Name: Tianxiu Ma

Student number: 230277302

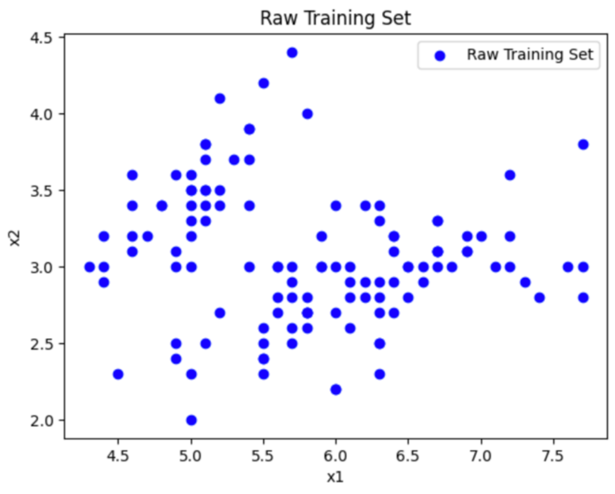
Assignment number: Assignment 1 part 2

module code: ECS708U/ECS708P

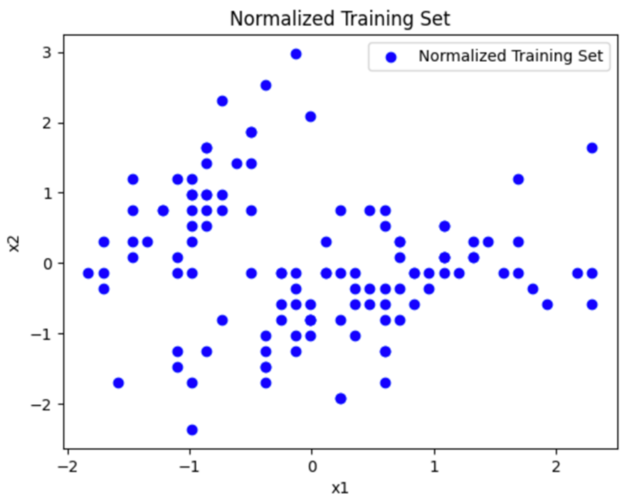
Assignment 1 part 2a: **Classification I**

Q1. We again notice that the attributes are on different scales. Use the normalisation method from last lab, to standardize the scales of each attribute on both sets. Plot the normalized and raw training sets; what do you observe? [2 marks]

Raw Training Sets:



Normalized Training Sets:

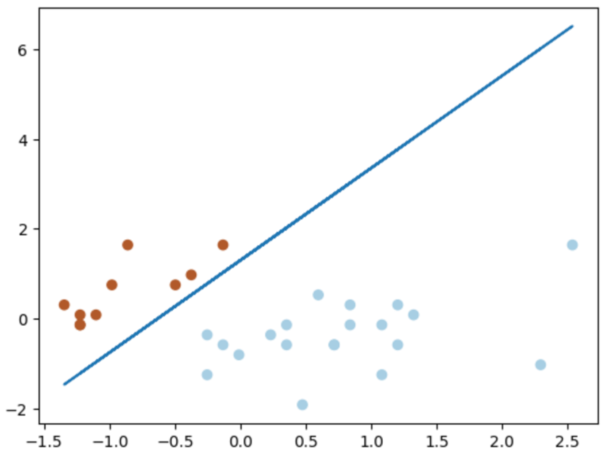


The normalized data has a mean of 0 and a standard deviation scaled to 1 for each feature. All features retain comparable scales to the original data, improving the model's numerical stability and limiting the impact of a feature with a wide value range. This facilitates faster convergence of optimization algorithms such as gradient descent.

Q5. Draw the decision boundary on the test set using the learned parameters. Is this decision boundary separating the classes? Does this match our expectations? [2 marks]

Yes, The decision boundary separates the classes. The decision boundary can make the behavior of the model more interpretable. Our expectation is to distinguish points belonging to different classes, and this decision boundary aligns with our expectations. This can help us interpret and evaluate the performance of machine learning.

