University of Tehran
College of Engineering
School of Electrical and Computer Engineering

Intelligent Systems

Dr.Hosseini

Homework 1

Soroush Mesforush Mashhad

SN:810198472

Aban 01

Contents

Abstract

In this assignment we shall become more familiar with **Logistic Binary Regression**, in this section we calculate the gradient of the given function analytically then we go on to implement **Gradient Descent**, **Stochastic Gradient Descent** and the accuracies in each scenario.

In the second part we shall attempt to optimize a function using the **Newton method** both analytically and by implementing it using python, in the end we use the **Genetic algorithm** to solve the optimization problem.

In the final section we study the **Support Vector Machine** and answer two analytic questions then go on to implement it via python.

1 Question 1 : Logistic Binary Regression

1.1 Part A : Gradient calculation

To calculate the gradient with respect to b and w we use partial derivatives.

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

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

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

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

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

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

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

So all in all we shall have the followng:

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

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

1.2 Part B : Gradient Descent

Here we have implemented Gradient Descent as we studied in the course materials, the output figures are as follows.

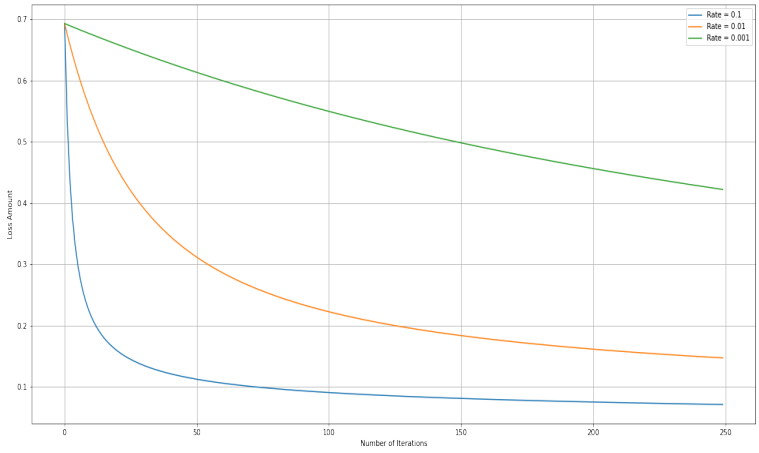


Figure 1: Gradient Descent on train data

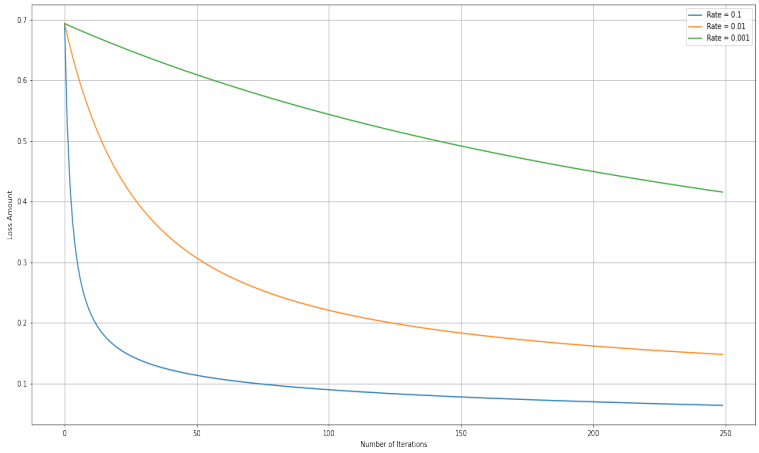


Figure 2: Gradient Descent on test data

1.3 Part C : Accuracy

Here we go on to calculate the accuracies, the outputs are as follows:

```
Accuracy of 1st, 2nd and 3rd state is :
97.92979297929793 96.7982512536968 96.2067635334962
```

Figure 3: Accuracy for first scenario

```
Accuracy of 1st, 2nd and 3rd state is :
98.25 96.5 95.75
```

Figure 4: Accuracy for second scenario

1.4 Part D : Stochastic Gradient Descent

Here we implement the SGD method for different batch sizes.

