Advanced Machine Learning

Sample Questions

1. **Explain the concept of backpropagation in the context of deep neural networks, and what are its key steps?**

Backpropagation, is a fundamental algorithm used in the **training** of deep neural networks. It is an **optimization** algorithm that **adjusts the weights** of the network to **minimize** the difference between the predicted output and the actual target output. Backpropagation is based on the principles of **gradient descent** and is an essential component of supervised learning in neural networks. The backpropagation algorithm computes the **gradient of the loss function** with respect to each weight via the **chain rule**. It is an **iterative** process to enhance the network's performance on the given task. By iteratively adjusting the parameters based on the calculated gradients, the network improves its ability to **generalize** and make accurate predictions on new, unseen data.

Key Steps of Backpropagation

i) Forward Pass

The process begins with a forward pass, where input data is fed through the neural network layer by layer, activating each neuron based on the weighted sum of its inputs and applying an activation function.

ii) Compute Loss

The output of the network is compared to the actual target values, and a loss (error) is calculated. This loss represents the difference between the predicted and actual outputs and serves as a measure of how well the network is performing on the given task.

iii) Backward Pass (Backpropagation):

The key step in backpropagation is the backward pass. The algorithm calculates the gradient of the loss with respect to the weights and biases of the network. This is done by applying the chain rule of calculus to compute the partial derivatives of the loss with respect to each weight and bias.

iv) Gradient Descent:

With the gradients obtained from the backward pass, the weights and biases are updated using a gradient descent optimization algorithm. The objective is to adjust the parameters in the direction that minimizes the loss.

Steps 1-4 are repeated iteratively for multiple epochs or until a convergence criterion is met. Each iteration refines the weights and biases of the network, gradually improving its ability to make accurate predictions.

2. **Describe the role of activation functions in deep neural networks. Provide examples of commonly used activation functions and discuss their characteristics.**

In mathematical terms, the activation function serves as a **gate** between the current neuron input and its output, going to the next level. Basically, it decides **whether neurons should be activated or not**. Activation functions play a critical role in deep neural networks by introducing **non-linearities** to the model. Without non-linear activation functions, the entire network would behave like a linear function, and the expressive power of the network would be severely limited. Activation functions are added to introduce non-linearity to the network, thereby **learn complex relationships** and **patterns** in data.

The popular activation functions are:

i) Sigmoid

The output value of it is between 0 and 1, we can use it for classification. Generate smooth gradient, but prone to vanishing gradient problem. It is computationally expensive since it uses exponent. Sigmoid Function curve looks like a S-shape.

ii) Tanh or Hyperbolic Tangent

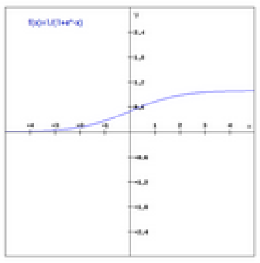
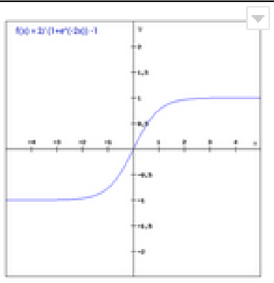
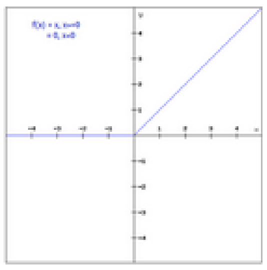
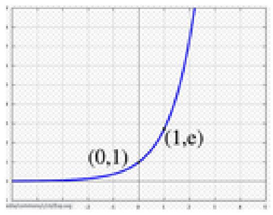
The range of the tanh function is from (-1 to 1). tanh is also sigmoidal (s - shaped). The advantage is that the negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero in the tanh graph.

iii) Relu

It returns 0 if the input is negative and the value of the input if the input is positive. It solves the problem of vanishing gradient for the positive side, however, the problem is still on the negative side. It is fast because we use a linear function in it.

iv) Softmax

It is usually used at the last layer for a classification problem because it returns a set of probabilities, where the sum of them is 1. Moreover, it is compatible with cross-entropy loss, which is usually the loss function for classification problems.

Function	Mathematical Expression	Range	Plot
Sigmoid	$\frac{1}{1 + e^{-x}}$	(0, 1)	
tanh	2 * sigmoid(2x) - 1	(-1, 1)	
ReLU	max(0, x)	[0, inf)	
Softmax	$s(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$	[0, 1]	

3. Discuss the challenges associated with training deep neural networks.

The challenges associated with the DNN training are:

i) Vanishing Gradient Problem:

Deep networks often face the vanishing gradient problem, where gradients become extremely small during backpropagation. This hinders the training of early layers as weight updates become negligible.

ii) Exploding Gradient Problem:

Conversely, the exploding gradient problem can occur, causing gradients to become extremely large. This leads to unstable training and difficulties in finding an optimal set of weights.

iii) Overfitting

Deep networks are prone to overfitting, especially when the number of parameters is high. Overfitting occurs when the model learns to perform well on the training data but fails to generalize to new, unseen data.

iv) Slow Convergence

Training deep networks can be computationally expensive and time-consuming. Slow convergence rates can be problematic, particularly when dealing with large datasets and complex architectures.

