Multilayer Perceptron Networks

CT-466 | Week 3 - Lecture 5

Instructor: Mehar Fatima Shaikh

Limitation of Single-Layer Perceptron

- A single-layer perceptron can only classify linearly separable problems.
- Example: AND, OR gates → linearly separable.
- But XOR (exclusive OR) and many real-world problems are non-linearly separable.
- This means single-layer perceptrons cannot capture complex patterns.

Role of Hidden Layers

- Adding hidden layers allows the network to combine multiple linear decision boundaries to form non-linear boundaries.
- Hidden neurons transform the input space into a higher-dimensional space where separation is possible.
- This idea is similar to the kernel trick in SVM.

Handling Complex Real-World Data

- Real data (images, speech, text, medical signals, etc.) is rarely linearly separable.
- Multilayer networks (MLPs, CNNs, RNNs, etc.) are needed to learn:
  - Non-linear mappings
  - Hierarchical feature representations
  - Temporal/spatial dependencies

Universal Approximation Theorem

- States that a multilayer feedforward neural network with at least one hidden layer and non-linear activation functions can approximate any continuous function to any desired accuracy.
- This makes multilayer networks extremely powerful and flexible.

Examples

- XOR Problem: Requires at least 2 hidden neurons to solve.
- Image Classification: Needs many hidden layers (CNNs).
- Speech Recognition: Needs recurrent or deep networks to capture sequences.

Multi Layer Perceptron Learning Algorithm

- A multi-layer perceptron (MLP) is a type of feedforward neural network composed of multiple layers of neurons.
- It consists of at least three layers: an input layer, one or more hidden layers, and an output layer.
- In an MLP, every neuron in one layer is fully connected to all neurons in the following layer.

Multi Layer Perceptron Learning Algorithm

- The input layer is the visible layer that simply passes the input data to the next layer.
- The layers that come after the input layer are called hidden layers.
- Hidden layers do not directly interact with the external environment, they neither receive raw inputs nor provide final outputs.
- The last layer is the output layer, which produces the final result as either a single value or a vector of values.

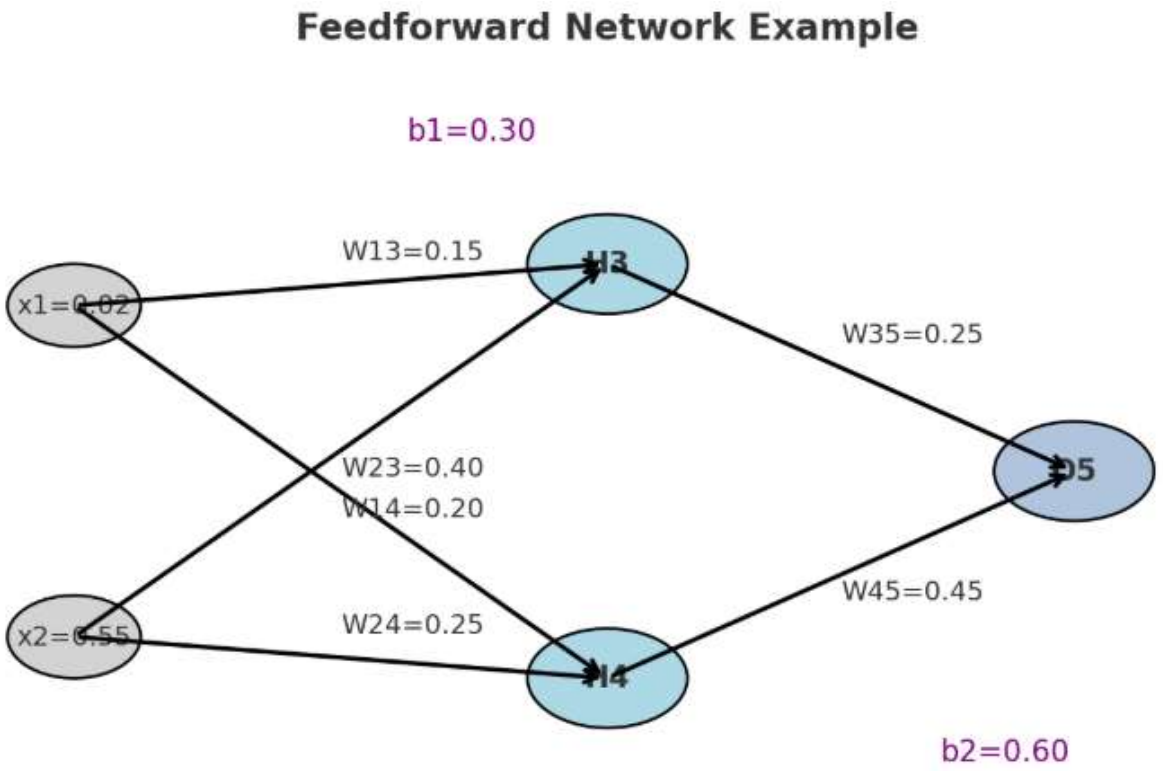
Feedforward Neural Network (FNN)

- A Feedforward Neural Network (FNN) is the simplest type of artificial neural network architecture.
- A feedforward network is a network of neurons where information flows strictly in one direction: from input layer → through hidden layers → to output layer.
- There are no loops or cycles in the connections.

