Coursework 1 Report

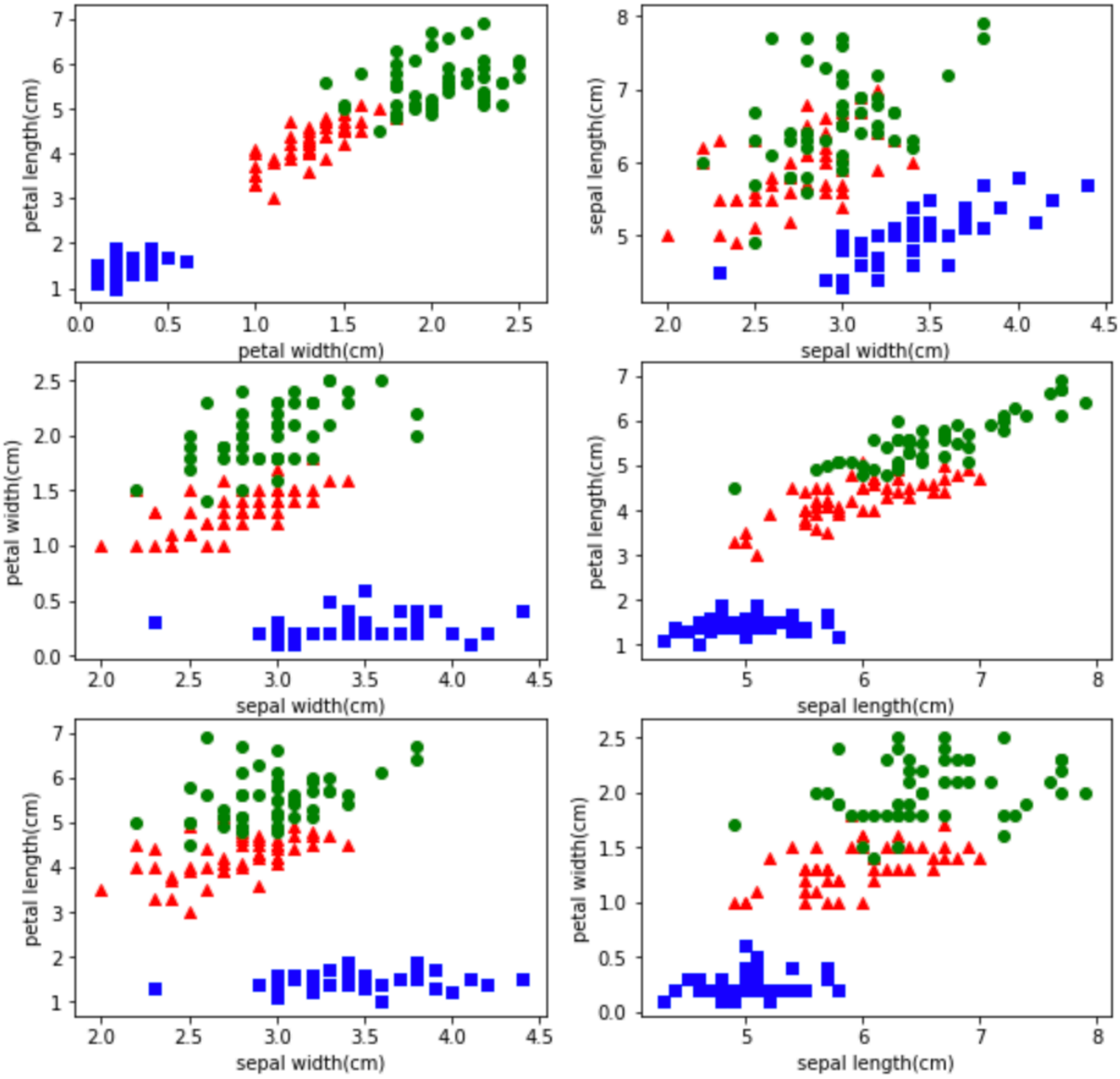
COMP5400M

201210146

ml17j43l

Question 1

Scatter plot for all sepal/petal length/width combinations



This plot are produced by matplotlib package ,Python. The code are provided in the attachment.

