

In the name of God



University of Tehran
College of Engineering
Faculty of ECE



Intelligent Systems

HW1

Reza Jahani

810198377

Fall 1401

Table of Contents

Binary Logistic Regression	3
Gradient Descent (Analytical)	3
Gradient Descent Computation	4
Accuracy Computation	4
Stochastic Gradient Descent	5
Non-Convex Optimization	7
Newton Method	7
Implementation	8
Genetic Algorithm	9
Support Vector Machine	10
Analytical	10
Implementation	10

Question 1 – Binary Logistic Regression

Part 1) Gradient Computation (Analytic)

$$J(w, b) = \frac{1}{n} \sum_{i=1}^n \log \left(1 + \exp \left(-y_i (b + x_i^T w) \right) \right)$$
$$\mu_i(w, b) = \frac{1}{1 + \exp \left(-y_i (b + x_i^T w) \right)}$$

Based on the notations we will have:

$$J(w, b) = \frac{1}{n} \sum_{i=1}^n \log \left(\frac{1}{\mu_i(w, b)} \right)$$
$$\nabla_w J = \frac{\partial J}{\partial w} = \frac{1}{n} \sum_{i=1}^n \mu_i(w, b) \cdot -\frac{1}{(\mu_i(w, b))^2} \cdot \frac{\exp(-y_i(b + x_i^T w))}{\left(\frac{1}{\mu_i(w, b)}\right)^2} \cdot (-y_i x_i^T) =$$
$$= \frac{1}{n} \sum_{i=1}^n \mu_i(w, b) \cdot \exp \left(-y_i (b + x_i^T w) \right) \cdot (-y_i x_i^T)$$
$$\nabla_w J(w, b) = -\frac{1}{n} \sum_{i=1}^n \mu_i(w, b) \cdot \left(\frac{1}{\mu_i(w, b)} - 1 \right) \cdot (y_i x_i^T)$$

Going through the same procedure we will have:

$$\nabla_b J(w, b) = -\frac{1}{n} \sum_{i=1}^n \mu_i(w, b) \cdot \left(\frac{1}{\mu_i(w, b)} - 1 \right) \cdot (y_i)$$

Part 2) Descending Gradient Computation

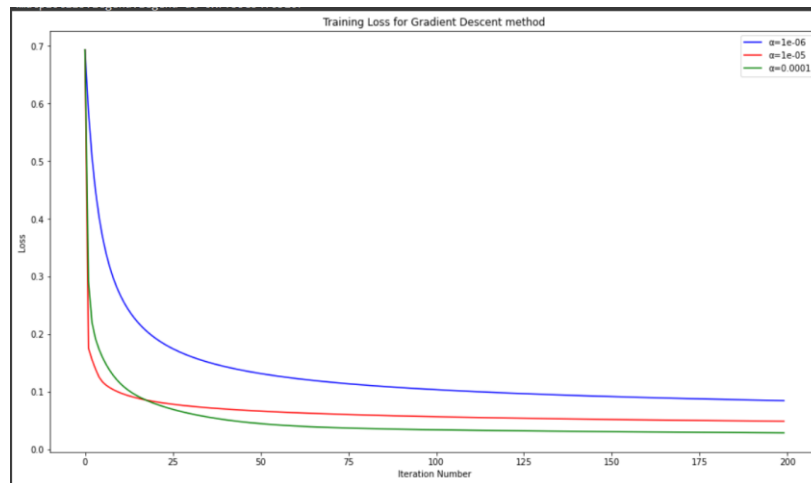


Figure 1.2.1

Above the training loss can be observed. The process which is used to reach this result is implementation of gradient descent. In the colab file all steps are obvious and clear employing different functions for each section of the implementation.

Part 3) Accuracy Computation

Below accuracy of the model on training data and test data is presented. These parameters are calculated using predicting the labels with the trained model and then comparing the predicted labels with the true labels.

Accuracy of Training Data:	Accuracy of Test Data:
Model 1: 97.53%	Model 1: 97.31%
Model 2: 98.5%	Model 2: 97.89%
Model 3: 99.1%	Model 3: 98.6%