1.4.1 $Batchsize = 1$

For this scenario we have:

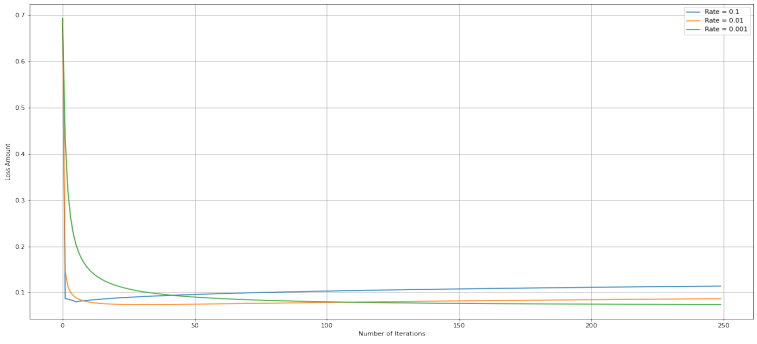


Figure 5: Stochastic Gradient Descent on train data

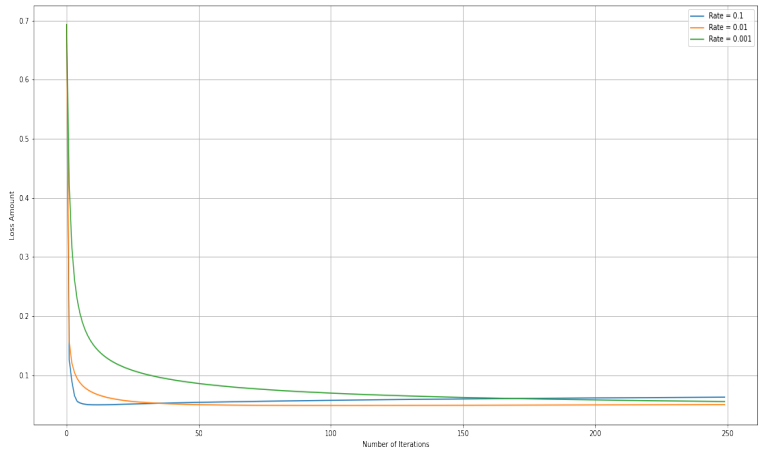


Figure 6: Stochastic Gradient Descent on test data

For the accuracies we have:

```
Accuracy of 1st, 2nd and 3rd state is :
97.27401311559727 97.27401311559727 97.35116368779735
```

Figure 7: Accuracy for first scenario

```
Accuracy of 1st, 2nd and 3rd state is :
98.125 98.375 98.5
```

Figure 8: Accuracy for second scenario

1.4.2 $Batchsize = 100$

For this scenario we have:

