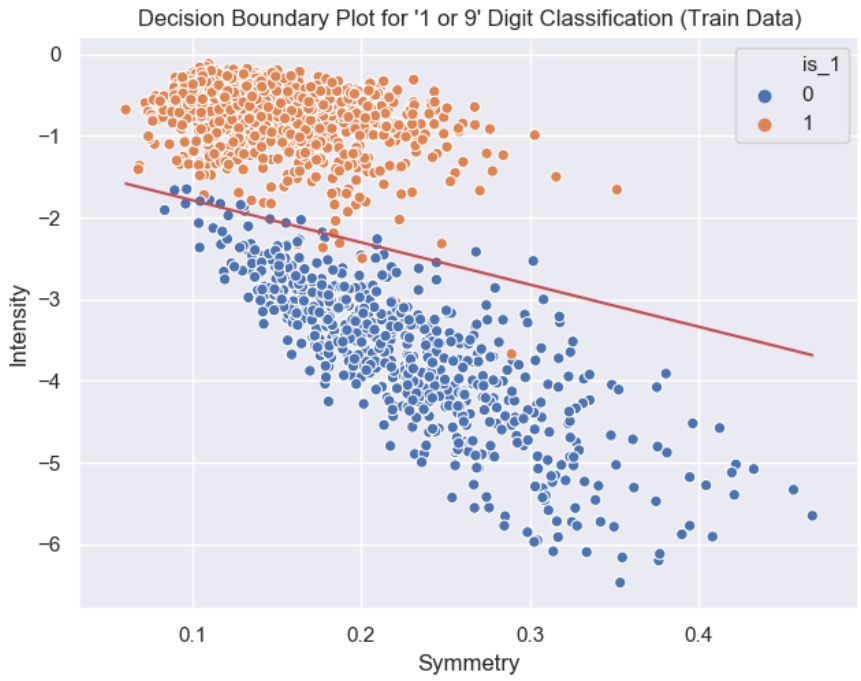

Figure 1: Feature data for digit 1 and 9 with decision boundary from fitting linear model with train data

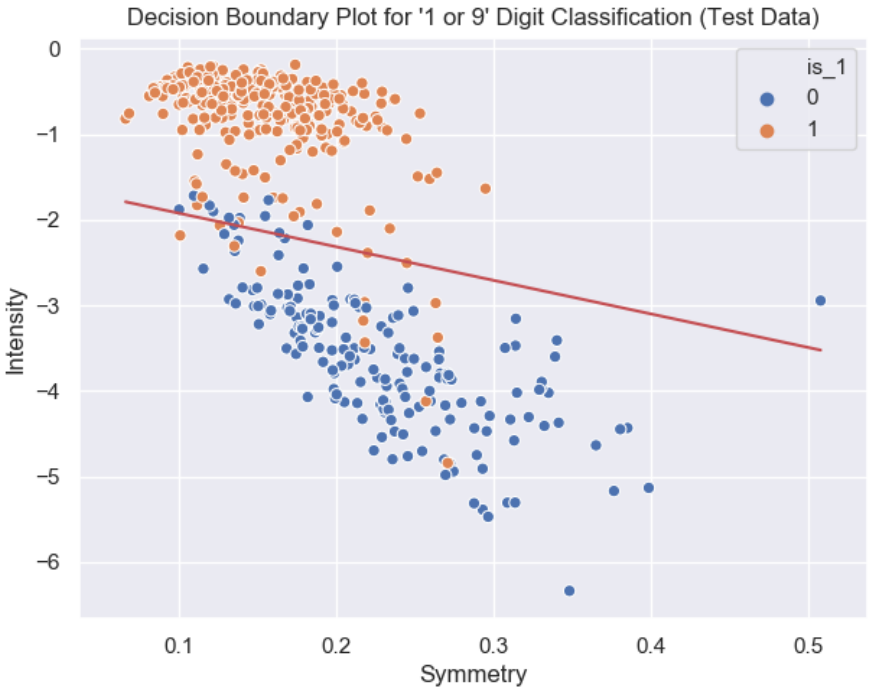

Figure 2: Feature data for digit 1 and 9 with decision boundary from fitting linear model with test data

