**Assignment - 16**

1. Explain the Activation Functions in your own language

1. Sigmoid:

Ans: Sigmoid activation function squashes the input values between 0 and 1, making it suitable for binary classification tasks.

It has a smooth S-shaped curve, allowing it to model non-linear relationships between inputs and outputs.

However, sigmoid suffers from the vanishing gradient problem, especially for very large or small input values, which can slow down learning in deep neural networks.

1. Tanh:

Ans: Tanh activation function squashes the input values between -1 and 1, centering them around zero.

Similar to sigmoid, tanh is used for modeling non-linearities in the data, but it has a larger range, making it suitable for tasks where the input values may be negative or positive.

Like sigmoid, tanh also suffers from the vanishing gradient problem.

1. ReLU:

Ans: ReLU activation function sets negative input values to zero and leaves positive values unchanged.

ReLU is computationally efficient and helps alleviate the vanishing gradient problem.

It is the most commonly used activation function due to its simplicity and effectiveness, especially in deep neural networks.

1. ELU:

Ans: ELU activation function is similar to ReLU for positive inputs but smoothly handles negative values by allowing them to have a small negative output.

ELU helps prevent dead neurons and can lead to faster convergence compared to ReLU.

1. LeakyReLU:

Ans: LeakyReLU activation function introduces a small slope for negative input values instead of setting them to zero completely.

It helps prevent dying ReLU units and provides a more robust gradient flow during training.

1. Swish:

Ans: Swish activation function is a smooth, non-monotonic function that applies a sigmoid-like transformation to the input.

It has been shown to perform well in some scenarios, but its effectiveness may vary depending on the dataset and architecture.

2. What happens when you increase or decrease the optimizer learning rate?

Ans: Impact of Increasing or Decreasing the Optimizer Learning Rate:

Increasing the learning rate can lead to faster convergence during training, but it may also risk overshooting the minimum and causing instability or divergence.

Decreasing the learning rate can improve the stability of training and help fine-tune the model's performance, but it may result in slower convergence and longer training times.

3. What happens when you increase the number of internal hidden neurons?

Ans: Impact of Increasing the Number of Internal Hidden Neurons:

Increasing the number of internal hidden neurons can increase the model's capacity to learn complex patterns and representations from the data.

However, too many neurons can lead to overfitting, where the model memorizes the training data instead of learning generalizable patterns. It may also increase computational complexity and training time.

4. What happens when you increase the size of batch computation?

Ans: Impact of Increasing the Size of Batch Computation:

Increasing the size of batch computation (batch size) can lead to more stable and consistent updates to the model's parameters during training.

Larger batch sizes can also improve computational efficiency by leveraging parallel processing capabilities of modern hardware.

However, larger batch sizes may require more memory and computational resources, and they may lead to slower convergence or suboptimal solutions in some cases.

5. Why we adopt regularization to avoid overfitting?

Ans: Reason for Adopting Regularization to Avoid Overfitting:

Regularization techniques such as L1 regularization (Lasso), L2 regularization (Ridge), or dropout are used to prevent overfitting, where the model learns to memorize the training data instead of generalizing to unseen data.

Regularization adds a penalty term to the loss function, discouraging overly complex models with large weights or activations.

By constraining the model's capacity and encouraging simpler solutions, regularization helps improve the model's ability to generalize to new, unseen data.

6. What are loss and cost functions in deep learning?

Ans: Loss and Cost Functions in Deep Learning:

Loss functions measure the difference between the predicted output of a neural network and the actual target values.

Cost functions (or objective functions) aggregate the losses from individual data points or batches to produce a single scalar value that represents the overall performance of the model on the entire dataset.

Common loss functions include mean squared error (MSE), cross-entropy loss, and Hinge loss, depending on the task and output type.

7. What do ou mean by underfitting in neural networks?

Ans: Underfitting occurs when a model is too simple to capture the underlying structure or patterns in the data.

In neural networks, underfitting may manifest as high training error and poor performance on both training and validation datasets.

It indicates that the model lacks the capacity to learn from the data effectively and may require more complex architectures or additional training iterations.

8. Why we use Dropout in Neural Networks?

Ans: Dropout is a regularization technique used in neural networks to prevent overfitting.

During training, dropout randomly sets a fraction of the neurons to zero, effectively removing them from the network temporarily.

By randomly dropping neurons, dropout prevents the network from relying too much on specific neurons or features, forcing it to learn more robust representations.

Dropout also acts as an ensemble technique, training multiple subnetworks simultaneously, which helps improve generalization performance.