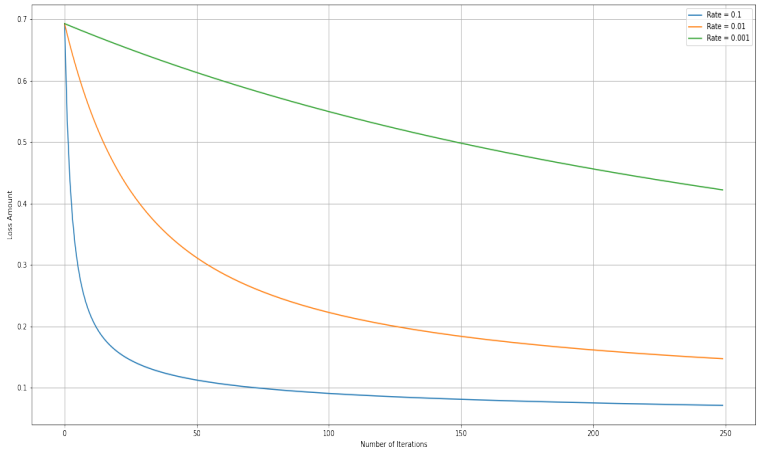


Figure 9: Stochastic Gradient Descent on train data

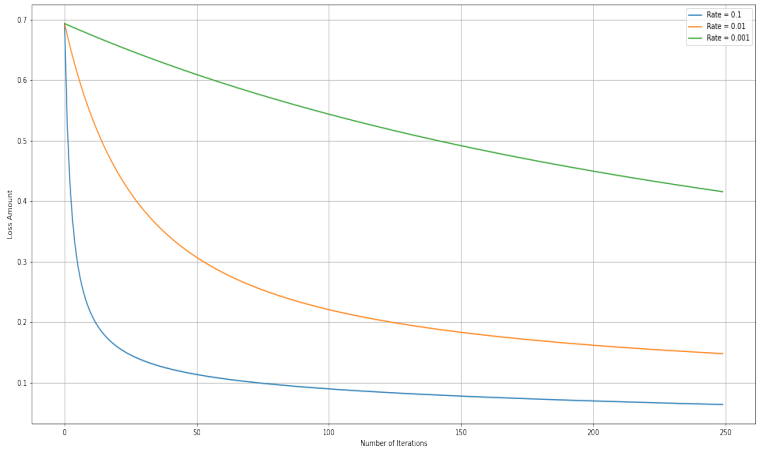


Figure 10: Stochastic Gradient Descent on test data

For the accuracies we have:

```
Accuracy of 1st, 2nd and 3rd state is :
97.92979297929793 96.7982512536968 96.2067635334962
```

Figure 11: Accuracy for first scenario

```
Accuracy of 1st, 2nd and 3rd state is :
98.25 96.5 95.75
```

Figure 12: Accuracy for second scenario

Here, the first part of this assignment is concluded.

2 Question 2 : Optimization in non - convex functions

2.1 Part A : Newton method (Analytic)

In this part we shall implement the Newton method for one iteration, as we studied in the lectures we have:

$$\underline{x}^{(k+1)} = \underline{x}^{(k)} - \mathcal{H}^{-1}\nabla f, \quad \underline{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \mathcal{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix}, \quad \nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix}$$

So in this case we shall have:

$$f(x_1, x_2) = 2x_1^2 + 2x_2^2 - 17x_2\cos(0.2\pi x_1) - x_1x_2$$

$$\frac{\partial^2 f}{\partial x_1^2} = 4 + 0.68\pi^2 x_2\cos(0.2\pi x_1), \quad \frac{\partial^2 f}{\partial x_1 \partial x_2} = -1$$

$$\frac{\partial^2 f}{\partial x_2 \partial x_1} = \frac{17\pi}{5}\sin(0.2\pi x_1) - 1, \quad \frac{\partial^2 f}{\partial x_2^2} = 4$$

$$\mathcal{H} = \begin{bmatrix} 4 + 0.68\pi^2 x_2\cos(0.2\pi x_1) & -1 \\ \frac{17\pi}{5}\sin(0.2\pi x_1) - 1 & 4 \end{bmatrix}, \quad \text{Starting Point : } (x_1, x_2) = (0, 0)$$

$$\mathcal{H} = \begin{bmatrix} 4 & -1 \\ -1 & 4 \end{bmatrix} \xrightarrow{\mathcal{H}^{-1}} = \frac{1}{det(\mathcal{H})}\mathcal{H}^* = \frac{1}{15}\begin{bmatrix} 4 & 1 \\ 1 & 4 \end{bmatrix} = \begin{bmatrix} \frac{4}{15} & \frac{1}{15} \\ \frac{1}{15} & \frac{4}{15} \end{bmatrix}$$

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 4x_1 + 3.4\pi x_2\sin(0.2\pi x_1) - x_2 \\ 4x_2 - 17\cos(0.2\pi x_1) - x_1 \end{bmatrix} \xrightarrow{(x_1^{(0)}, x_2^{(0)}) = (0,0)} = \begin{bmatrix} 0 \\ -17 \end{bmatrix}$$