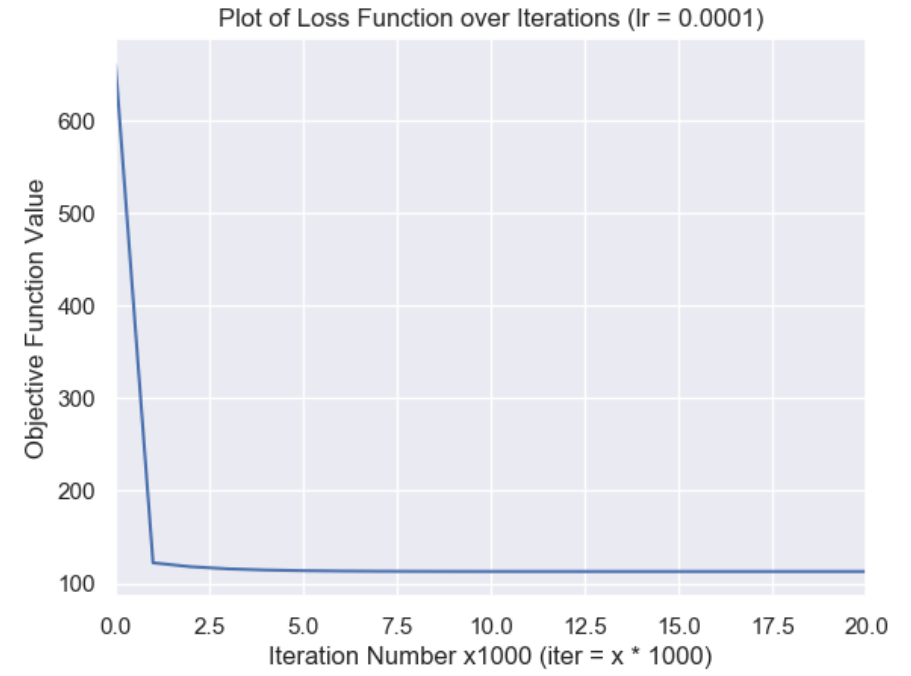

Figure 3: Objective function values as a function of the iterates during training

Logistic Regression

The next model that is built and analyzed is the logistic regression model. First we need to process the data to accommodate the model. For this model we classify a label as $l \in [0, 1]$, where we keep the same task as before, classifying between the digit 1 and the digit 9. I also add an intercept term in the data matrix to fit an intercept as well while training. The logistic model gives us probabilities of a data instance belonging to a specific class, which in this case is digit 1 or digit 9. To do this is uses a linear weight function to compute a value that is fed into the sigmoid function, which has the property of outputting a value between 0 and 1 like a probability. Using this function we can construct a loss function that is derived from maximizing the likelihood of classifying a specific class correctly giving an input instance. This loss function is conveniently name the maximum-log-likelihood ans is given by $L = \frac{1}{n} * (-y^T log(h) - (1-y)^T log(1-h)$ where h is the output from the sigmoid function. Now that we have our loss function we use an algorithm to iteratively descend down this loss function to find the minimum. I used the same gradient descent algorithm as the linear regression but now the gradient used for updating is given by the gradient of the maximum-log-likelihood loss function $\boldsymbol{\nabla}_L = X^T(h-y)/n$. The learning rate will be different for this model so I tested a few and found that $\alpha = 0.1$ provided good results. Figure 4 and figure 5 below show the data points with the calculated decision boundary for the training data and for the test data. The logistic regression decision boundary is slightly different from the linear regression decision boundary but fairly accurate as it seems. Running the predictions for the test data on the logistic regression model I obtain an accuracy of around 94.78%. This is once again a fairly good accuracy and very close to the accuracy of the linear model. Figure 6 shows the loss function values as a function of the iterates and you can see that it reaches a consistent value fairly quickly. Slower than that of the linear model but quickly nonetheless.

