Zeehan Rahman

Solution:

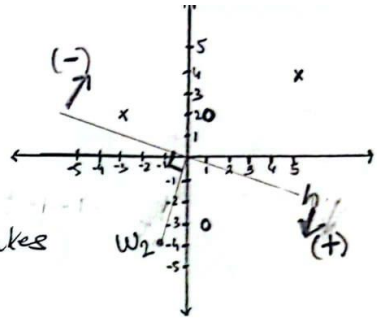
Method 1 and Method 2 both run the Pocket Algorithm. However, method 1 initializes the w vector with the first data point. In the case for method 2, linear regression algorithm is used to obtain a closed form solution for the weight vector.

The following steps are same for both the methods:

1. Get the dot product of the dataset and the weight vector w.(lets call it M1) (Predicted value)
2. Change the y label from 0 and 1 to -1 and 1
3. Get the dot product obtained of y and M1 (let's call it M2). When the actual value agrees with the predicted value (correct prediction) the corresponding element position of M2 is positive, as the sign agrees. When the actual value disagrees with the predicted value (incorrect prediction), the corresponding element position of M2 is negative. (see the example below)
4. The count of negative numbers are the total mistakes made by the current w vector. If the current mistake count is less than the lowest recorded mistake count, the lowest recorded is assigned the current mistake count.
5. Afterwards, a random mistake (if any) is picked from the list of mistakes, and the w vector is updated using the equation: w_t+1 = w_t + y_n_t * x_n_t, where w_t is the current vector, y_n_t is the actual label and x_n_t is the feature vector of the data point where the mistake occurred. This equation ensures the update weight vector moves in the correct direction, to (at least) fix the current mistake.
6. Repeat step 3 with till the given number of iterations. The final weight vector might not be the best weight vector as finding the best weight vector is NP hard.

   An example below shows the mathematical side of the above steps:

DATASET $\begin{bmatrix} 1 & 2 \\ -3 & 2 \\ 5 & 4 \\ 1 & -3 \end{bmatrix}$ $Y\begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix}$



$w_0 = [1\ 2] \Rightarrow \begin{bmatrix} 1 & -3 & 5 & 1 \\ 2 & 2 & 4 & -3 \end{bmatrix} \Rightarrow \begin{bmatrix} 5 & 1 & 13 & -5 \end{bmatrix} \Rightarrow \begin{bmatrix} 5 & 1 & 13 & -5 \\ 1 & -1 & -1 & 1 \end{bmatrix}$

$[+\ -\ -\ -] \Rightarrow$ 3 mistakes

$w_1 = w_0 + y_{n(1)} x_{n(1)} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} + (-1)[-3\ 2] = [4\ 0]$

$\Rightarrow [4\ 0] \begin{bmatrix} 1 & -3 & 5 & 1 \\ 2 & 2 & 4 & -3 \end{bmatrix} = \begin{bmatrix} 4 & -12 & 20 & 4 \\ 1 & -1 & -1 & 1 \end{bmatrix} \Rightarrow [+\ +\ -\ +] \Rightarrow$ 1 mistake

$w_2 = [4\ 0] + (-1)[5\ 4] = [-1\ -4]$

$\Rightarrow [-1\ -4] \begin{bmatrix} 1 & -3 & 5 & 1 \\ 2 & 2 & 4 & -3 \end{bmatrix} \Rightarrow \begin{bmatrix} -9 & -5 & -11 & 11 \\ 1 & -1 & -1 & 1 \end{bmatrix} \Rightarrow [-\ +\ +\ +] \Rightarrow$ 1 mistake

End of iterations:

As we can see in the graph above, line h that is perpendicular to w_2, predicts positive on one side (lower region ↓) and negative on the other side (upper region ↑).

As a result, w_2 makes only one mistake of classifying data point (1,2) as negative. This is also the best possible w as the data is not linearly seperable.

Training and Validation

Experiment:

1. The breast cancer dataset is loaded from skicit library. The data is split into training and testing (20%).
2. Afterwards, the weight vector is initialized with the first data point and then again using linear regression algorithm.
3. Consequently, the training dataset with the y vector and initialized weight vector is passed to the fit method. The fit method runs the Pocket Algorithm that returns the ideal weight vector with the corresponding number of mistakes. The mistake is divided by the size of the training data to get E_in.
4. In the testing phase, the E_out is calculated by dividing the number of mistakes made in the test data by the size of the test data.
5. E_in (X-axis) for both initialization method vs increasing number of iterations is plotted in a graph

Discussions:

The table below shows the E_out for the two-initialization method with 5 different sample data:

E_in (1st data point)	E_out (1st data point)	Difference	E_in (lin reg)	E_out (lin reg)	Difference
0.0703	0.1140	0.0437	0.039	0.088	0.049
0.061	0.11	0.049	0.046	0.035	0.011
0.073	0.061	0.012	0.035	0.035	0
0.061	0.096	0.035	0.037	0.035	0.02
0.068	0.078	0.010	0.039	0.035	0.04
0.057	0.149	0.092	0.032	0.070	0.038

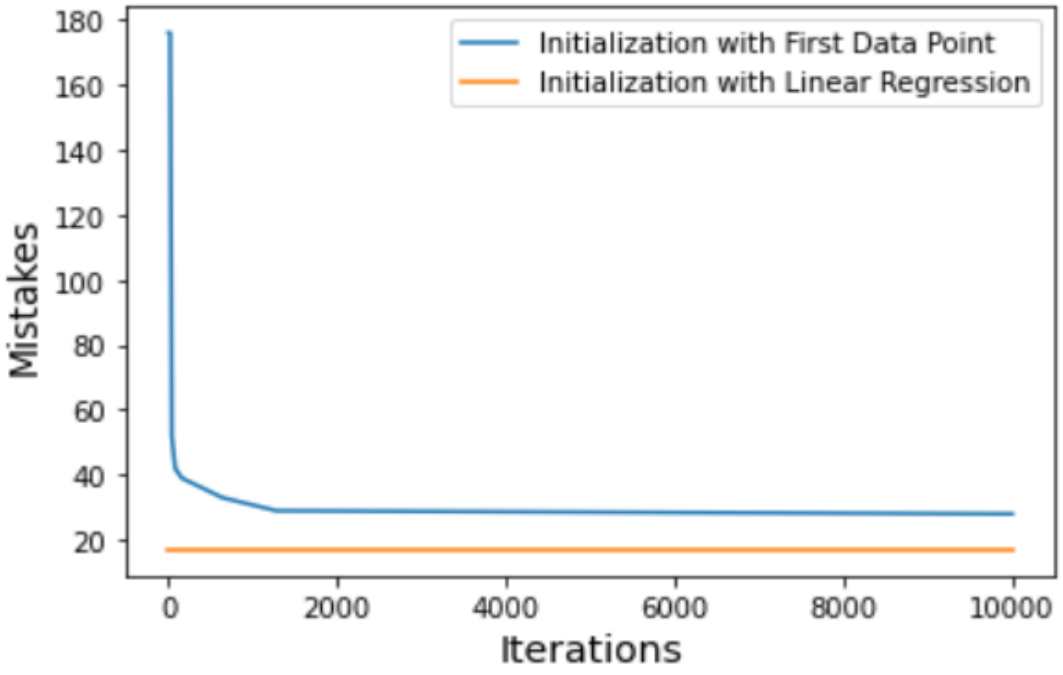
As the number of mistakes for the linear regression method is significantly lower, E_out for the linear regression is better than the E_out for the first data point.

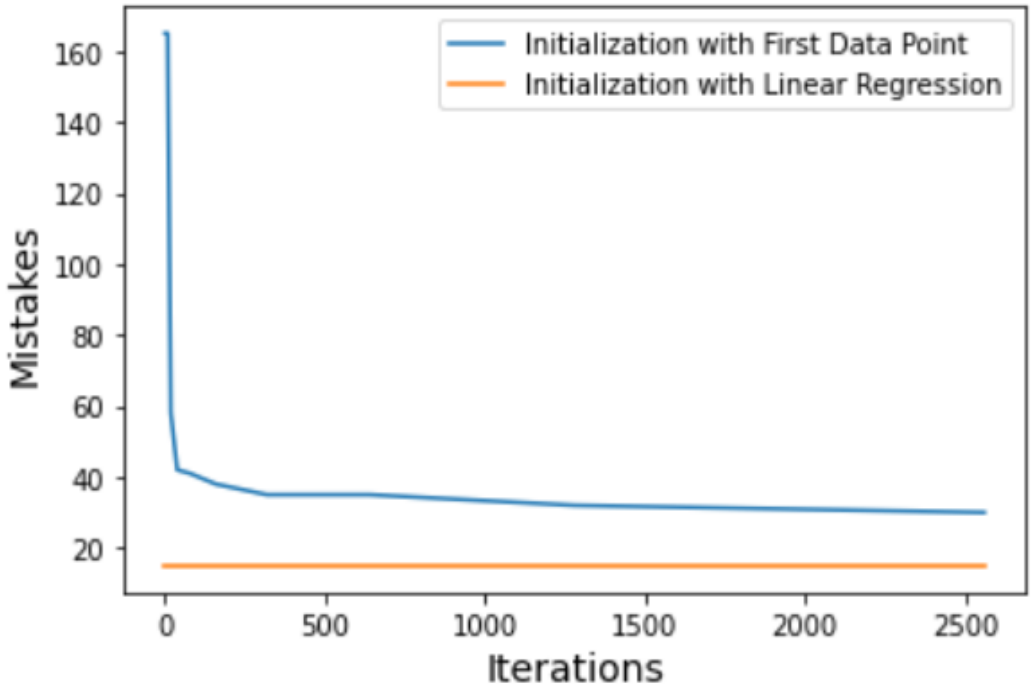
Also, our g hypothesis is impressive as the difference between E_in and E_out are small for both the methods and the E_in is very close to 0.