Now we shall calculate (x_1, x_2) for one iteration, we have:

$$\underline{x}^{(k+1)} = \underline{x}^{(k)} - \mathcal{H}^{-1}\nabla f \xrightarrow[k=0]{(x_1^{(0)}, x_2^{(0)}) = (0,0)} \underline{x}^{(1)} = \underline{x}^{(0)} - \mathcal{H}^{-1}\nabla f$$

$$\begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} \frac{4}{15} & \frac{1}{15} \\ \frac{1}{15} & \frac{4}{15} \end{bmatrix}\begin{bmatrix} 0 \\ -17 \end{bmatrix} \longrightarrow \begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \end{bmatrix} = \begin{bmatrix} \frac{17}{15} \\ \frac{68}{15} \end{bmatrix}$$

2.2 Part B : Newton method (Implementation)

Here we shall implement the Newton method using python, this has been done accordingly and the results are as follows.

```
x1 is : 0.13087466007249957 and x2 is : 4.268357652256218(after 100 iterations)
The Minimum Value for the function is : -36.40349774185023 (after 100 iterations)
```

Figure 13: Answers to the Optimization problem

The scatter plot is as follows.

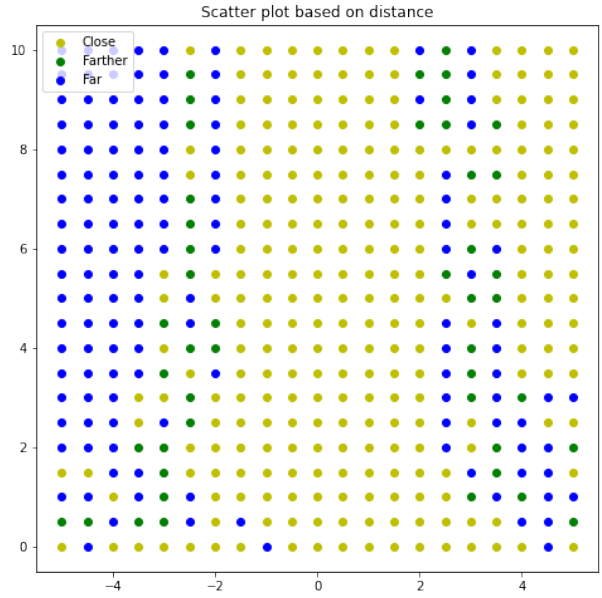


Figure 14: Scatter plot based on distance

Here we have essentially calculated the distance of the function value from the local minimum, it is understandable that this function probably has more than more local minimum which results in the variation of our scatter plot.

2.3 Part C : Genetic Algorithm (Implementation)

After studying various resources we decided to implement the genetic algorithm, to do so we we had to choose various parameters for our implementation.

We decided to use a uniform distribution to create the array of x_1 and x_2 so we can guaranteed that they will be in our desired range, we also needed to specify the length of our answer array and the number of iterations to generate a useful and precise number of answers, these chosen parameters are as follows:

$$\text{Anslength} = 750, \quad \text{Iterations} = 5 \times \text{Anslength}$$

These parameters were chosen by testing various amounts and checking the speed of the program, the final answers are as follows.

```
x1 =
0.11895953539174675
x2 =
4.2435585147612045
The value is :
-36.40000482314009
```

Figure 15: Answers to the Optimization problem via Genetic Algorithm

3 Question 3 : Support Vector Machine

3.1 Analytic

3.1.1 Part 1

We denote the below formula for the support vector machine classification.

$$\text{Minimize } \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i, \quad \text{Subject to } \xi_i = max(0, 1 - y_i(w^T x_i + b))$$

We know that $max(0, 1 - y_i(w^T x_i + b))$ is a form of the hinge loss function. We know that in the above notation y_i is the i^{th} target and $w^T x_i + b$ is the i^{th} output.

