Implementation

Linear Classifier using the Perceptron algorithm without bias

The data sets that needed to be classified were generated from two Bivariate Gaussian distributions with the same covariance matrix $C = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$ and different means $m1 = \begin{bmatrix} 0 & 5 \end{bmatrix}$,$m2 = \begin{bmatrix} 5 & 0 \end{bmatrix}$respectively. In order to classify the type of the generated and given data, the perceptron algorithm was developed since the data is linear separable.

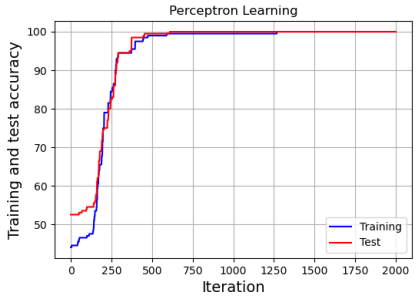

Figure 1: correct percentage in each iteration

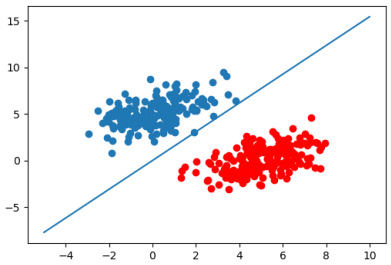

Figure 2: distribution and classification result

With *iteration* = 2000 and learning rate $\alpha = 0.002$, the correct classification percentage of the training set and test set both reached 100% as **Figure 1** shows. And the classification result and its decision boundary could be witnessed in **Figure 2**.

Linear Classifier using the Perceptron algorithm with bias

The data sets that needed to be classified were generated from two Bivariate Gaussian distributions with the same covariance as before but different means $m1 = \begin{bmatrix} 2.5 & 2.5 \end{bmatrix}$,$m2 = \begin{bmatrix} 10 & 10 \end{bmatrix}$. The same solution was used to do the classification.

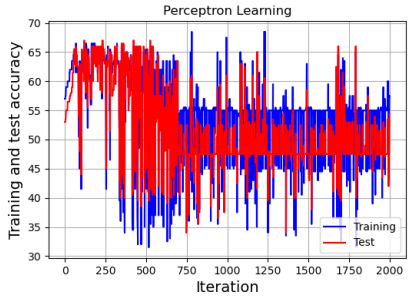

Figure 3: correct percentage in each iteration without bias

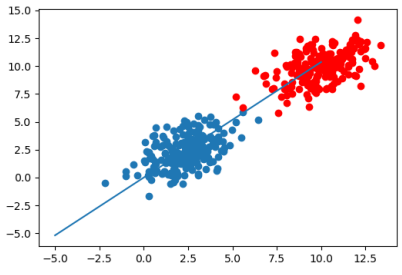

Figure 4: distribution and classification result without bias

Without the parameter bias, the decision boundary must pass through the origin which limited the performance of the algorithm as **Figure 3** and **Figure 4** shows.
By adding bias as one of the parameter in weight matrix, the accuracy on training and test sets was drastically improved as **Figure 5** and **Figure 6** shows.

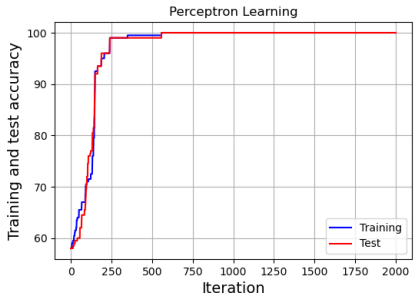

Figure 5: correct percentage in each iteration with bias

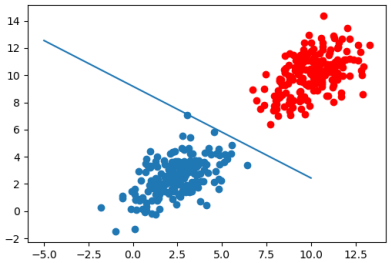

Figure 6: distribution and classification result with bias

Linear Classifier in scikit-learn

By using scikit-learn package, the procedure to develop a perceptron algorithm could be simplified. **Figure 7** shows the classification result.

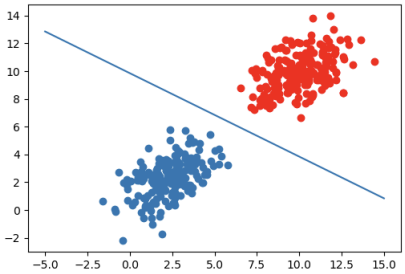

Figure 7: classification result by using scikit-learn

Iris Classification

The dataset of iris used to test my own perceptron algorithm is from UCI datasets. In order to fit it in the perceptron algorithm, only two classes and attribute of petal length and width were used for training and test.

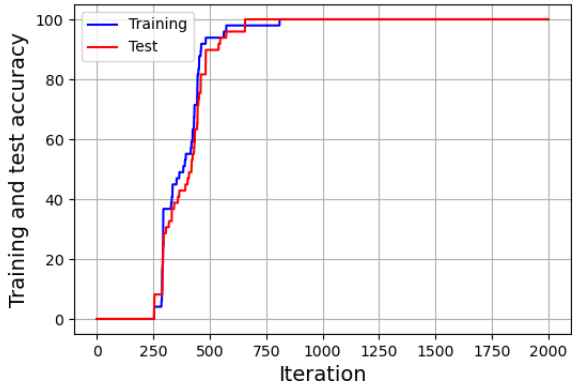

Figure 8: correct percentage in each iteration

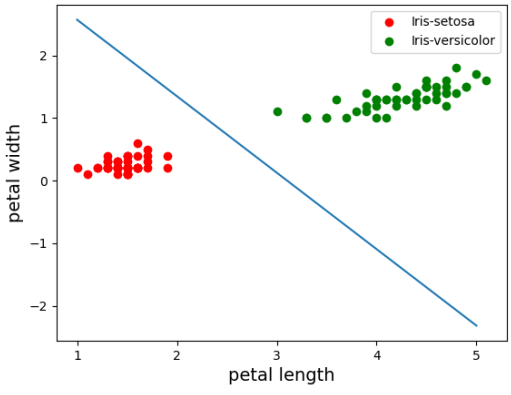

Figure 9: classification result

With *iteration* = 2000 and learning rate $\alpha = 0.001$, the correct classification percentage of the training set and test set both reached 100% as **Figure 8** shows. **Figure 9** demonstrates the classification result and the decision boundary.