Blue squares denote setosa

Green circles denote virginica

Red triangles denote versicolor

Question 2

For setosa and non-setosa, I believe that the perceptron can classify it perfect when we have the input set consist of 4 features, for petal length, petal width , sepal length and sepal width.

Base on the plot I draw previous, we can notice that the blue squares which denote setosa are linearly separable with other types of irises. A perceptron algorithm normally have pretty good performance on this kind of task.

Giving weight weight and bias are :

Sepal length : -0.16887743

Sepal width: -0.45897766

Petal length: 0.80908125

Petal width: 0.4307757

Bias is 0.041

Question 3

The implement details are in the code which as attachment.

Setosa vs. Non-setosa :

Result:

```
perceptron('Iris-setosa',1000)
```

```
This is a standard perceptron without learning rate
Perceptron for  Iris-setosa
The initial weights is:  [0.45112257 0.22102234 0.36908125 0.2907757 ]
x has length:  150
565 times training
For all data in the iris set, the prediction correct for 150.0 times
The accuracy is 1.0
weight is  [-2.44887743 -5.87897766  8.46908125  3.3907757 ]  bias is  1.2410779275027994
```

1. I expect the algorithm converge

2. This type of data is linear and separable, and the margin is very obvious.

3. It converges . For accuracy 100%

4.After adding learning rate and stopping criterion:

Result:

```
In [519]: perceptron_lambda('Iris-setosa',0.2,0.99,1000)

          This is a perceptron with learning rate
          Perceptron for  Iris-setosa
          The initial weights is:  [0.45112257 0.22102234 0.36908125 0.2907757 ]
          19 times training
          For all data in the iris set, the prediction correct for 150.0 times
          The accuracy is 1.0
          weight is  [-0.16887743 -0.45897766  0.80908125  0.4307757 ]  bias is  0.04107792750279943
```

Leaning rate : 0.2

Stopping criterion: stop when the accuracy reaches 99% or training times.

Result : the algorithm converge, accuracy is 100% after 19 times weight updating

This classification task is a linearly separable problem , So the algorithm should be converge with sensible learning rate and stopping criterion

virginica vs. non-virginica

Result:

```
perceptron('Iris-virginica',10000)

This is a standard perceptron without learning rate
Perceptron for  Iris-virginica
The initial weights is:  [0.45112257 0.22102234 0.36908125 0.2907757 ]
x has length:  150
10000 times training
For all data in the iris set, the prediction correct for 147.0 times
The accuracy is 0.98
weight is  [ 58.65112257  55.52102234 -83.63091875 -94.7092243 ]  bias is  -46.7589220724972
```

1. I do not expect the algorithm converge

2. According to the previous plot, this classification task is not completely linearly separable. This task exceeds the capabilities of the perceptron. In addition, without learning rate, each weight update is large, and the weight update process can easily cross the optimisation point. The weights thus obtained will reduce the accuracy of the prediction results and will not cause the algorithm to converge.

3. It does not converge, accuracy is 98% after 10000 times weight updating

4. After adding learning rate and stopping criterion:

Result:

```
In [520]: perceptron_lambda('Iris-virginica',0.05,0.99,1000)
```

```
This is a perceptron with learning rate
Perceptron for  Iris-virginica
The initial weights is:  [0.45112257 0.22102234 0.36908125 0.2907757 ]
507 times training
For all data in the iris set, the prediction correct for 149.0 times
The accuracy is 0.9933333333333333
weight is  [ 0.97612257  0.84602234 -1.46091875 -1.1392243 ]  bias is  0.6410779275027995
```

Leaning rate : 0.05
Stopping criterion: stop when the accuracy reaches 99% or training 1000 times.
Result : the algorithm converge, accuracy is 99.33% after 507 times weight updating.