We know that for each data to lie on the correct point of the margin the following constraints must be met.

$$w^T x_i + b \geq 1 \text{ when } y_i = 1, \quad w^T x_i + b \leq -1 \text{ when } y_i = -1$$

The above constraints can be written as follows.

$$y_i(w^T x_i + b) \geq 1$$

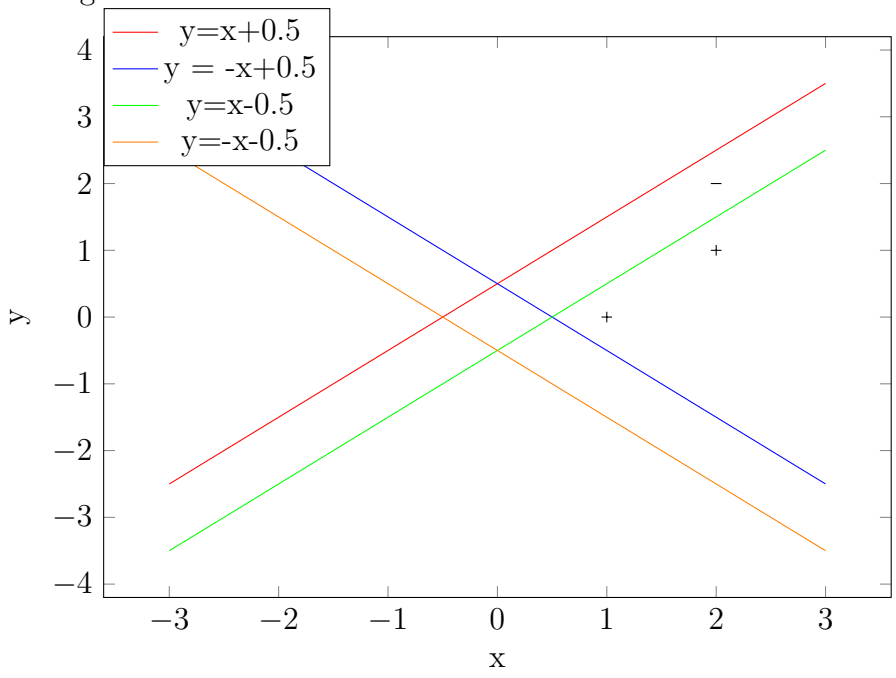
Now, if we assume that $\xi_i = 0$ for one sample, we have the following.

$$max(0, 1 - y_i(w^T x_i + b)) = 0 \longrightarrow 1 - y_i(w^T x_i + b) \leq 0 \longrightarrow y_i(w^T x_i + b) \geq 1$$

We easily proved that the constraints that we talked about hold if $\xi_i = 0$ for one sample, so this sample is on the support vector, or in other words this sample is on the correct side of the margin.

3.1.2 Part 2

We shall draw a graph depicting the points and different lines so we can have a more logical decision about the decision border.



By observing the following graph, it is obvious to us that if we connect the two positive points and make a line using them, and if we also make another line using the third support vector, it is quite obvious that the line between them is accepted as the decision border.

So all in all we have

$Border : y = x - 0.5$

3.2 Implementation

Here we have implementded the SVM using **Scikit-Learn**, in the first step we can check our accuracy.

```
Accuracy = 0.7866666666666666
```

Next we go on to plot the confusion and confidence matrices.

