Unit 7: Introduction to Artificial Neural Networks (ANNs)
Unit 7 Artefact

e-Portfolio Activity: Perceptron Activities

I loaded each of the three programmes below into Google Colab:

simple_perceptron.ipynb
perceptron_AND_operator.ipynb
multi-layer Perceptron.ipynb

A perceptron is the fundamental unity of an artificial neural network and consists of an input layer and an output layer. A multi-layer perceptron is a class of feedforward artificial neural network that has at least three layers of node, the input, hidden layer/s and output layer (Kubat 2017). The connections between the layers can manipulated by changing the weights. Inputs and weights can be defined by the ML developer. In this exercise the weights were created as a Numpy array.

After the initial inputs and weights, the summation and step function need to be defined. step_function can be used as the activation function.

Training the perceptron then iteratively updates the weights and biases based on the prediction error using the learning rate. The prediction error is the difference between the actual target value (label) and the predicted value generated by the perceptron. This error indicates how far off the perceptron's prediction is from the true value.

Example programme:

```
import numpy as np

# Define the inputs (X) and the corresponding labels (y)
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y = np.array([0, 0, 0, 1])  # AND gate

# Initialize the weights (w) and bias (b)
weights = np.random.rand(2)
bias = np.random.rand(1)

# Define the step function (activation function)
def step_function(x):
    return 1 if x >= 0 else 0

# Summation function
def summation(inputs, weights, bias):
    return np.dot(inputs, weights) + bias

# Training the perceptron
def train_perceptron(X, y, weights, bias, learning_rate, epochs):
    for epoch in range(epochs):
```

```
        for inputs, label in zip(X, y):
            weighted_sum = summation(inputs, weights, bias)
            prediction = step_function(weighted_sum)
            # Update weights and bias
            weights += learning_rate * (label - prediction) * inputs
            bias += learning_rate * (label - prediction)
    return weights, bias

# Train the perceptron
learning_rate = 0.1
epochs = 10
weights, bias = train_perceptron(X, y, weights, bias, learning_rate, epochs)

# Predict function
def predict(inputs, weights, bias):
    weighted_sum = summation(inputs, weights, bias)
    return step_function(weighted_sum)

# Print results
for inputs in X:
    prediction = predict(inputs, weights, bias)
    print(f"Input: {inputs}, Predicted Output: {prediction}")
```

perceptron_AND_operator.ipynb

I now understand that the perceptron is a simple fundamental unit of an ANN used in binary classification. It can be used to simulate basic logical operations like AND, OR and NOT.

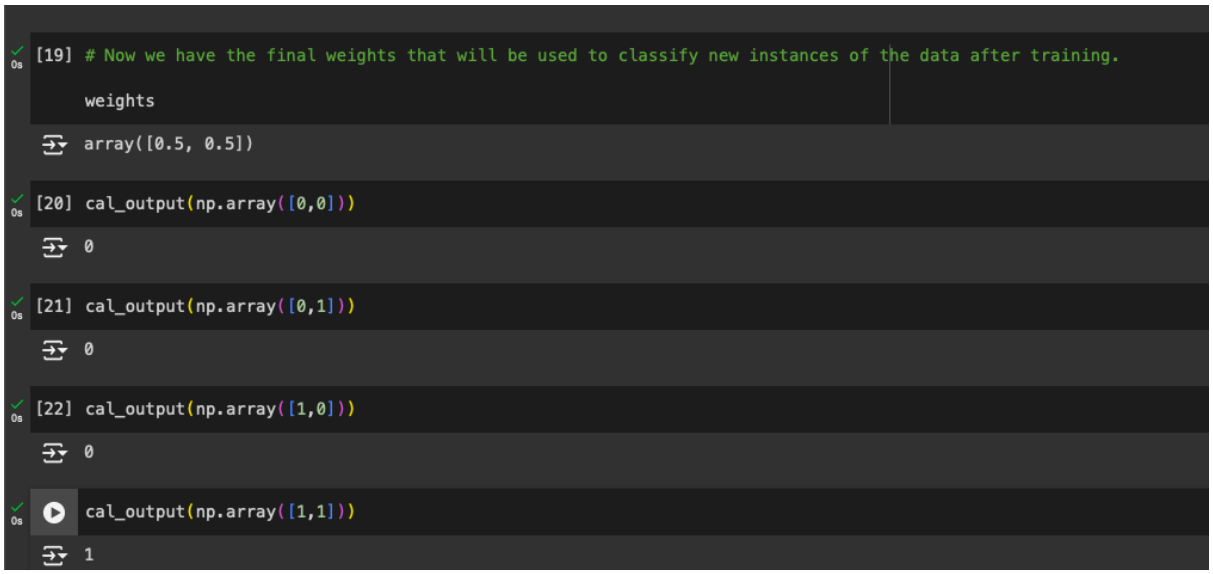
The AND operator returns ‘True’ only if both operands are ‘True’ (Kubat 2017).

```
0 AND 0 = 0
0 AND 1 = 0
1 AND 0 = 0
1 AND 1 = 1

array([[0, 0],
       [0, 1],
       [1, 0],
       [1, 1]])
```