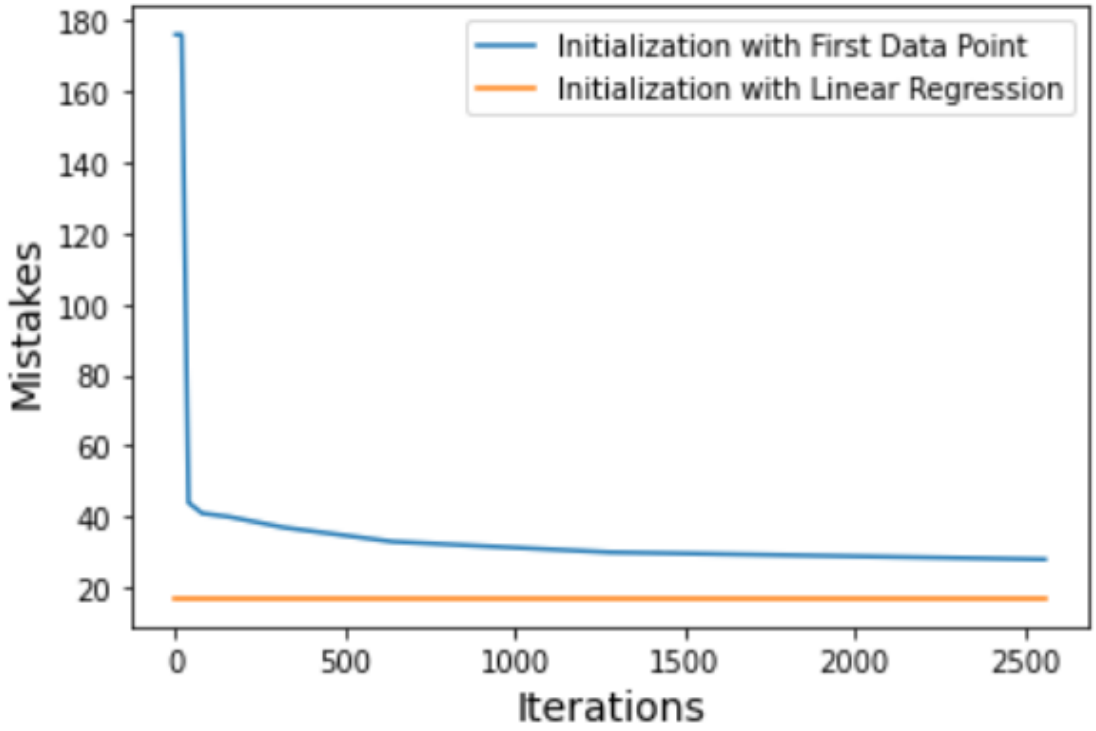
Observing the 5 graphs (below) of the performance plot of E_in, we notice that number of mistakes for the first method goes down as the number of iteration increases, indicating that our algorithm is learning.

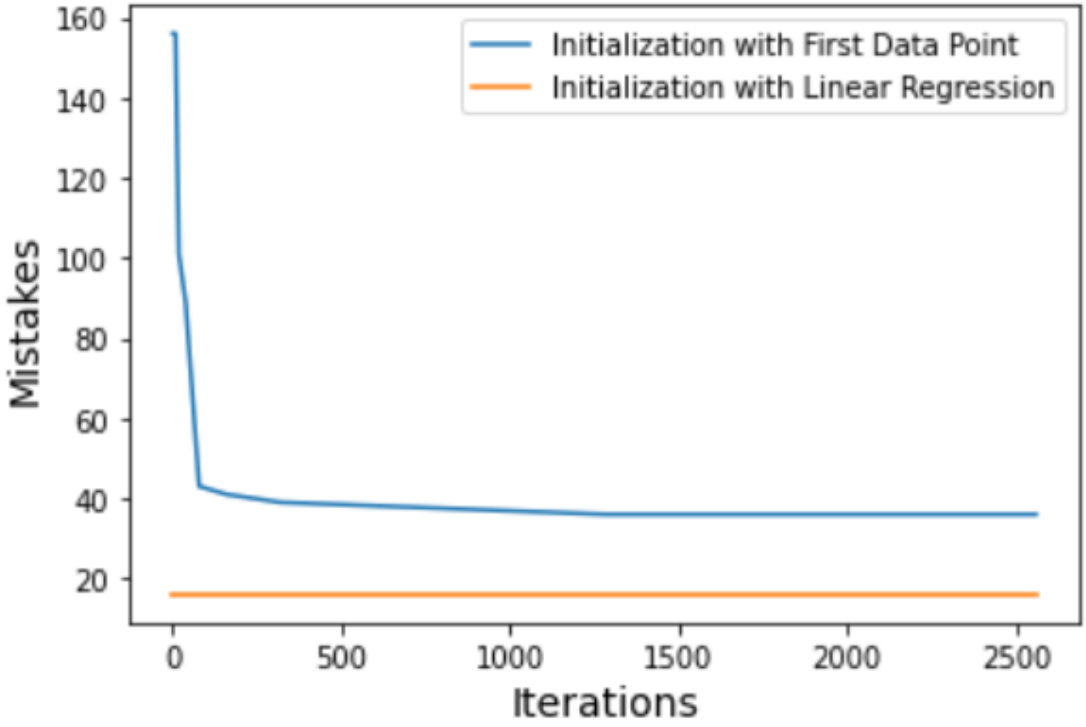
However, in the case for second initialization method, the number of mistakes starts at a significantly lower value and remains unchanged regardless of the number of iterations. As there can be noise in the dataset (the dataset is not linearly separable), this can be a probable reason why the number of mistakes does not go down.