4. What is regularization. Explain the L1 and L2 Regularization.

Regularization is a set of techniques used to ensure that a DL model can **generalize to new data** by **preventing overfitting.**

L1 or Lasso regularization introduce a **penalty term** into the model's **loss function** based on the **absolute values** of the model's parameters. Lasso **shrinks the less important feature's coefficient to zero**; thus, removing some feature altogether. So, this works well for **feature selection** in case we have **high-dimensional data with huge number of features**. It enables one to choose a **subset of the most important attributes** and result in **sparse solutions** (i.e. lots of zero weights) and more **robust to outliers.** The **size of a penalty term** is controlled by a hyperparameter **lambda**, which regulates the L1 regularization's regularization strength. As **lambda rises**, more parameters will be **lowered to zero**, improving regularization.

$$\text{cost function} = \sum_{i=1}^{n} \left(\mathrm{y}_{act} - \mathrm{y}_{pred}\right)^2 + \lambda \cdot ||w||_1$$

Unlike L1 regularization, L2 or Ridge regularization **does not induce sparsity**. Instead, it **shrinks** the weights **towards zero** without setting them exactly to zero. This results in a model that **considers all features** but reduces their overall impact on the final prediction. Ridge is **not robust to outliers** as square terms blow up the error differences of the outliers

$$\text{cost function} = \sum_{i=1}^{n} \left(\mathrm{y}_{act} - \mathrm{y}_{pred}\right)^2 + \lambda \cdot ||w||_2^2$$

5. Can you name and explain a few hyperparameters used for training a neural network?

Hyperparameters are any parameter in the model that that **cannot be directly learned** from the regular training process. The **value** of the hyperparameter has to be usually fixed **before**

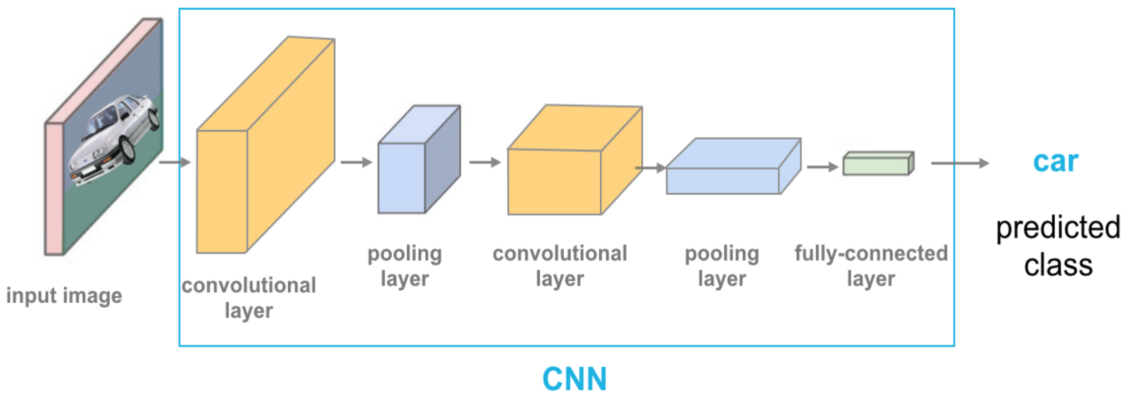
the actual training process begins. Hyperparameters are often used to **tune** the performance of a model, and they can have a **significant impact** on the model's **accuracy, generalization, and other metrics**.

1. Number of nodes: number of inputs in each layer.
2. Learning rate: the rate at which weights are updated.
3. Dropout rate: percent of nodes to drop temporarily during the forward pass.
4. Kernel: matrix to perform dot product of image array
5. Activation function: defines how the weighted sum of inputs is transformed into outputs (e.g. tanh, sigmoid, softmax, Relu, etc)
6. Number of epochs: number of passes an algorithm has to perform for training
7. Batch size: number of samples to pass through the algorithm individually.
8. Momentum: Momentum can be seen as a learning rate adaptation technique that adds a fraction of the past update vector to the current update vector.
9. Optimizers: They focus on getting the learning rate right.

6. Describe the architecture of a typical Convolutional Neural Network (CNN)?

CNN is a **feed-forward neural network.** In a typical CNN architecture, a few convolutional layers are connected in a cascade style. A CNN consists of:

(i) Convolutional Layer
(ii) Pooling Layer (POOL)
(iii) Dense Layer or Fully Connected (FC) Layer
(iv) Activation Layer



Convolution Layer

The **basic unit** of the CNN architecture is a convolutional layer. Convolutional layer performs an operation called a “**convolution**“ and computes the **convolution of input images** along with the NN **weights** and performs **feature extraction. The c**onvolution operation is **linear.** This layer’s main **processing parameters** are a group of learnable **filters or masks or kernels**, which generate the feature maps. The feature maps are used to define a new input to the next layer. The number of these layers can be decided depending on the complexity of the data.

Pooling Layer

The CNN architecture can be built by **stacking** pooling and convolutional layers in an intervened manner. Using the pooling layer **reduces the spatial resolution** of the feature maps. The frequently used pooling methods are: Average Pooling and Max Pooling. They are **spatially invariant** to input **distortions** and **translations** with less overfitting.

Fully Connected Layer

The feature map from the pooling layer is given to dense FC layers and is utilized as the **final layer for the classification**. **More abstract feature** representations are extracted while moving through the entire network. The FC layers perform **high-level reasoning** and generate new features from the existing features. The **neurons** in the FC layer are **fully connected** to all the previous layers. Generally, this layer has the **most number of weights** with no sharing. The FC layer can thus **take time to train** as compared to other layers.

Activation Layer

Usually, a **Relu** activation is employed in the CNN architecture. This layer introduces the non-linearity to the network and converts all the negative pixels to zero. The final output is a rectified feature map. For multi class classifications a **Softmax** activation is used at the end of FC layer.