COMP5400 Coursework2

Pengyuan Huang, ml18p3h

April 20, 2020

1 Introduction

This is the introduction of this . file about this coursework. This compressed file contains the recorded .txt data file and contains three .py files: plot.py is used for ploting figures of average fitness and best fitness, and separate.py is used to separate the data of prey and predator to generate two .txt files, compare.py is to put the curves of prey and predator in the same figure for comparsion. As for .png files, they are basically average fitness and best fitness figures drawn by Python, and they will appear in this report as well. The mouse network.cc is mouse.cc after using DynamicalNet instead of FeedForwardNet, and the mouse proximity.cc is mouse.cc after adding the proximity sensor, the mouse fitness.cc is mouse.cc after changing fitness function. Finally, COMP5400 CW1.tex is a LaTex source file, and this COMP5400 CW1.pdf report is generated after compiling .tex file.

2 Question 1

Make a scatter plot of petal length against petal width. This means that for each entry in the data set you plot a point in the 2D plane with petal length as y coordinate, and petal width as x coordinate. Use three different markers (or different colours), one for setosa, one for versicolor and one for virginica. Plot other sepal/petal length/width combinations as well. [10 marks]

Answer:

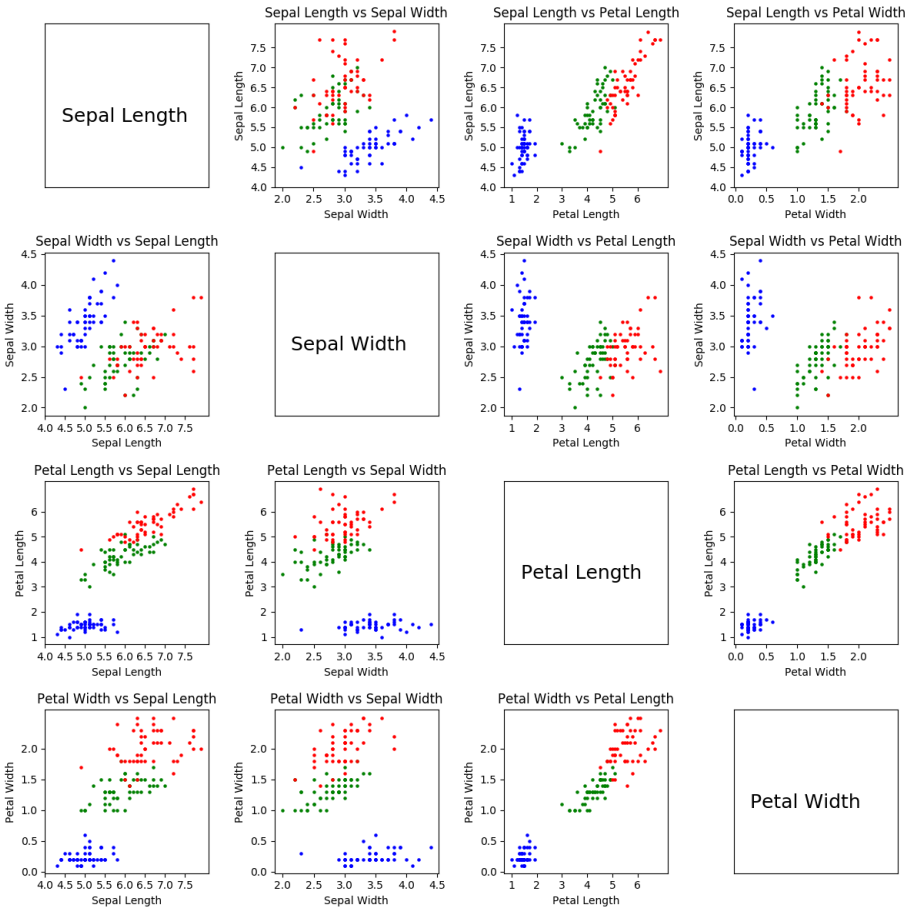


Figure 1: Iris Dataset (blue=setosa, green=versicolour, red=virginca)

