***J.NIVETHA (122011012816)***

ASSIGNMENT- 1

**Activation functions in Artificial Neural Networks (ANNs)**

It determines whether a neuron should be activated or not by applying a mathematical transformation to the input. They introduce non-linearity, enabling the network to learn complex patterns. Here are some common activation functions:

1. Linear Activation Function

Used in linear regression and some deep networks.

Limitation: Cannot capture complex relationships.

2. Non-Linear Activation Functions

a) Sigmoid (Logistic) Function

Outputs values between 0 and 1.

Good for probability-based outputs.

Issues: Vanishing gradient problem, slow convergence.

b) Tanh (Hyperbolic Tangent) Function

Outputs values between -1 and 1.

Zero-centered, better than sigmoid.

Issues: Still suffers from vanishing gradients.

c) ReLU (Rectified Linear Unit)

Faster training and widely used in deep learning.

Issues: Can suffer from dead neurons (dying ReLU problem).

d) Leaky ReLU

if , else

Solves the dying ReLU problem by allowing small negative values.

e) Parametric ReLU (PReLU)

Similar to Leaky ReLU but learns the negative slope parameter.

Improves performance in some deep networks.

f) ELU (Exponential Linear Unit)

if , else

Reduces vanishing gradients and improves learning.

g) Swish

Developed by Google, works better than ReLU in deep networks.

h) Softmax

Converts logits into probabilities (used in classification tasks).

Choosing an Activation Function:

Classification: Sigmoid, Softmax (for multi-class)

Hidden Layers: ReLU, Leaky ReLU, ELU, Swish

Regression: Linear, ReLU (in some cases)

**Optimizers in Artificial Neural Networks (ANNs)**

These are algorithms that adjust the model’s weights to minimize the loss function and improve accuracy. Here are the main types of optimizers

RMSProp Optimizer

- Modifies the Adagrad optimizer to use a moving average of the squared gradient

- Can help converge faster and is more robust to hyperparameter tuning

Adam Optimizer

- Combines the benefits of Adagrad and RMSProp optimizers

- Adapts the learning rate for each parameter based on the first and second moments of the gradient

AdamW Optimizer

- Modifies the Adam optimizer to use a decoupled weight decay

- Can help improve the performance of the Adam optimizer

Nadam Optimizer

- Combines the benefits of the NAG optimizer and the Adam optimizer

- Can help converge faster and is more robust to hyperparameter tuning

AMSGrad Optimizer

- Modifies the Adam optimizer to use a more conservative approach to adapting the learning rate

- Can help improve the stability of the Adam optimizer

These are some of the most common types of optimizers used in Deep Learning. The choice of optimizer depends on the specific problem, data, and desired performance.

**Types of layers of Artificial neural network in deep learning**

Input Layer

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

- Activation function: None (or linear)

- Purpose: Receives the input data and passes it on to the hidden layers

Hidden Layers

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

- Activation function: Non-linear functions such as ReLU, Tanh, Sigmoid, etc.

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

- Types of hidden layers:

- Dense layers: Fully connected layers where each neuron is connected to every neuron in the previous layer

- Convolutional layers: Used for image and signal processing, where neurons are connected to local regions of the input data

- Recurrent layers: Used for sequential data, where neurons have feedback connections to themselves

Output Layer

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

- Activation function: Depends on the problem type:

- Regression: Linear or non-linear functions such as ReLU, Tanh, etc.

- Binary classification: Sigmoid or logistic function

- Multi-class classification: Softmax function

- Purpose: Produces the final output of the network

Here's an example of a simple neural network architecture:

- Input layer: 784 neurons (28x28 images)

- Hidden layer 1: 256 neurons, ReLU activation

- Hidden layer 2: 128 neurons, ReLU activation

- Output layer: 10 neurons, Softmax activation (for multi-class classification)

**Loss functions in Artificial Neural Networks (ANNs)**

Measure the difference between predicted and actual values. The choice of loss function depends on the type of task (classification, regression, etc.). Here are the main types of loss functions:

Huber Loss

- Used for regression problems with outliers

- Measures the difference between predicted and actual values, with a threshold for outliers

- Formula: Huber = (1/n) \* ∑(if |y\_true-y\_pred| <= δ, (y\_true-y\_pred)^2, δ\*|y\_true-y\_pred|)

Cosine Similarity Loss

- Used for measuring the similarity between two vectors

- Formula: Cosine = -cosine\_similarity(y\_true, y\_pred)

Triplet Loss

- Used for measuring the similarity between two vectors, with a margin for dissimilar vectors

- Formula: Triplet = max(0, margin + cosine\_similarity(y\_true, y\_neg) - cosine\_similarity(y\_true, y\_pos))

Hinge Loss

- Used for binary classification problems with a margin

- Formula: Hinge = max(0, 1 - y\_true \* y\_pred)

Poisson Loss

- Used for regression problems with count data

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