UNDERSTANDING NEURAL NETWORKS AND THE MATHEMATICS BEHIND THEM – Onyiriuba Leonard

What we will be answering:

1. What are neural networks
2. What are artificial neural networks
3. What are the types of artificial neural network architecture we have?
4. What is forward propagation
5. What is backward propagation
6. What are the types of cost functions we use in deep learning and under what scenarios are they applied?
7. What is a perceptron?
8. How does a perceptron apply to deep learning networks?
9. Types of perceptron
10. How are perceptron implemented?
11. Explaining multi-layer perceptron
12. What are activation functions
13. Types of activation functions
14. Learning when to use each of the different types of activation functions
15. What is gradient descent
16. Types of gradient descent
17. Differentiating the types of gradient descent with examples
18. When will I choose to use Stochastic Gradient Descent
19. When will I choose to use Batch Gradient Descent
20. When will I choose to use Mini-Batch Gradient Descent
21. What is learning rate
22. What is vanishing gradient descent
23. What is the “Curse of Dimensionality” in deep learning
24. How many output nodes are we meant to have and how do we determine it?
25. Understanding the cost function and how it affects gradient descent
26. What does the graph of a cost or loss function in machine learning look like?
27. Each neuron in the input layer represents what in our deep learning process
28. How many nodes are we meant to have in the input layer?
29. What is a dense network?
30. What is a fully connected network and how is it different from a dense network
31. How do we know the number of hidden layers to have?
32. How do we know the number of nodes in each hidden layer?
33. How do we avoid overfitting when building deep networks?
34. How do we avoid underfitting in building deep neural networks
35. What are the different types of layers we can have when building a deep network?
36. Illustrating the cost function changes to changes in the predicted value of Y towards getting to the minimum point
37. Showing a step-by-step implementation of the mathematics behind the way a neural network learns
38. Showing a step-by-step implementation of the code behind how artificial neural networks learn
39. When will we use a fully-connected-layer in deep learning and what other alternatives do we have?
40. When are alternatives to a fully-connected-layer in deep learning and when are they applied
41. How does the amount of available data affect the number of nodes to use in the hidden layer
42. What exactly is going on in the hidden layer and what exactly are the neurons picking up?
43. What do the features learnt in the hidden layer signify?
44. How does implementing cross-validation help with finding a balance to determine the right number of hidden layers to have?
45. How to know the best learning rate to use
46. What are optimizers in deep learning
47. How do choose the right number of epochs
48. How do I interpret the results after epoch has been completed
49. What are the types of models we can import when using KERAS
50. What are the differences between the models provided by KERAS and when can they be implemented
51. Is a perceptron one hidden layer?
52. Is perceptron used only for Binary classification? And why? Why not regression?
53. Examples of a simple perceptron illustration and implementation?
54. What exactly is a Feed-Forward Neural Network doing? And why is it different from a Multi-Layer Perceptron
55. Is a Feed-Forward Neural Network the same thing as a Perceptron
56. What is ADAM and when is it used in DEEP LEARNING?
57. Does it mean that if you have a neural network with an input layer, one hidden layer with 5 nodes and an output layer, this is a single layer perceptron?
58. For a Multi Layer Perceptron, do we have more than one hidden layer?
59. Does adding more hidden layers give us higher results

S I M P L I F I E D QUESTIONS

LET’S START HERE:

It seems like you're looking for explanations to various concepts related to neural networks and deep learning. I'll provide brief answers to each of your questions:

1) What are neural networks?

Neural networks are a class of machine learning models inspired by the structure and functioning of the human brain. They consist of interconnected nodes (neurons) organized in layers, which process and transmit information through weighted connections.

2) What are artificial neural networks?

Artificial Neural Networks (ANNs) are a specific type of neural network designed to perform tasks like pattern recognition, classification, regression, and more. They consist of an input layer, one or more hidden layers, and an output layer.

3) What are the types of artificial neural network architecture?

There are various architectures, including feedforward neural networks (where data moves in one direction), convolutional neural networks (for image analysis), recurrent neural networks (for sequential data), and more advanced architectures like transformers and GANs.

4) What is forward propagation?

Forward propagation is the process through which input data is passed through the neural network's layers in order to compute an output. Each layer applies a transformation to the input using learned weights and activation functions.

5) What is backward propagation?

Backward propagation (also known as backpropagation) is the process of calculating the gradients of the model's parameters with respect to a chosen loss function. It's used for updating the weights during the training process using optimization algorithms like gradient descent.

6) What are the types of cost functions in deep learning and when are they used?

Common cost functions include Mean Squared Error (MSE) for regression, Cross-Entropy for classification, and various custom loss functions depending on the problem. They measure the discrepancy between predicted and actual outputs.

7) What are activation functions?