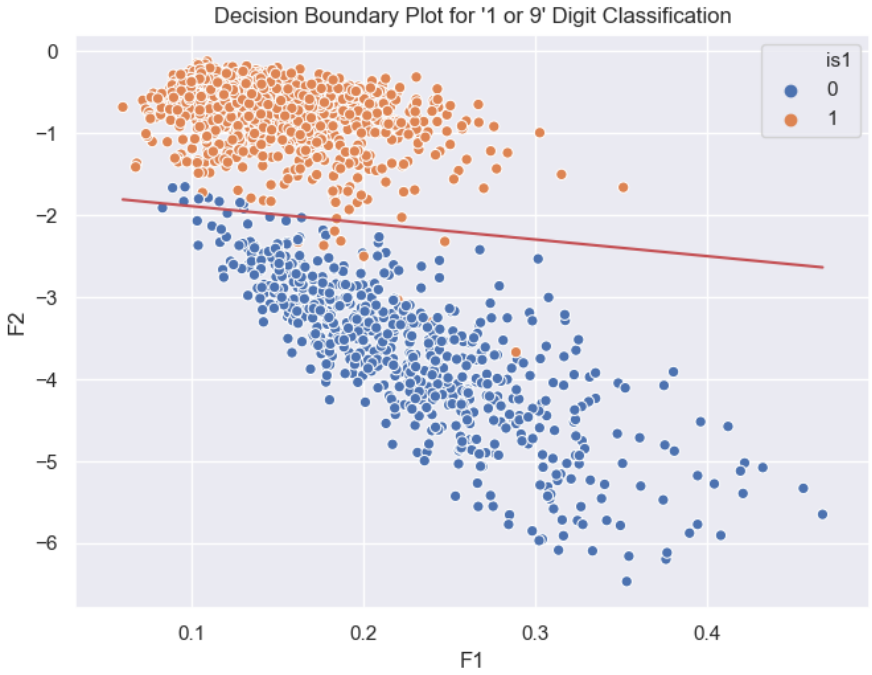

Figure 4: Feature data for digit 1 and 9 with decision boundary from fitting logistic model with train data

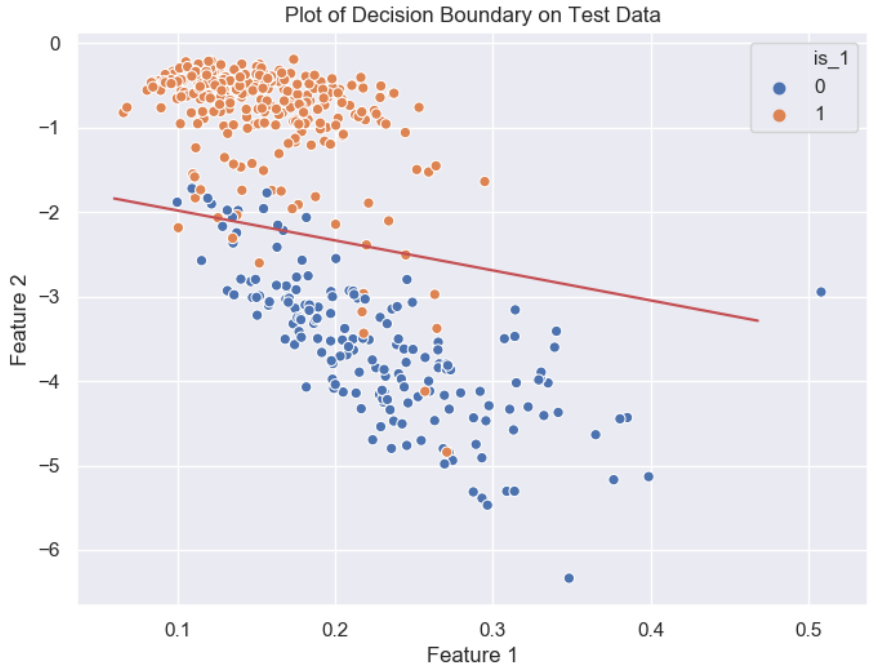

Figure 5: Feature data for digit 1 and 9 with decision boundary from fitting logistic model with test data

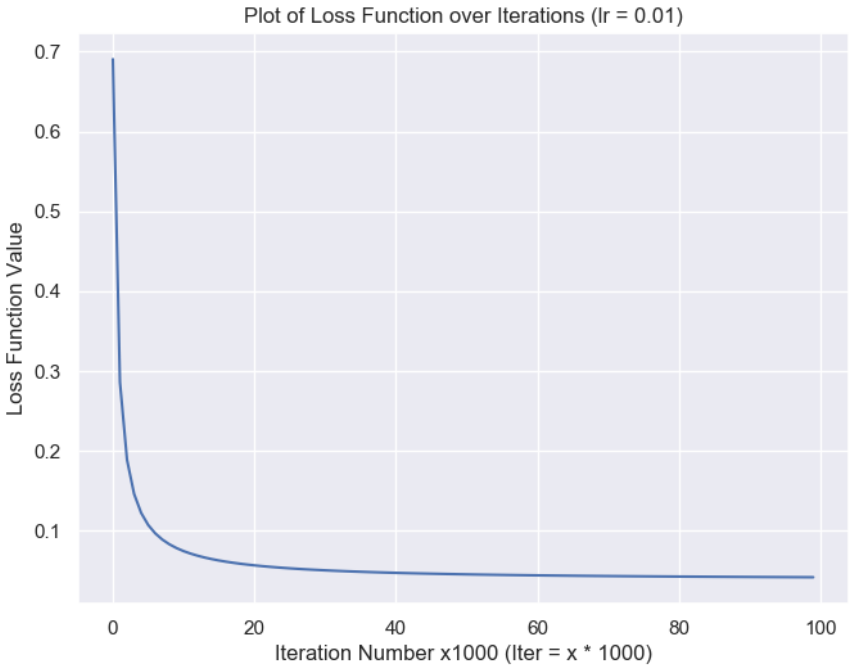

Figure 6: Feature data for digit 1 and 9 with decision boundary from fitting logistic model with test data