ARTIFICIAL NEURAL NETWORK

A report submitted in partial fulfillment of requirements for the degree of BE in Chemical Science and Engineering

Submitted By:

Jeevan Sapkota 0280119-20

DEPARTMENT OF CHEMICAL SCIENCE AND ENGINEERING
SCHOOL OF ENGINEERING
KATHMANDU UNIVERSITY

April, 2024

Contents

1 Introduction

Artificial Neural Networks (ANNs) are computational systems inspired by the human brain, imitating signal transmission in biological neurons. ANNs consist of layers of nodes, including an input layer, one or more hidden layers, and an output layer. These nodes, also known as perceptrons or artificial neurons, typically have one or more inputs, a bias, an activation function, and a single output. The perceptron processes inputs by multiplying them with corresponding weights and then passing them through an activation function to produce an output.

1. **Input:** The perceptron receives one or more numerical inputs. Each input is multiplied by a weight, reflecting its importance or strength in the final calculation.
2. **Weight:** Each input is assigned a weight, which determines its contribution to the perceptron's output.
3. **Bias:** A bias is an additional input to the perceptron with its own weight. It helps prevent the output from being zero when all inputs are zero, thus shifting the decision boundary.
4. **Activation Function:** The weighted sum of inputs and bias is passed through an activation function, which introduces nonlinearity and controls the firing of the perceptron. Common activation functions include the logistic (sigmoid) function, hyperbolic tangent function, and rectified linear unit.
5. **Output:** The activation function applied to the weighted sum produces the perceptron's output.

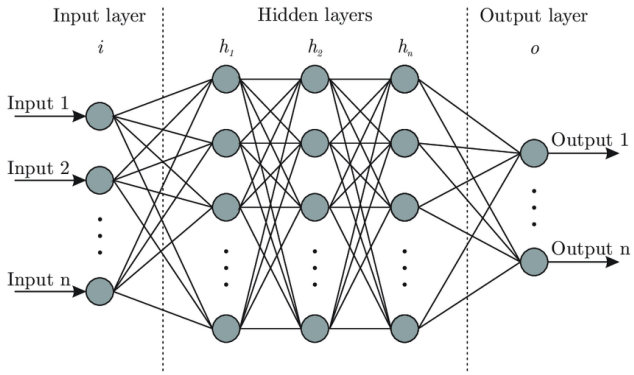

Figure 1: Artificial Network Network

A neural network consists of interconnected layers of perceptrons, typically including an input layer, one or more hidden layers, and an output layer. The input layer receives data from the external environment for analysis or learning. This data is then transformed through the hidden layers to provide valuable information for the output layer. Finally, the output layer generates a response, which can be numerical values or classifications, based on the input data.

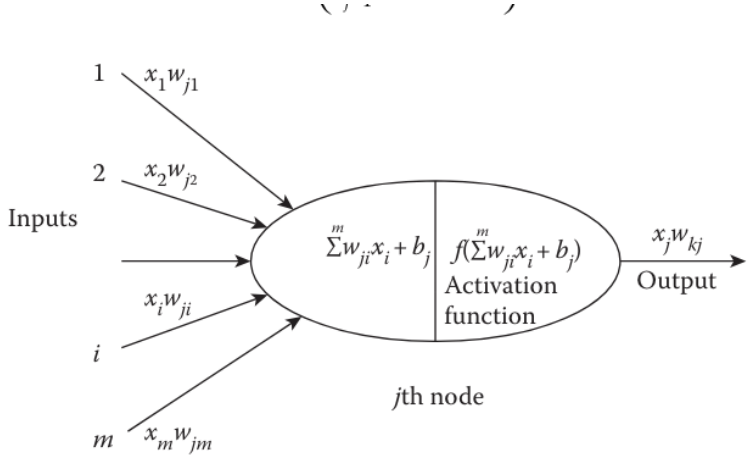

Figure 2: Information Processing in Single Neuron

2 Objectives

1. **Training ANN with Dataset:** Train an Artificial Neural Network (ANN) using a provided dataset, involving data preprocessing and model fitting.
2. **Performance Evaluation:** Evaluate the ANN's performance using standard metrics like confusion matrix and classification report to assess accuracy, precision, and recall.
3. **Hyperparameter Optimization:** Experiment with different hyperparameters (e.g., layers, neurons, activation functions) to improve the ANN's performance.