Activation functions introduce non-linearity to neural networks. They determine whether a neuron should activate (fire) based on the weighted sum of its inputs. Common activation functions include ReLU, Sigmoid, and Tanh.

8) What is gradient descent?

Gradient descent is an optimization algorithm used to minimize the cost function by iteratively adjusting the model's parameters (weights) in the direction of the steepest descent of the cost function gradient.

9) Understanding the cost function's effect on gradient descent:

The cost function guides gradient descent to find optimal weights. A well-chosen cost function ensures convergence to an accurate solution, while a poor choice might lead to slow convergence or getting stuck in local minima.

10) What does the graph of a cost or loss function look like?

The graph of a cost function typically shows its value on the y-axis and the model's parameters (weights) on the x-axis. It often resembles a bowl-shaped curve with a minimum point representing the optimal solution.

11) Each neuron in the input layer represents what in deep learning?

Each neuron in the input layer represents a feature or input variable in the dataset. It passes the input data to the subsequent layers for processing.

12) How many nodes should the input layer have?

The input layer should have a node for each feature in the dataset you're working with. The number of nodes is equal to the dimensionality of your input data.

13) What is a dense network?

A dense network (also known as a fully connected network) is a type of neural network architecture where each neuron in a layer is connected to every neuron in the previous and subsequent layers.

14) What is a fully connected network, and how is it different from a dense network?

A fully connected network is the same as a dense network. The terms are often used interchangeably to describe the architecture where every neuron is connected to every neuron in adjacent layers.

15) How do we determine the number of hidden layers to have?

The number of hidden layers is often determined through experimentation and domain knowledge. Too few layers might result in underfitting, while too many can lead to overfitting. Techniques like cross-validation can help find an appropriate balance.

16) How do we know the number of nodes in each hidden layer?

There is no fixed rule for determining the number of nodes in hidden layers. It often depends on the complexity of the problem, the amount of available data, and experimentation. Starting with a moderate number and adjusting based on performance is a common approach.

THEN WE MOVE HERE:

It looks like you have a comprehensive list of questions related to neural networks, their components, and various concepts in deep learning. While I can't provide an in-depth answer to all of these questions in a single response, I can certainly give you a brief overview of each topic. If you have more specific questions or need detailed explanations for any of the points, feel free to ask.

1) Neural Networks: Neural networks are a class of machine learning algorithms inspired by the human brain's structure and function. They consist of interconnected nodes, called neurons, organized in layers. Neurons in one layer receive inputs from the previous layer and produce outputs that become inputs for the next layer.

2) Artificial Neural Networks (ANNs): ANNs are a specific type of neural network used to model complex patterns and relationships in data. They consist of an input layer, one or more hidden layers, and an output layer. Each neuron in a layer is connected to neurons in the adjacent layers.

3) Types of ANN Architectures: There are several types of ANN architectures, including feedforward neural networks, recurrent neural networks (RNNs), convolutional neural networks (CNNs), and more. Each architecture is designed to handle different types of data and tasks.

4) Forward Propagation: Forward propagation is the process by which inputs are passed through the network's layers to generate predictions or outputs. Neurons calculate weighted sums of inputs and apply activation functions to produce outputs.

5) Backward Propagation: Backward propagation (also known as backpropagation) is the process used in training neural networks. It involves calculating the gradients of the cost function with respect to the network's weights and biases. These gradients guide weight updates during optimization.

6) Cost Functions: Cost functions measure the difference between predicted outputs and actual targets. Common cost functions include mean squared error (MSE) for regression tasks and categorical cross-entropy for classification tasks.

7) Perceptron: A perceptron is the simplest form of a neural network, consisting of a single neuron that takes multiple inputs, applies weights, sums them, and passes the result through an activation function to produce an output.

8) Perceptron in Deep Learning: Perceptrons serve as the basic building blocks of deep learning networks. In modern deep learning, more complex and layered architectures are used to model intricate relationships in data.

9) Types of Perceptrons: There are single-layer perceptrons and multi-layer perceptrons (MLPs). Single-layer perceptrons can only solve linearly separable problems, while MLPs with hidden layers can handle more complex tasks.

10) Perceptron Implementation: Perceptrons are implemented by assigning weights to input features, summing the weighted inputs, and passing the result through an activation function.

11) Multi-Layer Perceptron (MLP): MLP is a type of neural network architecture that consists of an input layer, one or more hidden layers with nonlinear activation functions, and an output layer.

12) Activation Functions: Activation functions introduce nonlinearity to the network, allowing it to capture complex relationships in data. Common activation functions include sigmoid, tanh, and ReLU.

13) Types of Activation Functions: Common activation functions include sigmoid, tanh, ReLU, Leaky ReLU, and softmax.