Structure

- **Input Layer**
  - Takes in raw data (features).
  - Does not perform computation, just passes values forward.
- **Hidden Layers** (can be one or many)
  - Each neuron computes a weighted sum of inputs, adds bias, and applies an activation function (like ReLU, sigmoid, tanh).
  - Extracts patterns and representations.
- **Output Layer**
  - Produces the final result (classification label, probability, or regression value).

Multi Layer Feed forward Network



Solution

- **Given**
- x1 = 0.02, x2 = 0.55
- w13 = 0.15, w23 = 0.4, w14 = 0.2, w24 = 0.25
- Hidden-layer bias b1 = 0.3
- w35 = 0.25, w45 = 0.45, Output bias b2 = 0.6

- **Step 1: Hidden Layer Nets**
- net3 = x1·w13 + x2·w23 + b1 = 0.02*0.15 + 0.55*0.4 + 0.3 = 0.523000
- net4 = x1·w14 + x2·w24 + b1 = 0.02*0.2 + 0.55*0.25 + 0.3 = 0.441500

- **Step 2: Hidden Layer Activations (Sigmoid)**
- h3 = σ(net3) = σ(0.523000) = 0.627849
- h4 = σ(net4) = σ(0.441500) = 0.608616

- **Step 3: Output Net**
- net5 = h3·w35 + h4·w45 + b2 = 0.627849*0.25 + 0.608616*0.45 + 0.6 = 1.030840

- **Step 4: Output Activation (Sigmoid)**
- o5 = σ(net5) = σ(1.030840) = 0.737079

- **Results**
- H3 = 0.627849, H4 = 0.608616, O5 = 0.737079

Multilayer feed-forward neural network

- A multilayer feed-forward neural network (or multi-layer perceptron, MLP) is a type of artificial neural network where neurons are arranged in layers, and information flows in one direction: from input to output. Here's a breakdown of its key components and how it operates:
- Components:

1. Input Layer: Neurons in this layer receive the input features of the data. Each neuron corresponds to one feature of the input data.
2. Hidden Layers: These layers sit between the input and output layers. Each hidden layer consists of neurons that perform calculations based on the inputs received from the previous layer. Multiple hidden layers allow the network to learn complex relationships in the data.
3. Output Layer: Neurons in this layer compute the final output of the network. The number of neurons in the output layer depends on the type of problem being solved (e.g., regression, classification).
4. Connections (Weights): Each neuron in one layer is connected to every neuron in the next layer. These connections have associated weights that determine the strength and direction (excitatory or inhibitory) of the signal between neurons.
5. Activation Function: Each neuron (except those in the input layer) typically applies an activation function to the weighted sum of its inputs. This introduces non-linearity into the model, enabling it to learn complex patterns in the data.
6. Bias: Each neuron may also have an associated bias term, which allows the activation function to shift left or right, adding flexibility to the model.

Feedback Neural Network (also called Recurrent Neural Network (RNN))

- A Feedback Neural Network is a type of artificial neural network in which connections between nodes form a directed cycle, allowing outputs to be fed back into the network as inputs. This feedback loop gives the network a kind of memory, enabling it to process sequences of data and retain past information.

Multi-layer perceptron

- Operation:
  - Forward Propagation:
    - During the forward pass, input data is fed into the network through the input layer.
    - The data is multiplied by the weights and passed through the activation function of each neuron in the hidden layers and finally in the output layer.
    - The output layer produces a prediction or classification result.
  - Backward Propagation (Training):
    - During training, the network adjusts its weights based on the error between the predicted output and the actual output (often using a loss function like mean squared error or cross-entropy).
    - Backpropagation calculates the gradient of the loss function with respect to each weight in the network, enabling weight updates that minimize the error.
    - This iterative process continues over multiple epochs (passes through the entire dataset) until the model converges to a satisfactory level of accuracy.

