Machine learning

Attached file: hw3.ipynb

First exercise:

i	$x_1^{(i)}$	$x_2^{(i)}$	$y^{(i)}$	
1	0	-1	0	
2	1	0	0	
3	2	1	0	$N_0 = 3$
4	1	1	1	
5	-1	1	1	
6	-1	-1	1	$N_1 = 4$
7	-1	-1	1	

$$w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

$$w^T x^{(i)} = w_1 x_1^{(i)} + w_2 x_2^{(i)}$$

$$\mu_0 = \frac{1}{N_0} \sum_{y^{(i)}=0} x^{(i)} = \frac{1}{N_0} \begin{bmatrix} \sum_{y^{(i)}=0} x_1^{(i)} \\ \sum_{y^{(i)}=0} x_2^{(i)} \end{bmatrix} = \frac{1}{3} \begin{bmatrix} 0+1+2 \\ -1+0+1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\mu_1 = \frac{1}{N_1} \sum_{y^{(i)}=1} x^{(i)} = \frac{1}{N_1} \begin{bmatrix} \sum_{y^{(i)}=1} x_1^{(i)} \\ \sum_{y^{(i)}=1} x_2^{(i)} \end{bmatrix} = \frac{1}{4} \begin{bmatrix} -1-1-1-1 \\ 1+1-1-1 \end{bmatrix} = \begin{bmatrix} -1/2 \\ 0 \end{bmatrix}$$

$$\mu_0' = w^T \mu_0 = w_1 \mu_{01} + w_2 \mu_{02} = w_1$$

$$\mu_1' = w^T \mu_1 = w_1 \mu_{11} + w_2 \mu_{12} = -\frac{1}{2} w_1$$

$$S_0'^2 = \sum_{y^{(i)}=0} (w^T x^{(i)} - \mu_0')^2 = \sum_{y^{(i)}=0} (w_1 x_1^{(i)} + w_2 x_2^{(i)} - w_1)^2 =$$

$$(0 - w_2 - w_1)^2 + (w_1 + 0 - w_1)^2 + (2w_1 + w_2 - w_1)^2 = 2(w_1 + w_2)^2$$

$$S_1'^2 = \sum_{y^{(i)}=1} (w^T x^{(i)} - \mu_1')^2 = \sum_{y^{(i)}=1} (w_1 x_1^{(i)} + w_2 x_2^{(i)} + \frac{1}{2} w_1)^2 =$$

$$(w_1 + w_2 + \tfrac{1}{2} w_1)^2 + (-w_1 + w_2 + \tfrac{1}{2} w_1)^2 + (-w_1 - w_2 + \tfrac{1}{2} w_1)^2 \times 2 =$$

$$(\tfrac{3}{2} w_1 + w_2)^2 + (\tfrac{1}{2} w_1 - w_2)^2 + 2(\tfrac{1}{2} w_1 + w_2)^2 = 3w_1^2 + 4w_2^2 + 4w_1 w_2$$

$$J(w_1, w_2) = \frac{(\mu_0' - \mu_1')^2}{S_0'^2 + S_1'^2} = \frac{\frac{9}{4} w_1^2}{5w_1^2 + 6w_2^2 + 8w_1 w_2}$$

$$\frac{\partial J}{\partial w_1} = \frac{9}{4} \times \frac{2w_1(5w_1^2 + 6w_2^2 + 8w_1w_2) - w_1^2(10w_1 + 8w_2)}{(5w_1^2 + 6w_2^2 + 8w_1w_2)^2} = 0$$

$$10w_1^3 + 12w_1w_2^2 + 16w_1^2w_2 - 10w_1^3 - 8w_1^2w_2 = 0$$

$$4w_1w_2(3w_2 + 4w_1 - 2w_1) = 0$$

$$\rightarrow \begin{cases} w_1 = 0 \\ w_2 = 0 \\ 3w_2 = -2w_1 \end{cases}$$

$$\frac{\partial J}{\partial w_2} = \frac{9}{4} \times \frac{0 - w_1^2(0 + 12w_2 + 8w_1)}{(5w_1^2 + 6w_2^2 + 8w_1w_2)^2} = 0$$