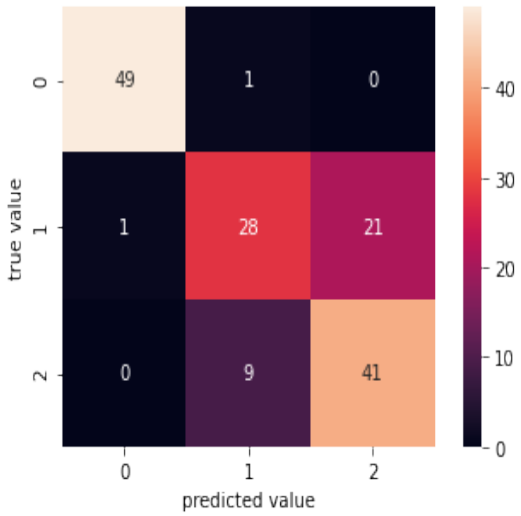


Figure 16: Confusion Matrix

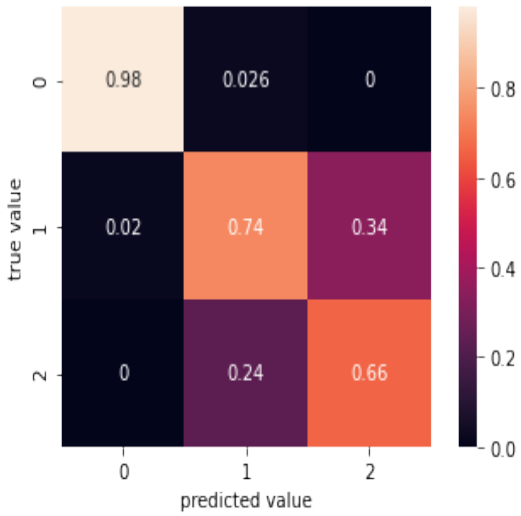


Figure 17: Confidence Matrix

It is obvious to us that the confidence matrix is the confusion matrix whilst its columns are divided by the sum of the elements on the said column, all in all we can conclude that it gives us some form of probability.

In the end we shall plot the classification regions.

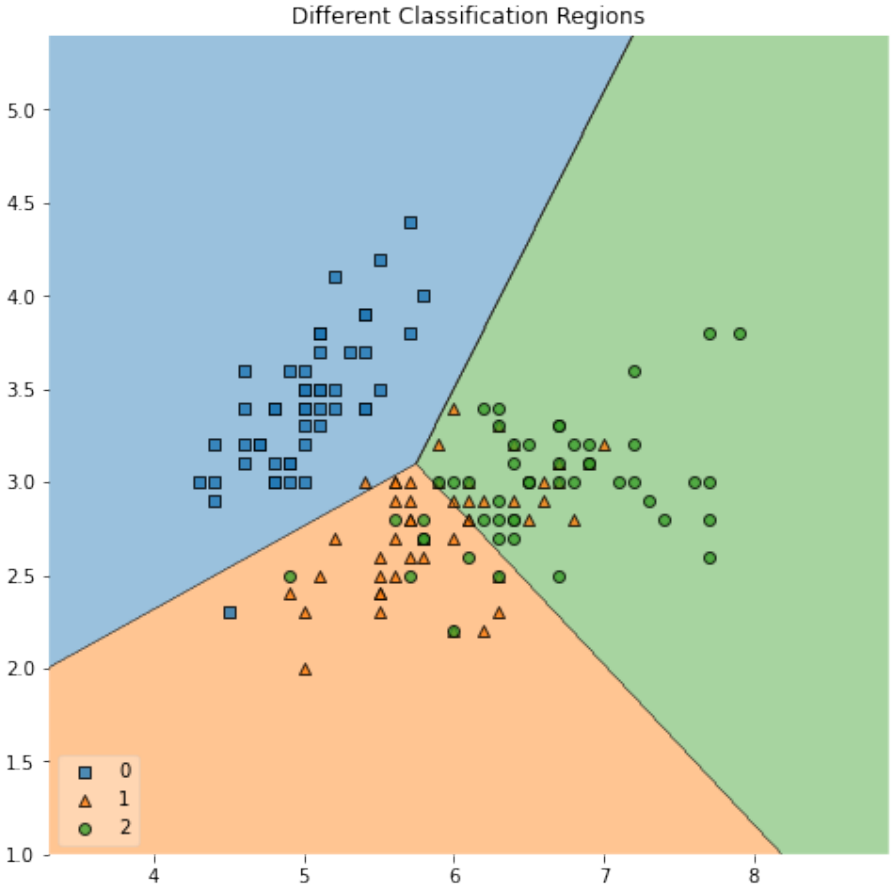


Figure 18: Classification regions

References

[1] Reshad Hosseini, *Intelligent Systems Lecture Notes, Fall 01*

[2] Jorge Nocedal, Stephen J.Wright, *Numerical Optimization (2nd edition), Springer, 2006*

[3] Gradient Descent

[4] Stochastic Gradient Descent

[5] Support Vector Machine

[6] Introduction to Genetic Algorithms

[7] Genetic Algorithm in Python

[8] Classification Regions