Figure 1.3.1

Part 4) Stochastic Gradient Descent

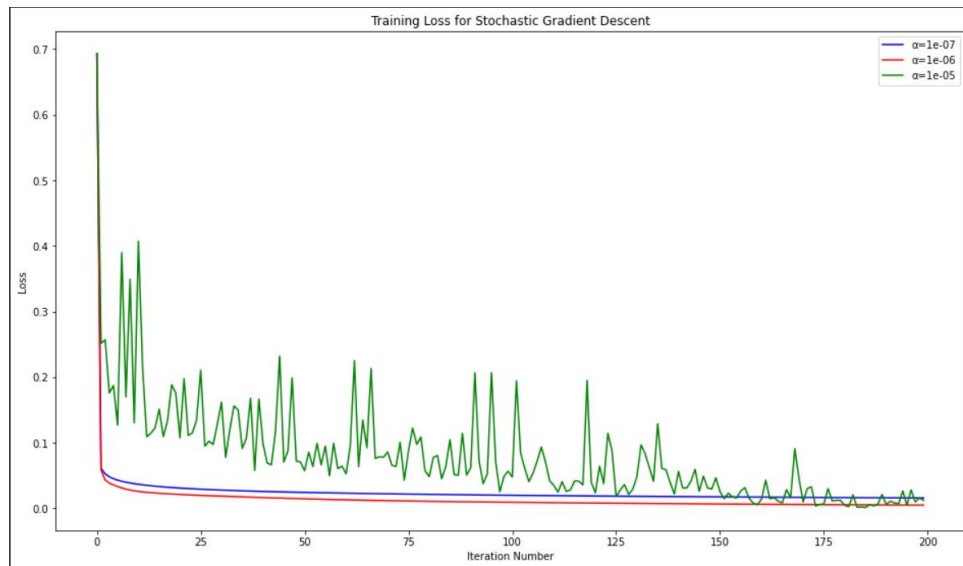


Figure 1.4.1

Using the implementation algorithm of Stochastic Gradient Descent, the model is trained with 3 different learning rates and observed above, the training loss is plotted per epochs. Using the same procedure, the model is used to predict the train and test data and after comparing predicted labels with the true labels, the accuracy is computed.



Figure 1.4.2

After this implementation we use the mini batch gradient descent algorithm. After the implementation and using the model to predict the data, the accuracy is computed and the result can be observed in figures 1.4.3 and 1.4.4.

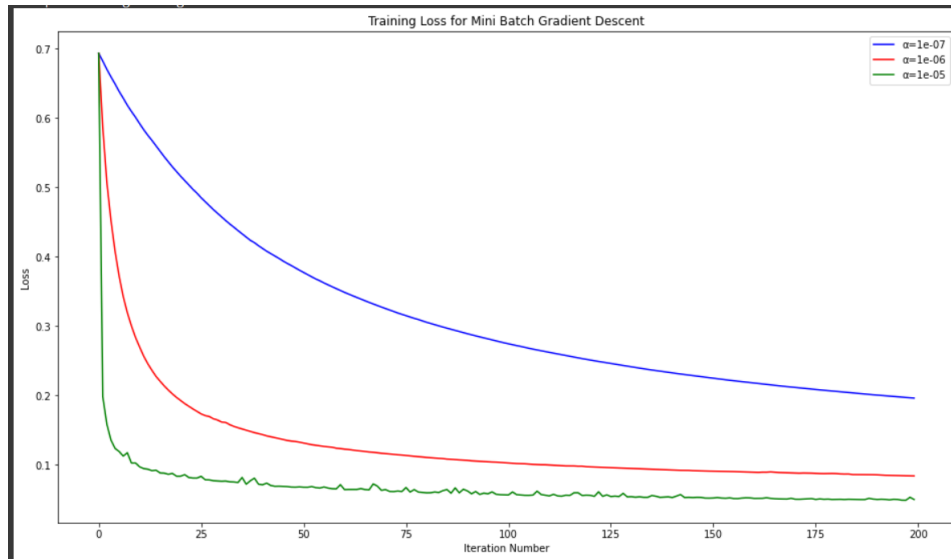


Figure 1.4.3



Figure 1.4.4

Question 2 – Optimization of Non-Convex Functions

Part 1) Newton Method (Analytic)

$$f(x) = 2x_1^2 + 2x_2^2 - 17 \cos(0.2\pi x_1) x_2 - x_1 x_2$$

$$\frac{\partial f}{\partial x_1} = 4x_1 + 17 \times 0.2\pi x_2 \sin(0.2\pi x_1) - x_2 \quad \frac{\partial f}{\partial x_2} = 4x_2 - 17 \cos(0.2\pi x_1) - x_1$$

$$\frac{\partial^2 f}{\partial x_1^2} = 4 + 3.4\pi x_2 \cos(0.2\pi x_1) \quad \frac{\partial^2 f}{\partial x_2^2} = 4$$

$$\frac{\partial^2 f}{\partial x_1 \partial x_2} = 3.4\pi \sin(0.2\pi x_1) - 1 \quad \frac{\partial^2 f}{\partial x_2 \partial x_1} = 17 \sin(0.2\pi x_1) - 1$$

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix} \quad \nabla^2 f = H = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix}$$