$$w_1^2(12w_2 + 8w_1) = 0$$

$$\rightarrow \begin{cases} w_1 = 0 \\ 3w_2 = -2w_1 \end{cases}$$

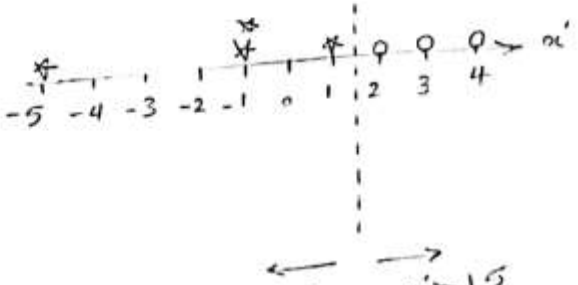
$$3w_2 = -2w_1 \qquad 2w_2^2 + w_1^2 = 1 \rightsquigarrow \frac{4}{9}w_1^2 + w_1^2 = 1 \rightsquigarrow w_1 = \frac{3}{\sqrt{13}}$$

$$w_2 = \frac{-2}{\sqrt{13}}$$

$$\uparrow \begin{bmatrix} 3 \\ -2 \end{bmatrix}$$

That gives us the unit vector. Wanted direction is -2x = 3y on xy plane. All data must be measured on this direction.

	x_1	x_2	$x' = w^T x$
1	0	-1	2
2	1	0	3
3	2	1	4
4	1	1	1
5	-1	1	-5
6	-1	-1	-1
7	-1	-1	-1



$x' \leq 1.5$ class ✱ $y = 1$

$x' \geq 1.5$ class O $y = 0$

Final classification model:

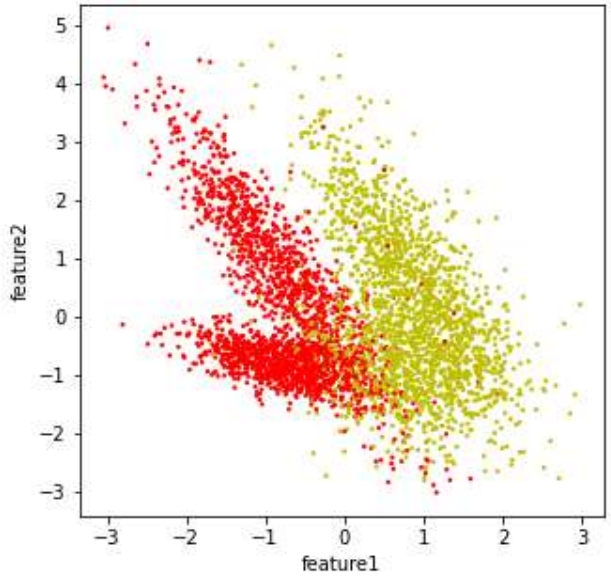
$$f(x_1, x_2) = 3x_1 - 2x_2 + 1.5$$

results on training data shows zero misclassified cases which is accuracy of 100 percent.

Second exercise:

The model of each step was built on the data of the train.csv file and the parameters of the model were evaluated with the test.csv data. This dataset has three columns, the first two columns, feature1 and feature2, are used as input features, and I labeled each data as a label, which has a value equal to 0 or 1, which I considered -1 and 1 for convenience. I also added a column 1 to make model bias calculations easier. These changes are made on both test and training groups before working with datasets. The category of both stages is a perceptron, whose learning rate I took as 0.001, and the condition is to stop the learning repetition algorithm and update the weights, or the number of misclassified data becomes zero, or repeat the process a maximum of 100 times. According to the confusion of the data of this problem, the first condition does not happen because the data cannot be separated with a linear model.

The graph of the two features on the figure below is red data such as label -1 and yellow data such as label 1.



The perceptron model of this step is a triple vector as follows.