Multi-layer perceptron

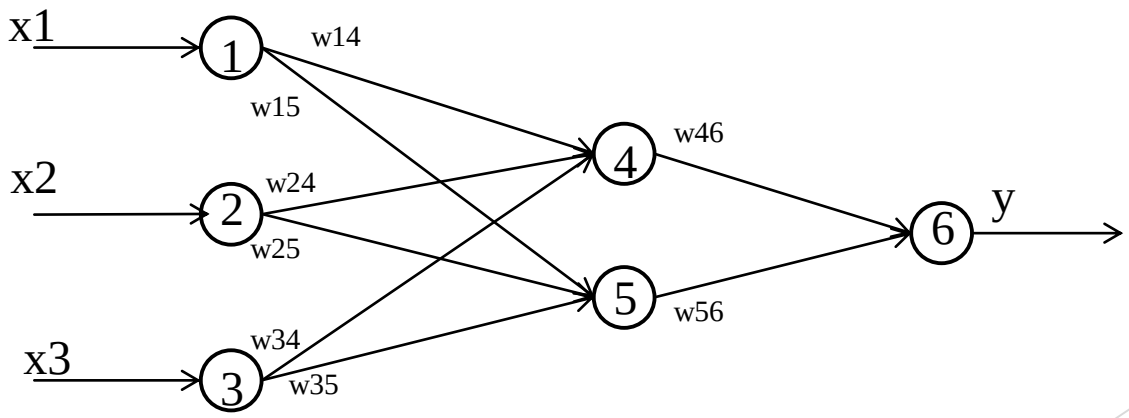
- Advantages:
  - Universal Approximator: A properly sized MLP with nonlinear activation functions can approximate any continuous function.
  - Versatility: Effective for both regression and classification tasks.
  - Scalability: Can handle large datasets and complex patterns.
- Considerations:
  - Overfitting: Can occur with large, complex networks if not regularized properly.
  - Training Time: Requires sufficient data and computational resources for training, especially with deeper architectures.
- In summary, a multilayer feedforward neural network is a foundational architecture in deep learning, capable of learning complex mappings between inputs and outputs through iterative training and adjustment of its internal weights.

Problem:

- Given:
- Inputs: x1=1, x2=0, x3=1
- Weights:

w14=0.2	w34=−0.5
w15=−0.3	w35=0.2
w24=0.4	w46=−0.3
w25=0.1	w56=−0.2

- Thresholds (biases):
  - θ4 = −0.4
  - θ5 = 0.2
  - θ6 = 0.1
- Target output: t = 1



Solution:

- Calculate activations in the hidden layer (Layer 2):
- Neuron z4(Hidden layer):
  - z4=
- Neuron z5 (Hidden layer):
- Apply the sigmoid activation function to z4,z5:

Calculate activations in the output layer (Layer 3) :

- Neuron z6 (Output layer):

- Apply the sigmoid activation function to y6
- $\hat{Y}6 = \frac{1}{1+e^{-z6}} = \frac{1}{1+e^{-(0.00462)}} = 0.48820$

Calculate the error at each node

Calculate the error at each node

Update weights

Update weights

Update weights

- **Output Layer Weights Update**
  - For the weights w46 and w56 connected to the output neuron z6:
- **Update w46:**
  - $\Delta w46 = -\eta \cdot \delta 6 \cdot y4 = 0.9 \cdot 0.1248 \cdot \approx -0.0373$
  - $w46' = w46 + \Delta w46 = -0.3 - 0.0373 = -0.3373$
- **Update w56:**
  - $\Delta w56 = -\eta \cdot \delta 6 \cdot y5 = -0.9 \cdot 0.1248 \cdot 0.525 \approx -0.0590$
  - $w56' = w56 + \Delta w56 = -0.2 - 0.0590 \approx -0.2590$

Hidden Layer Weights Update

Problem

- You are training a multi layer perceptron neural network for a particular classification task. After some investigation, your neural network is constructed with 5 inputs variables, one hidden layer with 12 nodes and one output layer with 3 nodes(the classes). How many network parameters are required to be trained? show detailed steps

Solution :

- **Input Layer Parameters:**
  - **You have 5 input variables.**
  - **Each input variable connects to every node in the hidden layer.**
  - **So, there are 5 connections (weights) per node in the hidden layer.**
  - **Total input layer parameters: 5×12=60**
  - **.**
- **Hidden Layer Parameters:**
  - **You have 12 nodes in the hidden layer.**
  - **Each node connects to every node in the output layer.**
  - **So, there are 12 connections (weights) per node in the hidden layer.**
  - **Total hidden layer parameters: 12×3=36**
- **Total Network Parameters:**
  - **Summing up the parameters from all layers:**
  - **Total Parameters=Input Layer Parameters+Hidden Layer Parameters+Output Layer Parameters**
  - **= 60 + 36 + 3 = 99**
- **Therefore, your MLP neural network requires 99 network parameters to be trained.**