3 Question 2

Based on the plots, do you believe that setosa vs. non-setosa can be learnt by a perceptron? That is, given a perceptron with 4 inputs, for petal length, petal width sepal length, and sepal width, can you find four weights and a bias which can classify setosa vs. non-setosa? Explain your answer. If you can, give these weights and bias. Draw the decision line of your perceptron on the plots of the last question. [15 marks]

Answer:
From any of the 12 diagrams above, we can all draw a straight line separating the points of the setosa class from the colors of the other two classes. This shows that the setosa class is linearly separated from the non-setosa class. Thus it can

be learned by the perceptron.
That setosa vs. non-setosa can even be classified by just two of these four features (petal length, petal width, sepal length and sepal width). There are many combinations of four weights and a bias which can classify setosa vs. non-setosa. For example, I choose the one in Q1, petal length against petal width. The figure as show below:

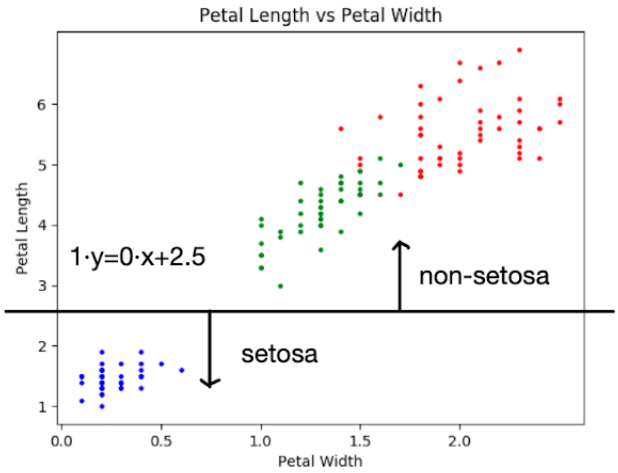


Figure 2: Petal Length vs. Petal Width

In this figure above, a line $1 * y - 0 * x - 2.5 = 0$ separates setosa and non-setosa. The points below this line are classified as setosa and the points above this line are classified as non-setosa. This line $1 * y - 0 * x - 2.5 = 0$ can be replaced to $1 * PL - 0 * PW - 2.5 = 0$. Thus, from the parameters of this line, these four weights and a bias are $W_{sl} = 0, W_{sw} = 0, W_{pw} = 0, W_{pl} = 1, bias = -2.5$.

4 Question 3

Implement the standard perceptron algorithm without learning rate and train a perceptron for the setosa vs. non-setosa classification problem. Do you expect the algorithm to converge? Explain why! Does the algorithm converge? Is the output correct? If it does not converge, now introduce a learning rate, and use a sensible stopping criterion. Report the learning rate, stopping criterion and argue whether the result you obtain is in line with what you know the algorithm is capable of. The same questions about virginica vs. non-virginica? versicolor vs non-versicolor? There is a major difference between versicolor and the other two. Explain the difference using the plots you made. [25 marks]

Answer:

4.1 setosa vs. non-setosa

I hope the algorithm to converge after training because it is easily to find a single dicision line to separate the points of setosa and the points of non-setosa in each subplot in Figure 1 of Q1. From the result we can conclude that the algorithm will converge and weights will not change frequently, with or without using learning rate. As figure shown below, with the learning rate 0.01, the accuracy after 150 epochs is nearly 96.67% (usually between 80% and 100%), and the number of misclassified points are only 5. The converge count is 135 means that weights are unchanged in 135 continuous epochs.

```
The converge count is 135
The weights is [ 0.20851083 -0.34759561 -0.09119818  0.33858504  0.0090486 ]
The number of misclassification is 5
The accuracy after 150 epochs is 96.6667%
The learning rate is 0.01
The perceptron was stopped after 150 epochs
```