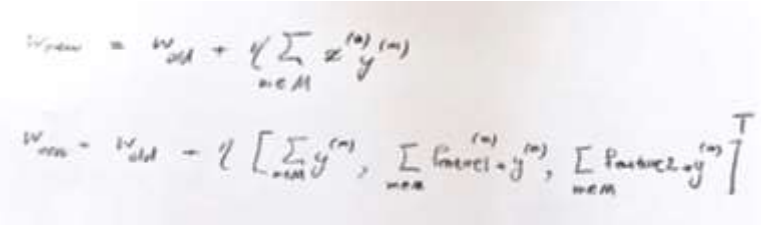
$$w_0 + w_1 \text{Feature 1} + w_2 \text{Feature 2} = 0$$

$$\forall\ x = [a, b]$$

$$\begin{cases} w_0 + w_1 a + w_2 b \geq 0 \rightarrow \text{assigned label} = 1 \\ w_0 + w_1 a + w_2 b < 0 \rightarrow \text{assigned label} = -1 \end{cases}$$

$$\Rightarrow \quad w = [w_0, w_1, w_2]^T$$

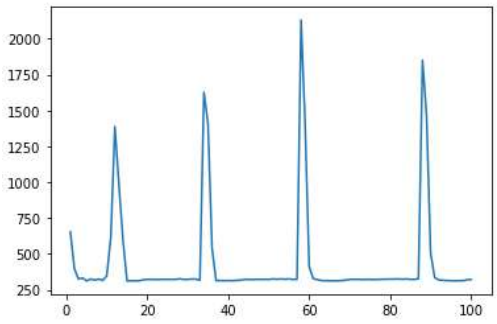
These values are obtained by updating the weights every time with all training data (batch mode) from the following relationship.



This model was implemented in the train_perceptron function. Each time, with the parameters obtained in the previous step, with the classify function, it obtains the label of the entire training set with the XW relation and places it in the 'assigned_label' column. Then, by comparing the calculated value with the data label, the find_missed function determines in the 'missclass' column whether the desired data is classified correctly or incorrectly. And then the update of the weights is calculated from the above relationship.

The result of vector data training is as follows. Also, the number of classification errors on the training data at each stage during the iterations is given below.

W0	W1	W2
0.096305	1.383040	0.372819



If the algorithm stopped at one of the peaks of the graph, the model would have a high error and could not be used. Better than the stop condition is a combination of error and number of repetitions. Here, my model does not suffer from such a problem, so the results can be analyzed with this condition.

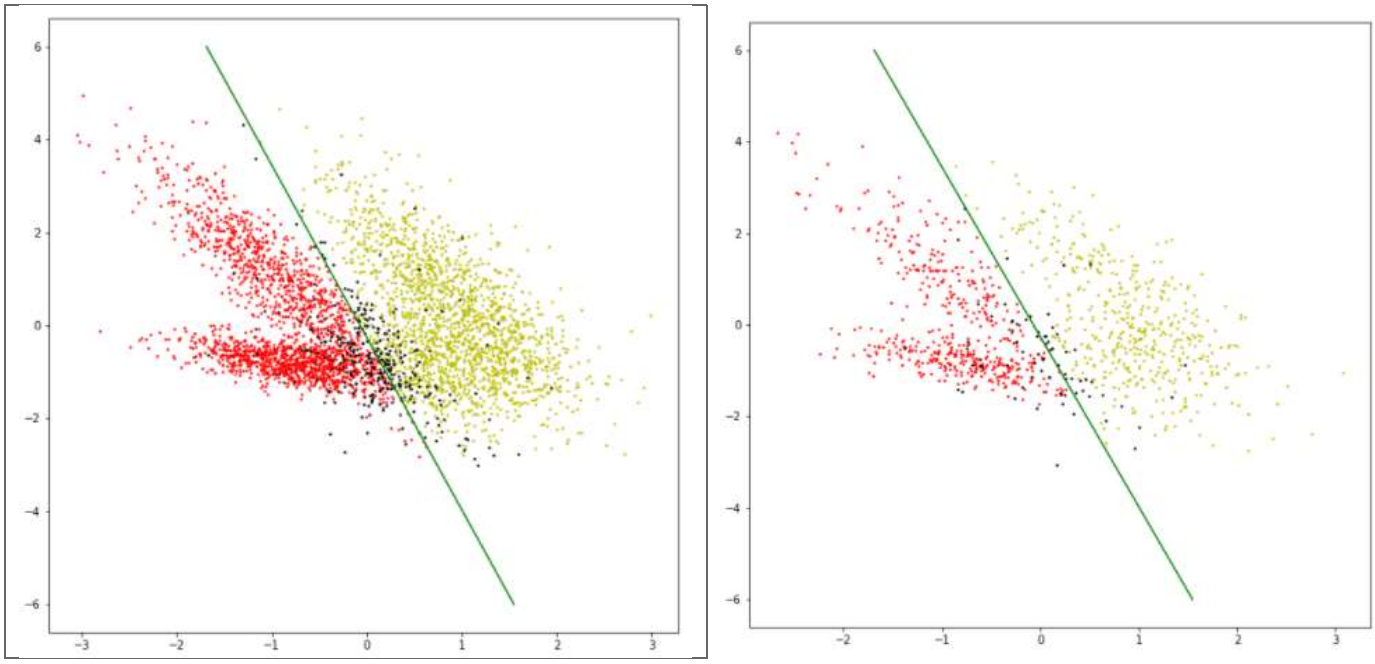
I fit the test and training data with the model and by counting the number of misclassifieds, the following accuracy and error criteria are obtained, and also on the graphs of both sets of test and training data, the green line is the perceptron model and the colored data are the data that are correctly classified and the black points are the points that the model estimated the error result for their class names.