In this exercise, I defined the inputs for the AND operator and the expected outputs. As before, the initial weights and learning rate was set. The activation function was the step_function. I then trained the perceptron and evaluated the output.

Final classification and output:



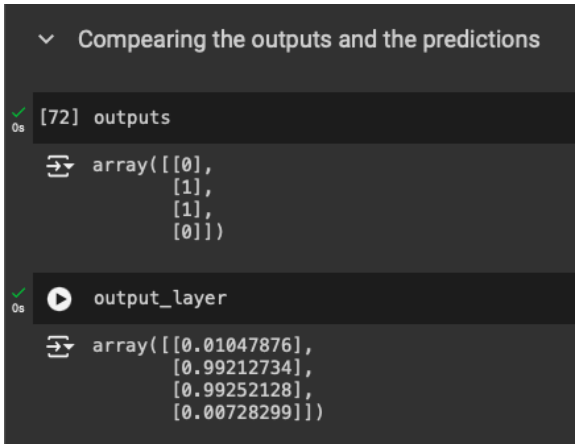
multi-layer Perceptron.ipynb

In this exercise I used the sigmoid function for the hidden and output layers. Sigmoid function is commonly used in neural networks especially in binary classification tasks because its output rang is between 0 and 1. The set up in terms of arrays for inputs and outputs was like previous exercises. In terms of the weights, I established initial weights between inputs and hidden layer, then another set of weights for the connection between the hidden layer and the output layer. Initialising them with random values was the approach taken in this exercise.

To run the completed ANN I had to debug a small typo: The variable weights1T should be weights_1.T in the calculation.

Epoch: 1 Error: 0.4999122229547342
Epoch: 100001 Error: 0.01705723177659601
Epoch: 200001 Error: 0.011840758693301923
Epoch: 300001 Error: 0.009597934423487369

Once I'd done this, the programme ran, and I could export plots showing the results from the ANN.



The ANN was able to output values close to the actual values.

Learning Outcomes

- Understand the applicability and challenges associated with different datasets for the use of machine learning algorithms.

Through these exercises involving perceptrons, the AND operator, and multi-layer perceptrons, I have demonstrated a comprehensive understanding of the applicability and challenges associated with different datasets in machine learning algorithms. In the first exercises, I implemented a simple perceptron and understood the foundational concept of how inputs and weights interact to produce outputs, revealing the basic mechanisms of machine learning models. Extending this to the AND operator showcased how specific logical functions could be modelled using perceptrons. This exercise allowed me to understand the importance of appropriate dataset structure and selection, and how this phase of the ML process can impact the model development and results generated. The multi-layer perceptron exercise revealed the complexity of more sophisticated models in ANNs, highlighting the challenges of training deeper models, such as the need for careful weight initialisation and the management of learning rates to ensure convergence and good modelling (Kubat 2017). Through these exercises and associated reading/learning, I encountered the real-world issues like overfitting, underfitting, and the necessity for data pre-processing techniques, illustrating the practical challenges of applying machine learning algorithms to diverse datasets (Kubat 2017). This hands-on experience has solidified my understanding of how dataset characteristics directly influence model performance and the importance of iterative tuning and validation in the machine learning process to obtain unbiased and applicable results (Kubat 2017). This is critical to manage the social, ethical and legal implications of ML.

References:

Kubat, M., 2017. An introduction to machine learning. Springer.