**Non-programming Assignment:**

1. **Why are multilayer (deep) neural networks needed?**
   * Multilayer neural networks, or deep neural networks (DNNs), are needed because they allow the model to learn complex and hierarchical patterns in the data. Each layer extracts increasingly abstract features from the input, enabling the network to understand intricate patterns and relationships that a single-layer network (shallow network) would not be able to capture. In other words, multiple layers allow the neural network to represent data more comprehensively.
2. **What is the structure of the weight matrix (how many rows and columns)?**
   * The weight matrix W[s] between layer −1 and layer s in a neural network has dimensions Ns×Ns−1​, where:
     + Ns is the number of neurons in the receiving layer (layer s),
     + Ns−1​ is the number of neurons in the sending layer (layer s−1).
3. **Describe the gradient descent method.**
   * Gradient descent is an optimization method used to minimize a function, such as a loss function in a neural network, by iteratively moving towards the direction of steepest descent (negative gradient). In the context of neural networks, the parameters (weights and biases) are updated after each iteration: W[s]=W[s]−α∂J/∂W[s] ​ where **α** is the learning rate and J is the cost (loss) function.
4. **Describe in detail forward propagation and backpropagation for deep neural networks.**
   * **Forward propagation:**
     + Forward propagation is the process of computing the output of the network for a given input by passing data from the input layer through each hidden layer to the output layer.
     + For each layer s:

Z[s]=W[s]A[s−1] + b[s]   
A[s]=f(Z[s])

where W[s] is the weight matrix, b[s] is the bias vector, A[s−1] is the output (activation) of the previous layer, and f is the activation function.

* + **Backpropagation:**
    - Backpropagation is the process of calculating the gradient of the loss function with respect to each parameter in the network, starting from the output layer and propagating backward through the layers.
    - For the output layer: δ[L]=A[L] - Y
    - For hidden layers: δ[s]= (W[s+1])Tδ[s+1]⋅f′(Z[s])
    - The weight and bias updates are then:

W[s]=W[s]−α.(1/m).δ[s](A[s−1])T

b[s]=b[s]−α.1/m.∑δ[s]

where **δ[s]** is the error term for layer s, and **α** is the learning rate.

1. **Describe linear, ReLU, sigmoid, tanh, and softmax activation functions and explain for what purposes and where they are typically used.**
   * **Linear activation function**:
     + f(z)=z
     + This function is used in the output layer of regression models where the output is a continuous value.
   * **ReLU (Rectified Linear Unit)**:
     + f(z)=max(0,z)
     + ReLU is widely used in hidden layers because it introduces non-linearity while being computationally efficient. It also helps mitigate the vanishing gradient problem.
   * **Sigmoid activation function**:
     + f(z)=1/1+e−z
     + Sigmoid is typically used in the output layer for binary classification problems, where the output is between 0 and 1.
   * **Tanh activation function**:
     + f(z)= (ez - e-z )/(ez + e-z )​
     + Tanh is often used in hidden layers of neural networks, especially in recurrent neural networks, as it outputs values in the range [-1, 1] and provides stronger gradients compared to the sigmoid function.
   * **Softmax activation function**:
     + f(zi)=ezi∑j=1Nezj
     + Softmax is used in the output layer for multiclass classification problems. It converts logits into probabilities, with the sum of the probabilities equal to 1.