Hessian Matrix

$$\nabla f = \begin{bmatrix} 4x_1 + 3.4\pi x_2 \sin(0.2\pi x_1) - x_2 \\ 4x_2 - 17 \cos(0.2\pi x_1) - x_1 \end{bmatrix} \quad H = \begin{bmatrix} 4 + 3.4\pi x_2 \cos(0.2\pi x_1) & 3.4\pi \sin(0.2\pi x_1) - 1 \\ 3.4\pi \sin(0.2\pi x_1) - 1 & 4 \end{bmatrix}$$

Iteration

0 $x = (0, 0) \rightarrow \nabla f = \begin{bmatrix} 0 \\ -17 \end{bmatrix} \quad H = \begin{bmatrix} 4 & -1 \\ 0 & 4 \end{bmatrix}$

$$x^{k+1} = x^k - (\nabla^2 f)^{-1} \nabla f$$

$$(\nabla^2 f)^{-1} = \begin{bmatrix} 1/4 & 1/15 \\ 1/15 & 1/4 \end{bmatrix} = \begin{bmatrix} 1/15 & 1/15 \\ 1/15 & 1/15 \end{bmatrix}$$

1 $x^{(1)} = x^{(0)} - (H)^{-1} (\nabla f)^{(0)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 1/15 & 1/15 \\ 1/15 & 1/15 \end{bmatrix} \begin{bmatrix} 0 \\ -17 \end{bmatrix} = \begin{bmatrix} 1.1333 \\ 4.533 \end{bmatrix}$

$$x^{(1)} = \begin{bmatrix} 1.1333 \\ 4.533 \end{bmatrix}$$

Point after one update

Part 2) Newton Method (Simulation)

In this section we aim to employ the newton method which follows the below algorithm to update the initial point and reach the minimum value of the 2 variable function.

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{pmatrix} \quad H = \nabla^2 f = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_2 \partial x_1} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_2^2} \end{pmatrix}$$

$$x^{(k+1)} = x^{(k)} - (H^{-1} \nabla f)_{@x^{(k)}}$$

After 20 iterations, we reach below results.

```
Final point after optimization: x=(0.13,4.27)
Minimum value of the function: f=-36.4
```

Figure 2.2.1

Using the algorithm presented as instructed comparison between initial values and the final optimum point is observed below in figure 2.2.2

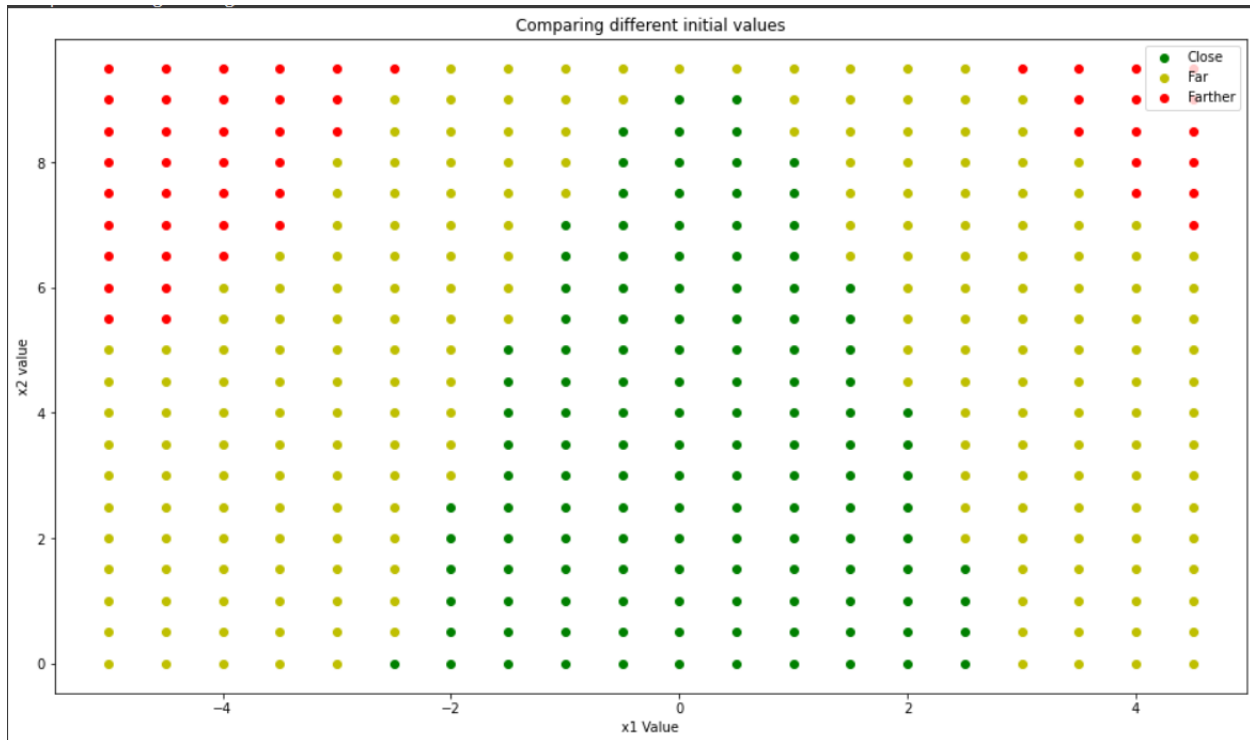
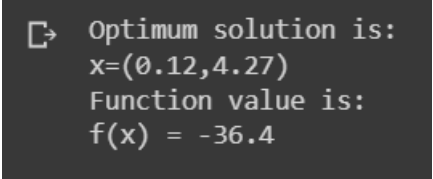