Figure 3: result of separate setosa from non-setosa after 150 epochs

4.2 virginica vs. non-virginica

As can be seen from the 12 subplots of Figure 1 in the Q1, the points of virginica partially overlap with the points of versicolor in each subplot, so that there is no decision line or decision boundary to separate all points of virginica from the points of non-virginica completely in these 2D subplots. Thus, virginica and non-virginica are not lineraly separable and the algorithm cannot converge completely.
By using a appropriate learning rate, the weights can be converged compared to without learning rate. The perceptron can find a hyperplane to separate the points of virginica and non-virginica nearly, as shown below. However, there still a few misclassified points on the margin of the part of virginica and non-virginica because there is no single hyperplace in the 4D to separate all points of virginica from the points of non-virginica actually.

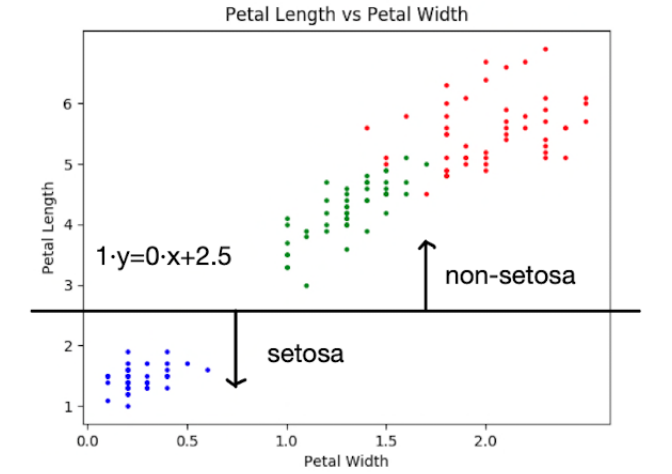


Figure 4: replace1

4.3 versicolor vs. non-versicolor

There is major difference between versicolor and the other two classes. Because as the Figure 1 shown above in Q1, the points of versicolor class lie between the points of the other two classes in each subplot. So, we cannot find just one decision line or decision boundary to almost separate the points of versicolor and non-versicolor, as shown below.

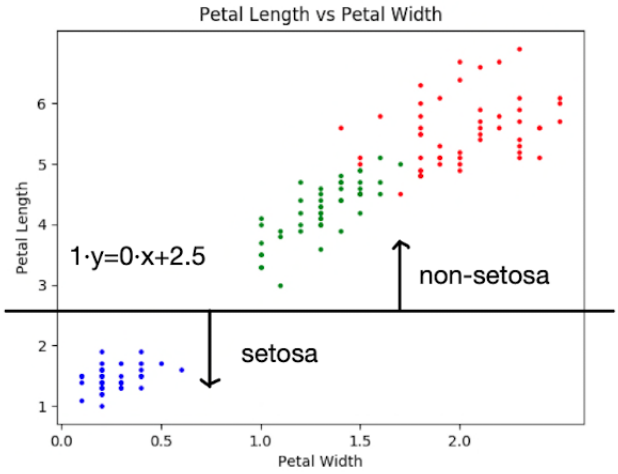


Figure 5: replace2

Hence, even we add learning rate to perceptron algorithm, the weights seem always change in each epoch which means that it cannot converge. The accuracy are between 49% to no more than 65%, and the number of misclassified points are always more than 50 (stopping criterion) when the training is not over.

5 Question 4

Using your earlier results, build a neural network with four inputs and three outputs. The network should produce the following desired classification: versicolor = (1,0,0), setosa = (0,1,0), virginica = (0,0,1). Do not use backpropagation. Combine the classifiers you have developed so far with some elementary logic - which you should implement using artificial neurons! Create a working program that implements your network. Your program must compile and run. Upon running, it must ask for 4 numbers: PL PW, SL, SW and produce a classification. In the report, explain the strategy you used, draw the full network and give all weights and biases. Evaluate the accuracy of your network.[20 marks]

Answer:
To build a neural network that produces the classification: versicolor = (1,0,0), setosa = (0,1,0), virginica = (0,0,1), we can combine the classifiers to build hidden layer.
1. At first, we can see that when the setosa and virginica output 0, the versicolor will output 1. This is a NOR gate. So, we can only use setosa and virginica to replace versicolor which means that there are only setosa and virginica neural in hidden layer without versicolor. Specifically, due to we define when output < 0, it will output for 0 and when output ≥ 0, it will output for 1. Hence, for

example, we can set setosa and virginica both output -1, and set bias to 0.5 to make balance to make virginica output 1 at the end. This a sub-neural network of versicolor as shown below.