According to the previous plot, this classification task is not completely linearly separable. This kind of task exceeds the range of capabilities of the perceptron. However, because there are not many points in the linear indivisible range, satisfactory results can be obtained by selecting the learning rate.

versicolor vs. non-versicolor

Result:

```
perceptron('Iris-versicolor',10000)
```

```
This is a standard perceptron without learning rate
Perceptron for  Iris-versicolor
The initial weights is:  [0.45112257 0.22102234 0.36908125 0.2907757 ]
x has length:  150
10000 times training
For all data in the iris set, the prediction correct for 115.0 times
The accuracy is 0.7666666666666667
weight is  [ -9.44887743  52.32102234 -27.93091875  64.5907757 ]  bias is  67.2410779275028
```

1. I do not expect it converge

2. According to the previous plot, this classification task is not completely linearly separable. This task exceeds the capabilities of the perceptron. In addition, without

learning rate, each weight update is large, and the weight update process can easily cross the optimisation point. The weights thus obtained will reduce the accuracy of the prediction results and will not cause the algorithm to converge.

3. It does not converge, accuracy is 76.67% after 10000 times weight updating

4. After adding an learning rate and stopping criterion

Result :

```
perceptron_lambda('Iris-versicolor',0.05,0.8,10000)
```

```
This is a perceptron with learning rate
Perceptron for  Iris-versicolor
The initial weights is:  [0.45112257 0.22102234 0.36908125 0.2907757 ]
3703 times training
For all data in the iris set, the prediction correct for 120.0 times
The accuracy is 0.8
weight is  [-0.84387743  2.35102234 -0.74091875  2.3307757 ]  bias is  1.9410779275028005
```

Leaning rate : 0.05

Stopping criterion: stop when the accuracy reaches 80% or training 10000 times.

Result : the algorithm converge, accuracy is 80% after 3703 times weight updating.

According to the previous plot, this classification task is not completely linearly separable. This kind of task exceeds the range of capabilities of the perceptron. So the algorithm can not performance well on this classification task.

The iris data in this category is very special and it is difficult to increase the rate of satisfaction to a satisfactory level. Through the observation of the previous plot, this type of data is in the middle of setosa and virginica in the matching of the two parameters. And it is inseparable from the linearity of virginica. Such a data set may be difficult to find a linear space in a four-dimensional space to separate it from other points.

Question 4

The implement details are in the code as attachments.

Strategy:

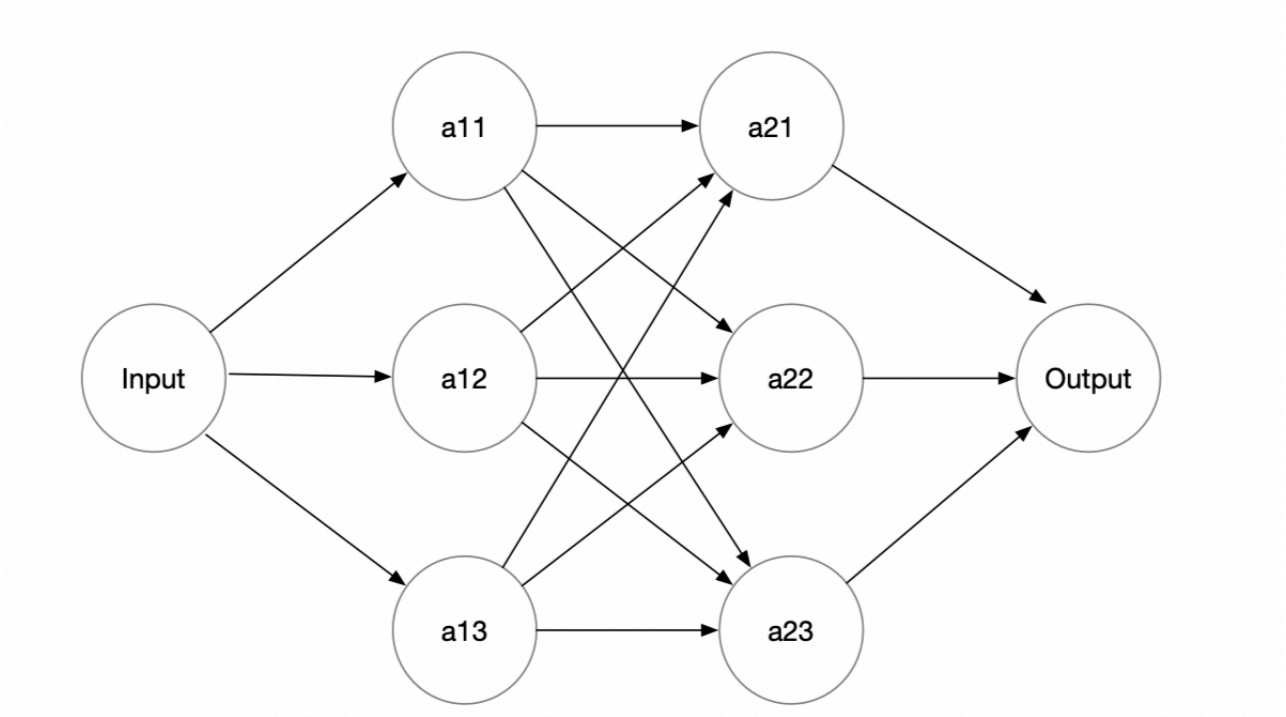
My strategy is to use a trained perceptron as the first layer of the neural network, without updating it's weights and biases. The three outputs of the first layer serve as inputs to the second layer, and three neurons are added to the second layer corresponding to three categories, using the same parameter update method as the standard perceptron. The output of each neuron in the second layer (0 or 1) is an integral part of the output layer

In the previous section, we implemented a binary classification of each category through a perceptron, but since the data is not completely linearly separable, some types of predictions are not good.

Since the backward propagation method cannot be used, I convert the predicted output of the neuron into (0,1), where 0 means not belonging to this class and 1 means that the prediction result belongs to this class.

It can be found that after the iris data is predicted using three different neurons, the three output sets are linearly separable in three dimensions. So this method should be effective

The network structure:



The weights:

For the first layer, w11 = [-2.80887743, 4.64102234, -1.01091875, 4.8707757]
b11 = 2.641077927502
w12 = [-0.04387743, -0.08897766, 0.23908125, 0.2557757]
b12 = 0.3410779275027994
w13 = [0.87612257, 0.79102234, -1.29591875, -0.9142243]
b13 = -0.10892207249720055

For the second layer, w21 = [0.15112257, 0.82102234, 0.66908125]
b21 = 0.30937304803019045
w22 = [0.5907757 , -0.35892207, 0.78204643]
b22 = 0.3290880984912425
w23 = [0.81109437, 0.70079054, -0.5740008]
b33 = 0.35192087642821096

Evaluation:

```
the accuracy is 0.98
-----------------------------
the training times is 28
-----------------------------
confusion matrix

            versicolor     setosa    virginica
Actual         50             50          50
Prediction     47             50          50
```

Accuracy : 98%
The training time: 28 times with learning rate 0.3

The accuracy of this neural network is not bad.

By observing the confusion matrix, we find that there are three cases that cannot be classified in the versicolor class (there are multiple 1s in the output), which I think is due to the data of versicolor.

Essentially, the parameters of this neural network are updated in the same way as neurons. The accuracy achieved now is the best result without using the activation function and the backward propagation method.

Since several data are different in label, they are too similar in parameters, and two-layer linear segmentation still cannot distinguish them.Especially the versicolor specie.

I think adding activation and backward propagation can solve this problem.