**M.RASIK AHAMED**

**122012012874**

**ASSIGNMENT -1**

**Activation functions used in Artificial Neural Networks (ANN):**

1. Sigmoid Activation Function

- Mathematical Representation: σ(x) = 1 / (1 + e^(-x))

- Range: (0, 1)

- Purpose: Used for binary classification problems

- Pros: Maps inputs to a probability value

- Cons: Suffers from the vanishing gradient problem, not computationally efficient

2. ReLU (Rectified Linear Unit) Activation Function

- Mathematical Representation: f(x) = max(0, x)

- Range: [0, ∞)

- Purpose: Used for hidden layers in deep neural networks

- Pros: Computationally efficient, easy to compute, helps avoid vanishing gradient problem

- Cons: Not differentiable at x=0, can result in dying neurons

3. Tanh (Hyperbolic Tangent) Activation Function

- Mathematical Representation: tanh(x) = 2 / (1 + e^(-2x)) - 1

- Range: (-1, 1)

- Purpose: Used for hidden layers in deep neural networks

- Pros: Maps inputs to a value between -1 and 1, helps avoid vanishing gradient problem

- Cons: Can result in vanishing gradients, not as computationally efficient as ReLU

**Types of layers in an Artificial Neural Network (ANN):**

1. Input Layer

- Purpose: Receives the input data

- Number of neurons: Equal to the number of features in the input data

- Activation function: None

2. Hidden Layers

- Purpose: Performs complex representations and transformations of the input data

- Number of neurons: Can vary, but typically between 10-1000

- Activation function: ReLU, Tanh, Sigmoid, etc.

3. Output Layer

- Purpose: Produces the final output of the network

- Number of neurons: Equal to the number of classes or outputs

- Activation function: Sigmoid (binary classification), Softmax (multi-class classification), Linear (regression)

4. Convolutional Layers (CNNs)

- Purpose: Extracts features from images and other spatial data

- Number of neurons: Can vary, but typically between 10-1000

- Activation function: ReLU, Tanh, Sigmoid, etc.

**Types of optimizers used in Artificial Neural Networks (ANN):**

1. Stochastic Gradient Descent (SGD)

- Updates the weights based on the gradient of the loss function for a single example

- Simple to implement, but can converge slowly

2. Mini-Batch Gradient Descent (MBGD)

- Updates the weights based on the gradient of the loss function for a mini-batch of examples

- Balances the trade-off between SGD and batch gradient descent

3. Momentum Optimizer

- Adds a fraction of the previous update to the current update to escape local minima

- Can help converge faster, but requires careful tuning of hyperparameters

4. Nesterov Accelerated Gradient (NAG) Optimizer

- Modifies the momentum optimizer to use the future gradient to update the weights

- Can converge faster than the momentum optimizer

5. Adagrad Optimizer

- Adapts the learning rate for each parameter based on the gradient

- Can help converge faster, but may not perform well for sparse data

**Types of loss functions used in Artificial Neural Networks (ANNs):**

1. Mean Squared Error (MSE) Loss

- Used for regression problems

- Measures the average squared difference between predicted and actual values

- Formula: MSE = (1/n) \* ∑(y\_true - y\_pred)^2

2. Mean Absolute Error (MAE) Loss

- Used for regression problems

- Measures the average absolute difference between predicted and actual values

- Formula: MAE = (1/n) \* ∑|y\_true - y\_pred|

3. Cross-Entropy Loss (Binary Classification)

- Used for binary classification problems

- Measures the difference between predicted probabilities and actual labels

- Formula: CE = -(y\_true \* log(y\_pred) + (1-y\_true) \* log(1-y\_pred))

4. Categorical Cross-Entropy Loss (Multi-Class Classification)

- Used for multi-class classification problems

- Measures the difference between predicted probabilities and actual labels

- Formula: CE = -∑(y\_true \* log(y\_pred))

5. Kullback-Leibler (KL) Divergence Loss

- Used for measuring the difference between two probability distributions

- Formula: KL = ∑(y\_true \* log(y\_true/y\_pred))