Deep Learning Assignment-1

1) The function of a summation junction of a neuron is to bind the weights and inputs together and calculate their sum. The threshold activation function is to achieve a non-linearity in the dataset. And also to make sure that by neuron is activated fine.
2) A step function is an activation function which has an amplitude value that ranges from 0 to maximum value. The difference between step function and threshold function is that step function is a simple kind of threshold function for some input values if y/output is greater than threshold value , neuron will activate otherwise not.
3) McCulloch–Pitts model of neuron:: In 1943 , this model was developed by two electrical engineers. They have binary devices. Each neuron has a fixed threshold theta. The neuron receives inputs from excitory synapses, which has same weight value. Inhibitory inputs have an absolute veto power over any excitory inputs. At each time step the neurons are simultaneously updated by summing the weighted excitatory inputs and setting the output to 1 if the sum is greater than or equal the threshold AND if the neuron receives no inhibitory input.
4) ADALINE network model:: A network with single perceptron of a linear activation function is a ADALINE network model. In this model, input layer is directly connected to output layer. Weights between input and output layer are adjustable. It follows delta rule. The learning rule is found to minimize the mean square error between activation and target values. Adaline consists of trainable weights, it compares actual output with calculated output, and based on error training algorithm is applied.
5) Simple perceptron:: In simple peeceptron, there is only one neuron, one biased value, and output layer(binary classification). It uses training algorithm/ back propagation to update weight and bias value to get acceptable output value.
   But it fails with real world dataset . Real world dataset consists of nos. of targeted values, lots of outliers, non-linearity of dataset. With simple perceptron, we can't train non-linear complex dataset with linear activation function.

6) Linearly inseparable problems are that decision problems are not solved by single hyperplane / linear decision boundary. Hidden layer is the layer between input layer and output layer, Where neural network takes output of weights and biased values with activation function. Hidden Layer also learns the complex pattern in data by adjusting weights of neurons.
7) XOR problem in simple perceptron :: XOR is a linear inseparable problem, that can't be distinguished by a decision boundary. We can't apply simple perceptron for XOR problem.
8) XOR problem in MLP:: XOR problem can be solved by MLP. For this, 1 hidden layer should be taken. Input layer should have 2 neurons. And one output layer. Tanh and sigmoid activation functions are being used by hidden layers and output layers. Initialize the weights and biased values with some random values.
9) Single-layer feed forward architecture of ANN:: It starts from getting input values from outside, pass them to next hidden layer by initializing weights and biased values. To activate the neural network, we use activation function according to the types of tasks.
10) A Competitive Neural Network, or Competitive Learning Network, is an artificial neural network architecture for unsupervised learning. It features a layer of neurons where each neuron represents a possible category or cluster. When presented with an input, the neuron whose weight vector is most similar to the input "wins" the competition and is considered the winner. The winner represents the best match for the input data. The network's weight vectors are adjusted to become more similar to the input data, allowing the network to adapt and self-organize. Competitive networks are used for tasks such as data clustering, dimensionality reduction, and vector quantization, and the Kohonen Self-Organizing Map (SOM) is a well-known example of a competitive network.
11) Back propagation steps:

    The backpropagation algorithm for training multi-layer feedforward neural networks involves the following steps:

    1. **Initialization**: Set initial weights and biases.
    2. **Forward Pass**: Compute the network's output for a given input.

3. **Calculate Error**: Measure the error between predicted and actual output.
4. **Backward Pass**: Propagate error gradients backward through the network.
5. **Weight and Bias Updates**: Adjust weights and biases to reduce error.
6. **Repeat for Multiple Epochs**: Iteratively update the network through multiple passes of the dataset.
7. **Termination**: Stop training based on specified criteria (e.g., number of epochs, error threshold).
8. **Validation and Testing**: Assess the model's performance on validation and test data.

12) Advantages of Neural Networks:

1. Powerful function approximators.
2. Adaptability and generalization.
3. Feature learning.
4. Parallel processing.
5. Scalability.
6. Availability of frameworks.

Disadvantages of Neural Networks:

1. Complexity and hyperparameter tuning.
2. Data-hungry nature.
3. Overfitting.
4. Computational intensity.
5. Lack of interpretability.
6. Hyperparameter sensitivity.
7. Data quality and potential biases.
8. Security vulnerabilities (adversarial attacks).

13) RELU:: The Rectified Linear Unit (ReLU) function is a popular activation function used in artificial neural networks. It is defined as follows: f(x) ={x, if $x > 0$ and 0, if $x < 0$

In other words, if the input to the ReLU function is positive, it returns the input value unchanged; if the input is non-positive (zero or negative), it returns zero. The ReLU function is known for its simplicity and effectiveness, and it offers several advantages:

Advantages of the ReLU Function:

Simplicity: The ReLU function is simple to implement and computationally efficient. It involves only a basic thresholding operation.

Mitigates Vanishing Gradient Problem: ReLU activations do not saturate for positive values, which helps mitigate the vanishing gradient problem during training. This allows for more effective training of deep networks.

Sparsity: ReLU activations often result in sparse representations, where many neurons remain inactive (output zero), which can lead to more efficient and expressive network representations.

Biological Inspiration: The ReLU activation function is inspired by the way neurons in the human brain operate, firing when the input signal surpasses a certain threshold.

Despite its advantages, the ReLU function has a drawback known as the "dying ReLU" problem, where some neurons may get stuck in an inactive state during training and never recover. This occurs when the input to a ReLU neuron is always non-positive, and the gradient remains zero, preventing weight updates. To address this issue, variations of ReLU have been introduced, such as Leaky ReLU, Parametric ReLU, and Exponential Linear Unit (ELU), which allow a small gradient for negative inputs to prevent neurons from becoming completely inactive.

Gradient Descent::

Gradient Descent is an optimization algorithm used to minimize a loss function in machine learning. It adjusts model parameters iteratively by computing the gradient of the loss with respect to the parameters and updating them in the direction that reduces the loss. It's essential for training models but requires careful tuning of parameters like the learning rate. Variants include Stochastic Gradient Descent, Mini-Batch Gradient Descent, and Batch Gradient Descent.

Gradient Descent is an optimization algorithm used to minimize a loss function in machine learning and deep learning. It is a fundamental technique for training neural networks and many other optimization problems. Here's a brief overview of how it works:

1.Objective Function: Gradient Descent is used to minimize a specific objective function, often called the "loss function" or "cost function." This function measures the difference between the predicted and actual values of a model.

2. Initialization: Start by initializing the model's parameters (weights and biases) with some initial values. These parameters are the variables that the optimization process seeks to adjust to minimize the loss.

3. Gradient Calculation: For a given set of parameters, calculate the gradient of the loss function with respect to each parameter. The gradient points in the direction of the steepest increase in the loss function.

4. Update Parameters: Adjust the parameters in the opposite direction of the gradient to reduce the loss. This adjustment is made using the formula:

New Parameter = Old Parameter - (Learning Rate * Gradient)

The learning rate is a hyperparameter that controls the step size of the update.

5. Iteration: Repeat steps 3 and 4 for a predefined number of iterations (epochs) or until convergence is achieved. During each iteration, the model's parameters are updated to minimize the loss function.

6. Convergence: Convergence is reached when the loss function reaches a minimum (or a plateau where it no longer decreases significantly), or when a stopping criterion is met, such as a maximum number of iterations or a target loss value.