train

```
total =
4000
total_missclassed =
323
misclassed_ratio =
0.08075
correct_ratio =
0.91925
```

test

```
total =
1000
total_missclassed =
73
misclassed_ratio =
0.073
correct_ratio =
0.927
```



Although the use of linear classification is not the best model for classifying these data (because they do not have a linear boundary), the performance of the model on both test data and training data has an accuracy higher than 90%. If I choose a more complex model that, in addition to the linear relationship between the features with the label, takes into account the effect of other factors such as the product of mixed features or their non-linear relationships with the label, the accuracy of the model will be higher on the training data as well. As long as it does not suffer from overfitting error and extreme complexity of the model.) But even this linear model has a good performance.

Using LDA, I move the X vector that has two features to a space with one less dimension. That is, I convert all the data into a number and fit a perceptron model on this number. This time my perceptron model will be as below and has two parameters.

$$w_0 + w_1 * feature\_lda = 0$$

$$\forall\ x = feature\_lda$$

$$w_0 + w_1 x \geq 0 \rightarrow \text{assigned label} = 1$$

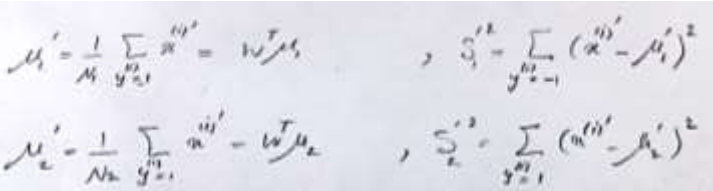
$$w_0 + w_1 x < 0 \rightarrow \text{assigned label} = -1$$

To calculate the LDA characteristic of the data, I need to find the vector in the direction that if the data of two classes are written on that direction, they have the maximum distance from the data of the opposite class and the minimum distance from the data of the same class. For this purpose, I am looking for a vector in the two-dimensional space of feature1 and feature2 that maximizes the following function.