The following is the image of the decision boundary:



Q6. Using the 3 classifiers, predict the classes of the samples in the test set and show the predictions in a table. Do you observe anything interesting? [4 marks]

```
    setosa_pred  versicolor_pred  virginica_pred predict_classes
0      0.036003         0.737879        0.453590      versicolor
1      0.999957         0.171286        0.109165          setosa
2      0.000003         0.839608        0.946120       virginica
3      0.038518         0.589673        0.706698       virginica
4      0.007307         0.766411        0.529304      versicolor
5      0.999632         0.319481        0.080462          setosa
6      0.282334         0.550244        0.530486      versicolor
7      0.001717         0.386465        0.969575       virginica
8      0.000417         0.916626        0.385456      versicolor
9      0.076291         0.733561        0.372187      versicolor
10     0.013133         0.348758        0.948923       virginica
11     0.999290         0.586456        0.021742          setosa
12     0.999906         0.306401        0.047724          setosa
13     0.999506         0.535002        0.025599          setosa
14     0.999984         0.129417        0.132330          setosa
15     0.155063         0.339944        0.861282       virginica
16     0.000843         0.464451        0.969029       virginica
17     0.040438         0.828096        0.252428      versicolor
18     0.055260         0.668095        0.559483      versicolor
19     0.000361         0.580371        0.955191       virginica
20     0.999678         0.421318        0.046553          setosa
21     0.015164         0.475615        0.888562       virginica
22     0.999746         0.282693        0.099666          setosa
23     0.000492         0.605201        0.940174       virginica
24     0.017080         0.204804        0.978110       virginica
25     0.001046         0.434555        0.968327       virginica
26     0.000104         0.851983        0.774552      versicolor
27     0.001280         0.346571        0.981928       virginica
28     0.998682         0.535828        0.039288          setosa
29     0.999323         0.501996        0.037438          setosa
```

Specific Observations:

- Rows where the predicted class is setosa (predict_classes column) generally have a very high probability in the setosa_pred column. Similar observations can be made for other classes; for example, rows with predicted class versicolor have a high probability in the versicolor_pred column.

- predict_classes: This column appears to show the predicted class for each instance. It derived from the class with the highest predicted probability.

summary:
It looks like the model is making predictions with high confidence, as the probabilities are close to 0 or 1 in many cases. When evaluated using the accuracy metric, the model demonstrates high performance.

Q7. Calculate the accuracy of the classifier on the test set, by comparing the predicted values against the ground truth. Use a softmax for the classifier outputs. [1 mark]

```
softmax_predictions = torch.softmax(concatenated_matrix, dim=1)
predicted_labels = torch.argmax(softmax_predictions, dim=1)

true_labels = torch.argmax(y_test, dim=1)

correct_predictions = (predicted_labels == true_labels).sum().item()

accuracy = correct_predictions / len(y_test)
print("Accuracy on the test set: {:.2f}%".format(accuracy * 100))
```

```
Accuracy on the test set: 90.00%
```

Accuracy of 90% means that the model correctly predicted the class labels for 90% of the samples in test set. It indicates better performance, but it might leave room for improvement.

Q8. Looking at the datapoints below, can we draw a decision boundary using Logistic Regression? Why? What are the specific issues or logistic regression with regards to XOR? [2 marks]

It is evident from the presented data points that we are unable to correctly classify all of the points in the XOR problem by applying Logistic Regression to construct a single decision boundary.
The above data points are non-linear, since logistic regression is a linear classifier, it is unable to process datasets that are not linearly separable.
In order to define suitable decision limits and capture the non-linear patterns in the data, more sophisticated models are required to solve the XOR problem, such as non-linear activation function neural networks are frequently used. Neural networks are more effective in situations like XOR because they can learn non-linear decision boundaries and are better suited to capturing complex patterns in the data.