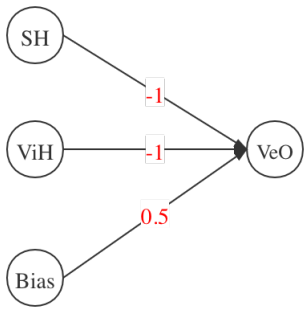


Figure 6: versicolor neuron

- SH: setora perceptron in the hidden layer
- ViH: virginica perceptron in the hidden layer
- VeO: versicolor in the output layer
- SO: setosa in the output layer
- ViO: virginica in the output layer
- PL: petal length
- PW: petal width
- SL: sepal length
- SW: sepal width
- Bias: biases in the input layer and hidden layer

2. Then, it is clear that the setosa neural will output 1 if the perceptron of setosa inputs 1 so that it is a OR gate actually. In addition, due to we want to get setosa = (0,0,1), so we should make other elements to 0 which means we should set the weight of virginica to 0. To make balance, we set the value of bias to -0.5. The neural shows as following:

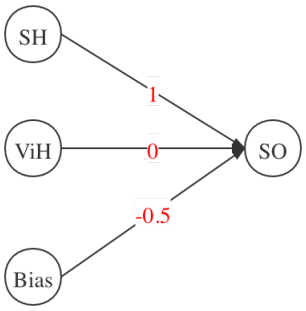


Figure 7: setosa neuron

3. Next, the virginica is same as the setosa. Thus, we set the weight of virginica to 1, and weight of setosa to 0, and bias to -0.5. There is the neural network below:

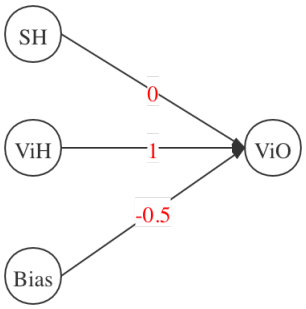


Figure 8: virginica neuron

4. The three outputs and hidden layer of the neural network have been built. Then, we should make 4 parameters: PL, PW, SL, SW as the input layer of this neural network. We use np.random.randn() to get 4 weights and a bias with standard normal distribution (the initialization resutls are different when we run it because it is random). Build the input layer by using these 4 weights and a bias as values for PL, PW, SL, SW and bias of input layer. The full neural network with all weights and biases as shown below:

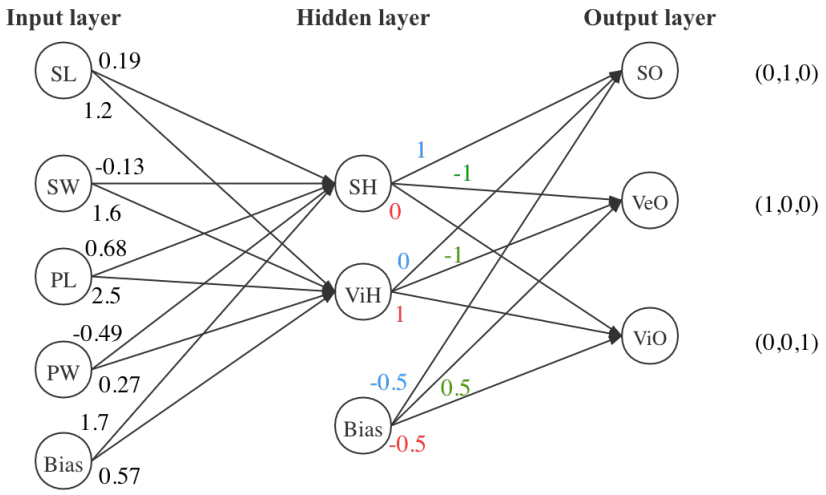
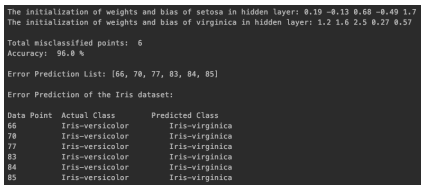
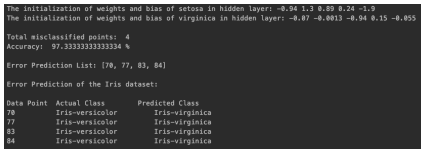


