**Assignment No: 1**

# **Feedforward Neural Network**

## **Problem Statement**

Implementing Feedforward Neural Networks in Python using Keras and TensorFlow.

## **Objective**

* To understand the basic structure of feedforward neural networks.
* To learn how to preprocess data for training neural networks.
* To implement a feedforward neural network model using Keras and TensorFlow.
* To evaluate model performance using validation data.
* To visualize training loss and validation loss over epochs.

## **Software and Hardware Requirements**

* **Operating System:** Windows/Linux/MacOS
* **Kernel:** Python 3.x
* **Tools:** Jupyter Notebook, Anaconda, or Google Colab
* **Hardware:** CPU with minimum 4GB RAM; optional GPU for faster training
* **Libraries and Packages:** TensorFlow, Keras, NumPy, Pandas, Matplotlib, Scikit-Learn

## **Theory**

### **Definition**

A Feedforward Neural Network is a type of artificial neural network where connections between the nodes do not form cycles. The information moves in only one direction—forward—from the input nodes, through the hidden nodes (if any), and to the output nodes.

### **Structure**

* **Input Layer:** Receives the input features.
* **Hidden Layers:** One or more layers where computation occurs. Each neuron in a layer is connected to every neuron in the next layer.
* **Output Layer:** Produces the output of the network.

### **Activation Functions**

Functions like ReLU (Rectified Linear Unit), Sigmoid, and SoftMax are used to introduce non-linearity into the model.

### **Backpropagation**

A key algorithm used for training the network, where the error is propagated backward through the network to update weights.

## **Methodology**

1. **Data Acquisition:**
   * Load the dataset (winequality-red.csv) using Pandas to analyze the chemical properties of red wine and their quality.
2. **Data Preparation:**
   * Split the dataset into training (75%) and validation (25%) sets.
   * Normalize the feature values to a range between 0 and 1 to facilitate faster convergence during training.
3. **Model Architecture:**
   * Create a sequential model using Keras.
   * Add multiple dense layers:  
     + Input layer with 64 units and ReLU activation function.
     + Hidden layer with 64 units and ReLU activation function.
     + Output layer with a single unit for regression.
4. **Model Compilation:**
   * Compile the model using the Adam optimizer and Mean Absolute Error (MAE) as the loss function.
5. **Model Training:**
   * Fit the model to the training data while validating on the validation set.
   * Track the loss metrics over a specified number of epochs.
6. **Model Evaluation:**
   * Use the trained model to predict the quality of wine based on the validation dataset.
   * Compare the predicted values with actual values for assessment.
7. **Loss Visualization:**
   * Plot the training and validation loss over epochs to visualize the model’s performance and check for overfitting.

## **Advantages**

* **Non-linearity Handling:** Feedforward neural networks use activation functions (like ReLU, sigmoid, or tanh) that introduce non-linearities, allowing them to learn complex relationships in data that linear models cannot capture.
* **Flexibility in Architecture:** These networks can be easily modified to suit various tasks by adjusting the number of layers, neurons, and types of activation functions.
* **Scalability:** Feedforward neural networks can scale well with the addition of more hidden layers and neurons.
* **Robustness:** When properly trained, they can generalize well to unseen data.
* **Parallel Processing:** Their structure supports parallel computation on GPUs, speeding up training.

## **Limitations**

* **Data Requirements:** Large amounts of labelled data are required for effective training.
* **Computational Cost:** Training deep networks can be expensive and resource-intensive.
* **Black-Box Nature:** Interpretability is limited, making them less suitable in domains requiring explainability.
* **Overfitting Risk:** Complex networks may overfit the training data.
* **Hyperparameter Sensitivity:** Performance is sensitive to hyperparameter choices, requiring careful tuning.

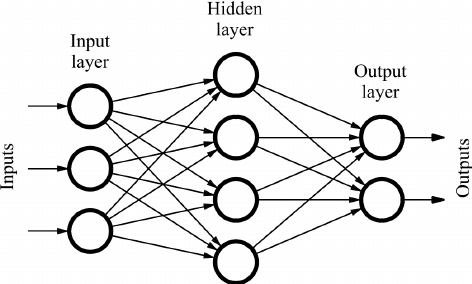
## **Applications**

* Predictive modeling in various fields such as healthcare, finance, and manufacturing
* Classification and regression tasks
* Real-time data analytics
* Image and signal processing

## **Working / Algorithm**

1. **Import Libraries:** Use libraries such as NumPy, Pandas, and TensorFlow.
2. **Load Dataset:** Read winequality-red.csv into a Pandas DataFrame.
3. **Split Dataset:** Allocate 75% for training and 25% for validation.
4. **Normalize Data:** Use Min-Max normalization to scale feature values to the (0, 1) range.
5. **Separate Features and Targets:** Define X\_train, X\_val, y\_train, y\_val.
6. **Define Input Shape:** Based on the number of features in X\_train.
7. **Build Model:** Use Keras to create a sequential model:
   * Dense(64, activation='relu')
   * Dense(64, activation='relu')
   * Dense(1)
8. **Compile Model:** Use Adam optimizer and MAE loss function.
9. **Train Model:** Fit the model on training data and validate using the validation set.
10. **Evaluate Model:** Make predictions and compare them to actual values.
11. **Visualize Training Loss:** Plot training vs. validation loss to assess model performance.
12. **Deploy Model:** Ready for inference on new/unseen data.

## **Diagram**



## **Conclusion**

In conclusion, the Feedforward Neural Network (FNN) algorithm is a powerful and versatile approach to predictive modeling. It is well-suited for both classification and regression tasks, such as predicting wine quality based on various chemical features. By learning complex, non-linear relationships between inputs and outputs, FNNs can offer valuable insights and accurate predictions. However, their performance depends on adequate data, computing resources, and careful tuning. Despite these challenges, FNNs are among the most effective models for complex data-driven tasks.