Assignment 1 part 2b : **Neural Networks**

Q1. Why is it important to use a random set of initial weights rather than initializing all weights as zero in a Neural Network? [2 marks]

Here are many reasons why using a random set of initial weights is important:

(1) Neural networks use a learning algorithm called gradient descent, which adjusts weights based on the gradient of the loss function with respect to the weights. If all weights are initialized to zero, then during training, every neuron in the layer will learn the identical characteristics. This means that there won't be any variations in learning between the symmetric neurons with the identical weights and gradients.

(2) During training, symmetric neurons keep their symmetry, and during backpropagation, they receive updates in the same way. The network finds it more challenging to recognize complex patterns and representations when there is little variation among the neurons.

(3) In order for the network to adjust and specialize each neuron to capture distinct features of the input data, differential learning is essential.

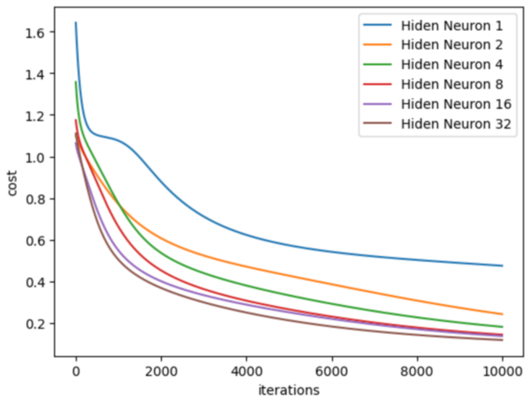
In conclusion, using a random set of starting weights facilitates differential learning by assisting in the disruption of neuronal symmetry and allowing the neural network to extract a variety of meaningful representations from the input data.

Q2. How does a NN solve the XOR problem? [1 marks]

The XOR problem can be resolved by a neural network (NN) by utilizing hidden layers and **non-linear** activation functions. Neurons in the hidden layers of a neural network are activated by non-linear activation functions, such as rectified linear unit (ReLU), hyperbolic tangent (tanh), or sigmoid. The network can recognize intricate relationships in the data thanks to the non-linear activation functions that introduce non-linearity. And the network can identify and represent non-linear patterns in the input data thanks to the hidden layers. Consequently, a neural network can be trained to represent complex relationships in the input data, such as the XOR pattern, by learning to create decision boundaries that linear models are unable to.

Q3. Explain the performance of the different networks on the training and test sets. How does it compare to the logistic regression example? Make sure that the data you are refering to is clearly presented and appropriately labeled in the report. [8 marks]

```
Hidden Neurons: 1, Accuracy on Test Data: 90.00%
Hidden Neurons: 2, Accuracy on Test Data: 100.00%
Hidden Neurons: 4, Accuracy on Test Data: 100.00%
Hidden Neurons: 8, Accuracy on Test Data: 100.00%
Hidden Neurons: 16, Accuracy on Test Data: 100.00%
Hidden Neurons: 32, Accuracy on Test Data: 100.00%
```



The **performance** of the **different networks** on the training and test sets is shown above.

And some other images is shown from “wk6_NN.ipynb” file. As the number of hidden neurons increases, the model becomes more capable of fitting the test data well, resulting in improved test accuracy. There seems to be an optimal spot when the number of hidden neurons is more than one, where the test accuracy is high, and the model is not overly complex, indicating that it can capture complex patterns in the data.

Comparison to Logistic Regression:

a. Neural networks with an appropriate number of hidden neurons seem to outperform logistic regression in terms of accuracy.
b. Neural networks are more adept at representing highly nonlinear relationships than the linear logistic regression model, which can be helpful when dealing with complex datasets.

In conclusion, given the right architecture, neural networks could outperform logistic regression in the given task. Selecting the appropriate number of hidden neurons is crucial for striking a balance between generalization and model complexity.