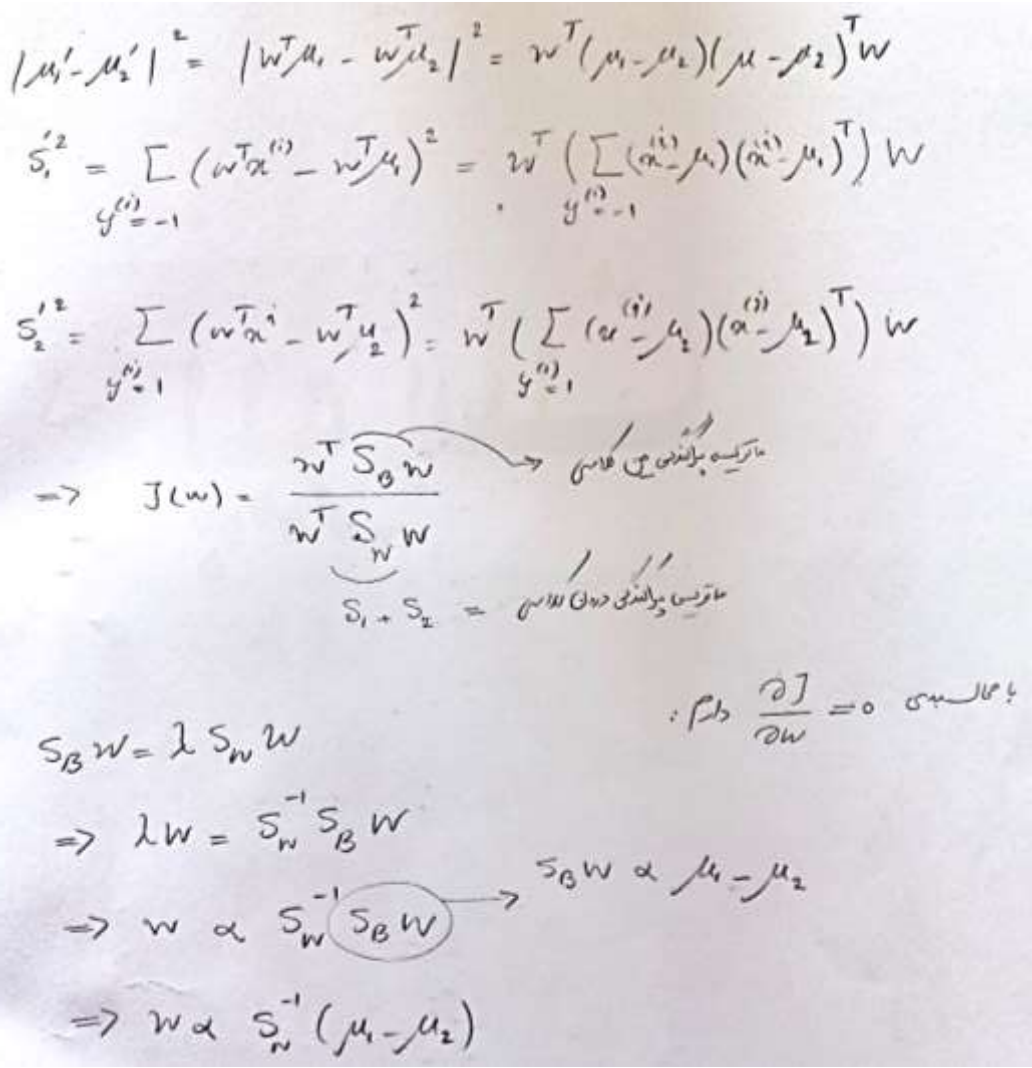
$$J(w) = \frac{(\mu_1' - \mu_2')^2}{s_1'^2 + s_2'^2} \quad , \quad w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

$$X = \begin{bmatrix} feature\ 1 \\ feature\ 2 \end{bmatrix} \longrightarrow X' = w^T X \in \mathbb{R}$$

that in this function:



What I need to calculate the LDA of the data is the alignment of the vector. So I don't need to calculate the maximum value. Just getting the alignment is enough. According to the following relations:



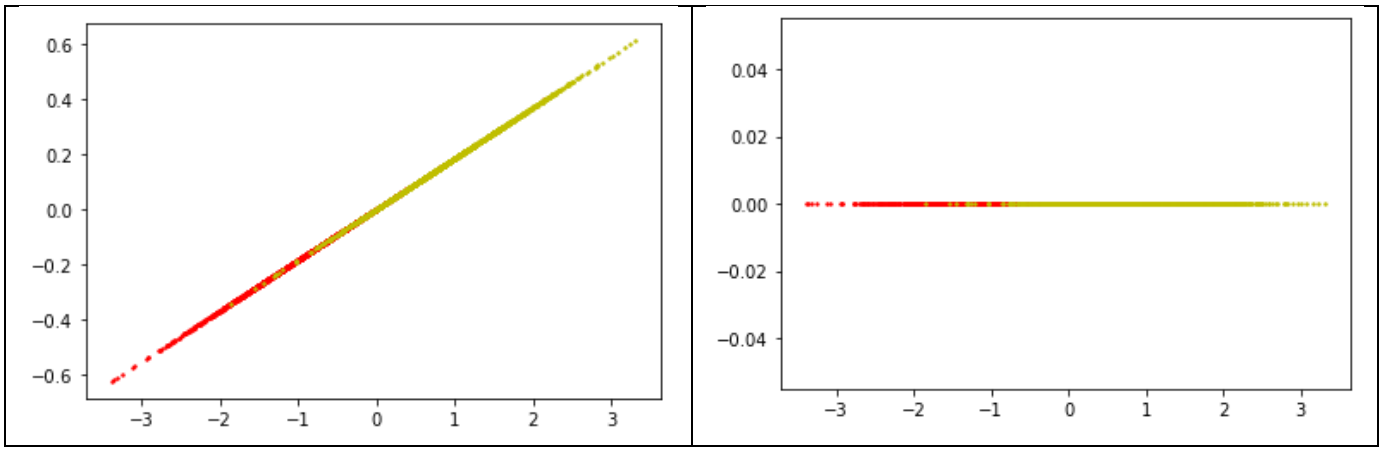
It is enough to calculate the inverse of the SW matrix and the difference of the averages of the two characteristics of the training data to find the line of w. The LDA(d) function implements this task and maps all the data on this line and keeps the obtained number in a column named lda on the data. This column will be the input of the new perceptron for model building and classification. Finally, it returns a vector in the desired direction.

Before feeding the data to LDA, I perform a required standardization on the feature1 and feature2 columns of the following relation. (The same standardization must be done for the test data before sending it to the LDA function with the same mean and variance obtained from the training data. Calculate this mean and variance of the training data from the mean and variance of the original and unknown distribution of the data in I commented.)

The obtained vector w:

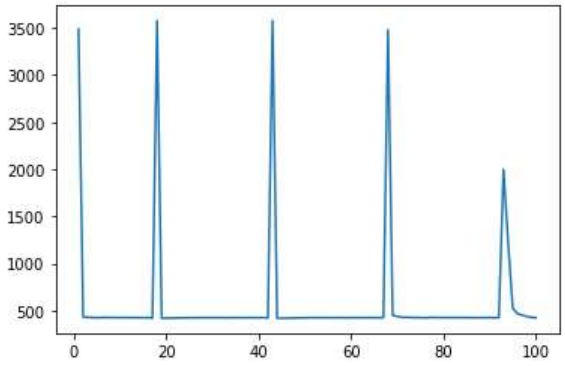
0.98342694
0.18130485

We see the value of the lda column shown on the horizontal line, and if I find this number for each data on the w line, the line obtained in the figure below is the same line that maximized the J function. When working on the test data and evaluating the model, the lda value of the test data will be calculated with this vector.



I teach the perceptron model with the same method as explained in the previous part. As mentioned above, this model has one input and two parameters, and this input is lda. The obtained model:

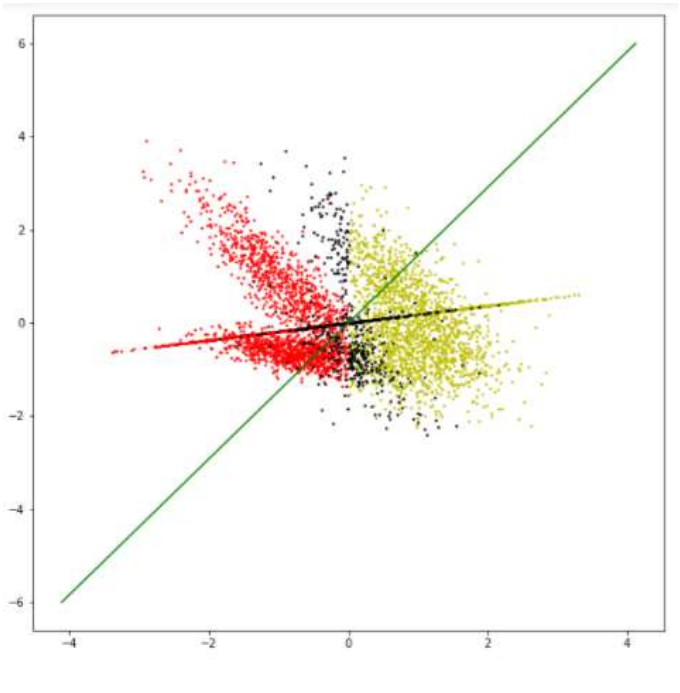
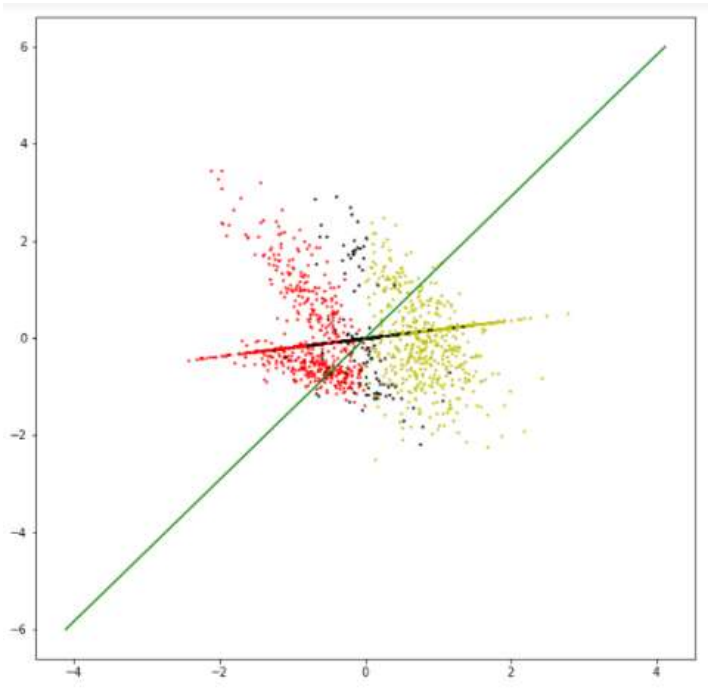
W0	W1
-0.006724	1.456784



Regarding the peaks of the blue graph and the stopping condition of the algorithm, what I said is still true, but here, since it stopped on a small error, I will continue the work. The obtained perceptron model vector has almost zero bias. In fact, this model works on one-dimensional data, which is obtained from fitting on a vector, and the hyperplane of this space is a point, so it becomes a constant value or threshold, according to which a step function The value differentiates between two classes. (The step function here is sign.)

Fitting the test and training data has the following results. On the diagrams in this section, the green line is the perceptron model. The second line is the mapping of lda values on the w line. The point of intersection of these two lines is the dating criterion. That is, for the data whose lda value is on the left side of that point, they will be placed in the red category -1 and otherwise, they will be placed in the yellow category -1. Colored

data is correctly classified data and black data is incorrectly classified data. It should be noted that the scattered points shown (outside the line) are the main points of feature1 and feature2, and they were displayed only to determine whether they were classified correctly or incorrectly, so that the comparison of this state with the state without LDA would be more compact. These feature1 and feature2 values are not directly entered in the model, and the model here is a two-parameter perceptron that only works on the data of the lda line. And the input data space is one-dimensional.

train	***test***
total = 4000 total_missclassed = 428 misclassed_ratio = 0.107 correct_ratio = 0.893	total = 1000 total_missclassed = 99 misclassed_ratio = 0.099 correct_ratio = 0.901
	

The general status of classification on both data series is very close to each other as before. If I consider the horizontal line graph of the test and training data, it is clear that a threshold point cannot be found on this single point data on which the model is built. So, the one-dimensional LDA data of this problem is still not separable into a linear state. And on the two graphs above and the black dots, if I consider the LDA line alone and without slope, I have the same horizontal line that I drew earlier for the training data, with the estimated values of the model instead of the real labels. are drawn and the black color is all the points whose real class does not apply to the threshold found for this model. Despite this confusion of data, the model still does not perform so badly. But if I compare the results with the results of the previous perceptron model, although the difference is not very big, a significant drop in the accuracy of the model has been found in both the test and training data. In the previous section, I said that the main error of the model is due to its low complexity. What happened was that by making the data of the problem one-dimensional, the model and its inputs became a little simpler, and the separation that the previous model was able to achieve on the data with two characteristics in the scalar mode of the data and with the second model, which is only from one The point

that was formed became narrower. The result is that the accuracy of the model drops by about three percent on the test data of the second model compared to the first. Using LDA on these data does not give a better result.