3 Methodology

1. **Python Libraries and Imports:** Necessary Python libraries such as Pandas for data manipulation and analysis, and scikit-learn (SKLEARN) for machine learning, were imported. Functions from scikit-learn for data standardization (StandardScaler), data splitting into train and test sets (train_test_split), defining the neural network model (MLPClassifier), and model evaluation (classification_report) were imported.
2. **Data Loading and Visualization:** The csv data file labeled "wine.data" was read, and the data was visualized in Python. Statistical values of the wine data were observed. The feature "Cultivator" was identified as the target variable to be predicted by the neural network model.
3. **Data Splitting:** The available data was split into training and testing sets using the train_test_split function from the sklearn.model_selection library.
4. **Data Normalization:** Before training the model, the data were normalized using the StandardScaler function from the sklearn.preprocessing library. This step scales the data to fit the neural network model.
5. **Neural Network Model Creation:** The neural network model was created using the MLPClassifier function imported from the sklearn.neural_network library. The function was provided with the size and number of hidden layers, i.e., hidden_layer_sizes=(13, 13, 13), and

the maximum number of iterations taken by the algorithm, i.e., max_iter=500.

6. **Model Training:** The neural network model was fitted with the X_TRAIN and Y_TRAIN values to adjust the bias and weight matrix of the network.
7. **Model Evaluation:** Finally, the model was given the X_TEST data to predict the output (in this case, classify the wine bottle). The evaluation of the trained neural network model was done using built-in functions from sklearn.metrics such as confusion_matrix and classification_report.

4 Results and Discussion

In the following section,we include the evaluation of the trained neural network model using the confusion matrix and classification report.

[1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report,confusion_matrix

#Data Loading and Exploaration
wine = pd.read_csv('wine.data',names=["Cultivator","Alcohol","Malic␣
 ↪acid","Ash","Alcalinity of ash", "Magnesium","Total␣
 ↪phenols","Flavanoids","Nonflavanoid phenols","Proanthocyanins","Color␣
 ↪intensity","Hue","OD280/OD315 of diluted wines","Proline"])
wine.head()
wine.describe().transpose()

wine.shape[0]

#Data Preparation
X = wine.drop("Cultivator",axis=1)

y= wine["Cultivator"]
y.head()
y.describe().transpose()

num_iterations = 300

#Looping the process to make confusion matrix simpler
for i in range(num_iterations):
    X_train, X_test, y_train, y_test = train_test_split(X,y)

    #Data Preprocessing
    scaler= StandardScaler()
    scaler.fit(X_train)
```

```
    X_train= scaler.transform(X_train)
    X_test= scaler.transform(X_test)

    #Model Building
    mlp= MLPClassifier(hidden_layer_sizes=(13,13,13),max_iter=500)
    mlp.fit(X_train,y_train)
    predictions= mlp.predict(X_test)

#Evaluation
print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))
```

```
[[12  0  0]
 [ 0 22  0]
 [ 0  0 11]]
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        12
           2       1.00      1.00      1.00        22
           3       1.00      1.00      1.00        11

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

The resultant confusion matrix for the model can be seen from the output above. From the matrix, it is shown that the neural network was able to correctly classify each bottle of wine to its respective cultivator. This conclusion is generated as there are no "non-zero" values in the off-diagonal positions of the matrix. The diagonal elements represent that 12 sample bottle wines predicted were correctly classified from Cultivator 1. Similarly, 22 and 11 sample bottle wines predicted from the model were correctly classified from Cultivator 2 and 3, respectively.

The classification report obtained for the three cultivators is displayed above. The report consists of model metrics such as precision, recall, and F1-score. Precision is the ratio between true positives to predicted positives by the model, while recall is the ratio of true positives by the model to actual positives in the datasheet, and F1-score is the harmonic mean of precision and recall. Since our model was able to correctly classify each cultivator, all three parameters have the value of 1. With no class imbalances, both the macro average and weighted average were 1, indicating excellent performance by the model.

5 Conclusion

In this lab report, we successfully trained an Artificial Neural Network (ANN) model to classify wine bottles into their respective cultivators using the provided dataset. Through the evaluation of the model using the confusion matrix and classification report, we observed that the neural network demonstrated excellent performance, correctly classifying each bottle of wine to its cultivator with high precision, recall, and F1-score. The absence of non-zero values in the off-diagonal positions of

the confusion matrix indicates the robustness of the model in making accurate predictions. These results suggest that the ANN model can be effectively utilized for wine cultivator classification tasks. Further experimentation and optimization of hyperparameters could potentially enhance the model's performance even further.