Figure 9: full network with all weights and biases

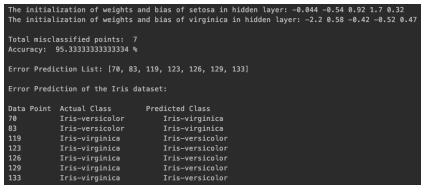
Because the weights are initialized randomly each time, the accuracy of each run is also different. I decide to run 4 times and calculate an average of the accuracy as a result. The accuracies for the four runs are: 96.0%, 97.3%, 95.3%,



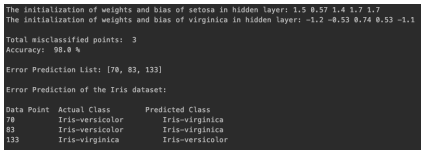
(a) accuracy1.



(b) accuracy2.



(c) accuracy3.



(d) accuracy4.

Figure 10: Accuracies

98.0%. Thus, the average of accuracies is 96.65%, and as shown in the figure, there are an average of $(6 + 4 + 7 + 3)/4 = 5$ misclassified points per run. The standard deviation is 1.0596%.

6 Question 5

A tutorial will demonstrate an application of the Keras framework to solve the XOR problem. This implementation will be made available to you. You will then adapt this implementation for use on the iris data set. Apply your implementation in a program that can classify the iris data. Create a demo program that can take 4 numbers PL PW, SL, SW and produce a classification. Evaluate the performance of your algorithm. Explain whether it is better or worse than the network you built in question 4. Give two reasons why your network cannot achieve perfect classification. Experiment with the number of hidden nodes. Consider experimenting with the loss function. Make sure you create a training data set, and not to evaluate your network on this training data set. [30 marks]

Answer:
Compared to AND or OR problems, XOR problem is inseparable, as shown below. However, perceptron is a linear classification model, so that perceptron cannot solve XOR problem. We choose use the Keras framework to solve the XOR problem.

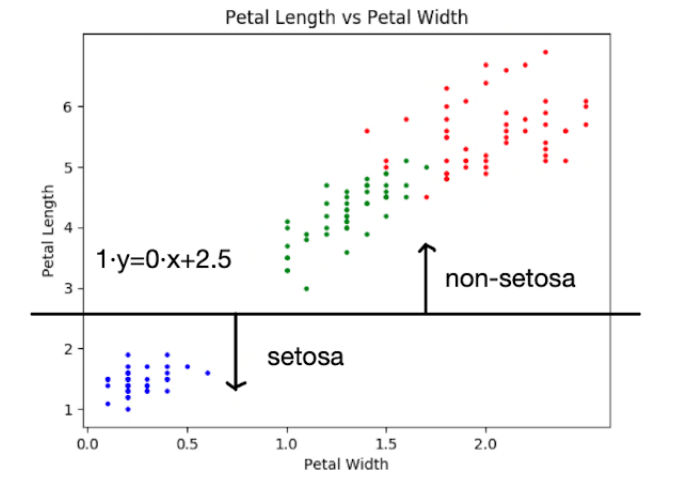


Figure 11: replace2