Figure 2.2.2

Part 3)

In this section, genetic algorithm was employed to reach the optimum solution for the function. As illustrated in the code in the google colab file, below results were obtained in figure 2.3.1. As it can be seen, the result is so close to the real answer.



```
↳ Optimum solution is:  
x=(0.12,4.27)  
Function value is:  
f(x) = -36.4
```

Figure 2.3.1

Question 3 – Support Vector Machine

Part 1) Analytic

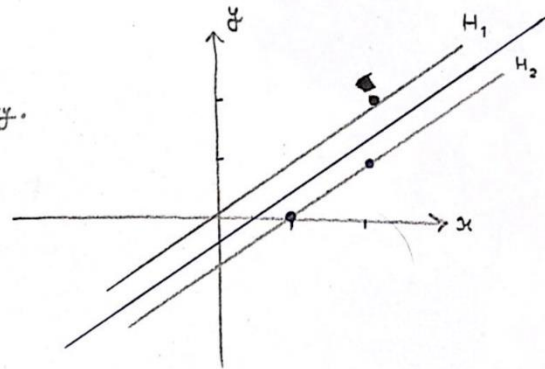
1) $E_i = \max(0, 1 - y_i(\omega^T x_i + b))$
 if $E_i = 0 \rightarrow 1 - y_i(\omega^T x_i + b) \leq 0$
 $y_i(\omega^T x_i + b) \geq 1 \Rightarrow$ Sample x_i , Label = +1
 (Maybe the sample is on H_1 plane)

2) $\bullet : -$
 $\bullet : +$

The blue line indicates the optimum decision boundary.

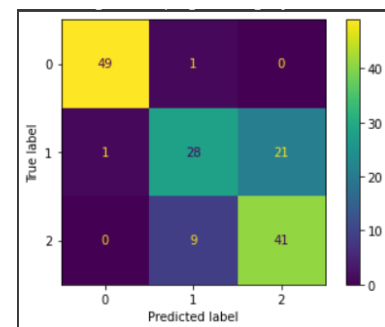
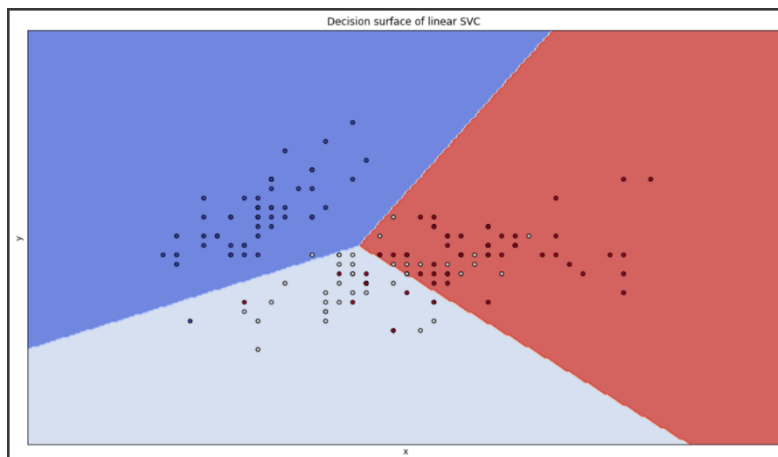
$$y = x - 1/2$$

The third choice ✓ (3)



Part 2) Implementation

Applying the support vector machine algorithm on the dataset using the sklearn library, we can reach results as below. More explanations and details are included in the colab file.



Confusion Matrix

The Confidence Matrix:
 x-axis: Predicted Labels / y-axis: True Labels

0	0.33	0.01	0.0
1	0.01	0.19	0.14
2	0.0	0.06	0.27
.	0	1	2

Model's Accuracy: 78.67%

- Additional Section
- The file employed for implementation is HW1_810198377.ipynb