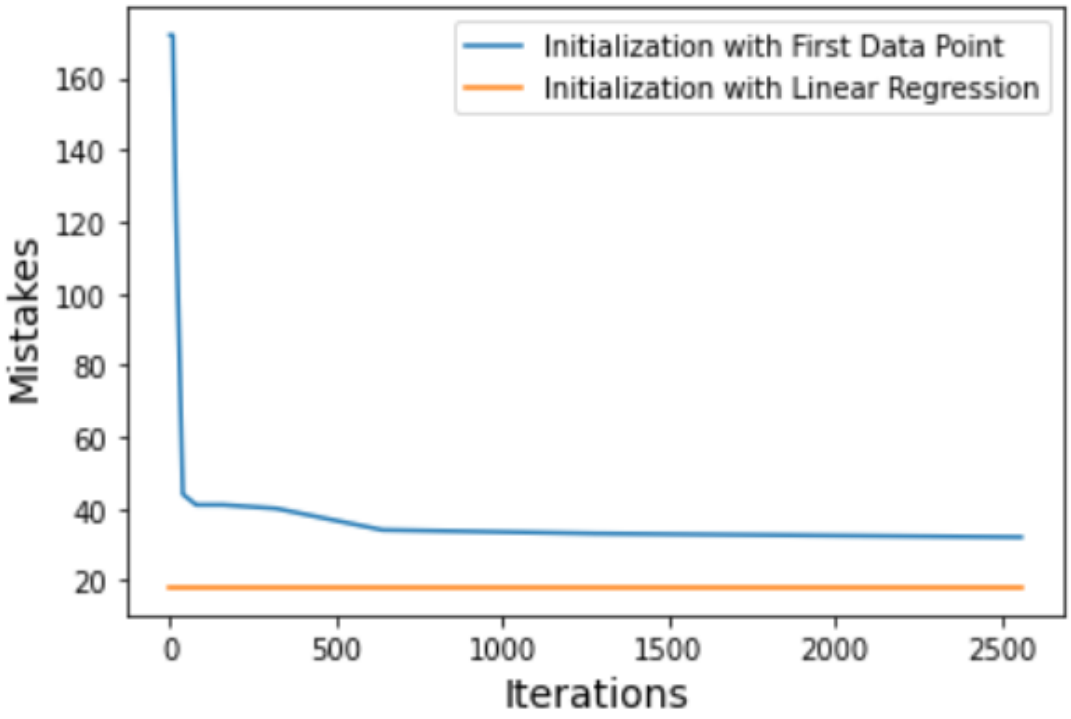
Result:


Initialization with First Data Point
Initialization with Linear Regression
Mistakes
Iterations
180
160
140
120
100
80
60
40
20
0
2000
4000
6000
8000
10000


Initialization with First Data Point
Initialization with Linear Regression
Mistakes
Iterations
160
140
120
100
80
60
40
20
0
500
1000
1500
2000
2500


Initialization with First Data Point
Initialization with Linear Regression
Mistakes
Iterations
180
160
140
120
100
80
60
40
20
0
500
1000
1500
2000
2500


Initialization with First Data Point
Initialization with Linear Regression
Mistakes
Iterations
160
140
120
100
80
60
40
20
0
500
1000
1500
2000
2500


Initialization with First Data Point
Initialization with Linear Regression
Mistakes
160
140
120
100
80
60
40
20
0
500
1000
1500
2000
2500
Iterations