14) Choosing Activation Functions: Activation functions are chosen based on the task and network architecture. ReLU is often preferred in hidden layers due to its efficiency, while softmax is used for multiclass classification.

15) Gradient Descent: Gradient descent is an optimization algorithm used to minimize the cost function by adjusting the weights and biases of the neural network based on the calculated gradients.

16) Types of Gradient Descent: Gradient descent comes in various forms, including batch gradient descent, stochastic gradient descent (SGD), and mini-batch gradient descent.

17) Differentiating Gradient Descent Types: Batch GD uses the entire dataset, SGD uses one data point at a time, and mini-batch GD uses a subset of the data. SGD and mini-batch GD often converge faster but have more noise.

18) Stochastic Gradient Descent (SGD): SGD is chosen when the dataset is large, as it processes individual examples and is computationally efficient.

19) Batch Gradient Descent: Batch GD is used when the dataset fits into memory, providing more stable updates due to the use of the entire dataset.

20) Mini-Batch Gradient Descent: Mini-batch GD strikes a balance between SGD and batch GD, offering faster convergence and reduced noise compared to SGD.

21) Learning Rate: The learning rate determines the step size taken during gradient descent. It affects the rate of convergence and stability of optimization.

22) Vanishing Gradient Descent: This occurs when gradients in deep networks become too small during backpropagation, leading to slow learning in early layers.

23) Curse of Dimensionality: This refers to the challenges posed by high-dimensional data in deep learning, where data becomes sparse and distances between data points lose meaning.

24) Output Nodes: The number of output nodes depends on the task. For regression, there's typically one output node per continuous target variable. For classification, it's equal to the number of classes.

25) Cost Function's Effect: The cost function guides gradient descent to find optimal weights that minimize prediction errors.

26) Cost Function Graph: The graph of a cost function usually resembles a bowl shape, with the minimum representing the point of lowest error.

27) Neurons in Input Layer: Each neuron in the input layer represents a feature or input variable of the dataset.

28) Nodes in Input Layer: The number of nodes in the input layer corresponds to the number of input features.

29) Dense Network: A dense network refers to a fully connected neural network, where each neuron is connected to every neuron in the adjacent layers.

30) Fully Connected vs. Dense Network: Both terms are often used interchangeably to refer to the same concept of all-to-all connections between neurons in adjacent layers.

31) Number of Hidden Layers: The number of hidden layers is determined through experimentation and problem complexity. Generally, starting with a single hidden layer is common, and complexity may demand more layers.

32) Number of Nodes in Hidden Layers: The number of nodes in hidden layers depends on the complexity of the problem, available data, and experimentation.

33) Avoiding Overfitting: Overfitting can be mitigated by using techniques like dropout, regularization, and collecting more diverse data.

34) Avoiding Underfitting: Underfitting can be reduced by increasing model complexity, using appropriate activation functions, and providing sufficient training data.

35) Types of Layers: Common types of layers include fully connected (dense) layers, convolutional layers, pooling layers, and recurrent layers.

36) Cost Function Minimization: The cost function guides the learning process towards the minimum point by adjusting weights and biases.

37) Step-by-Step Learning Implementation: The step-by-step learning process involves feeding input data, calculating weighted sums, applying activation functions, computing the cost, and adjusting weights through backpropagation.

38) Step-by-Step Code Implementation: Code for neural network learning involves setting up layers, initializing weights, implementing forward and backward propagation, and updating weights using an optimizer.

39) Fully-Connected Layer Usage: Fully-connected layers are used in various tasks, such as image classification and natural language processing.

40) Alternatives to Fully-Connected Layer: Alternatives include convolutional layers for image data, recurrent layers for sequential data, and attention mechanisms for sequence-to-sequence tasks.

41) Data Amount and

Hidden Nodes: More data often allows for larger hidden layers, as it helps prevent overfitting and provides better generalization.

42) Hidden Layer's Role: Hidden layers learn complex representations of the input data, extracting higher-level features that improve the network's ability to make accurate predictions.

43) Features Learned in Hidden Layer: The features learned in hidden layers can represent patterns, edges, textures, and other informative structures present in the data.

44) Cross-Validation for Hidden Layers: Cross-validation helps find the optimal number of hidden layers by assessing performance on different subsets of the training data.

45) Choosing Learning Rate: The learning rate is chosen through experimentation, trying different values and observing convergence speed and stability.

46) Optimizers: Optimizers are algorithms that update network weights during training to minimize the cost function. Examples include SGD, Adam, and RMSProp.

47) Choosing Number of Epochs: The number of epochs is chosen by monitoring the validation loss. Training can stop when validation loss plateaus.

48) Interpreting Epoch Results: After each epoch, you can analyze training and validation loss to understand the model's learning progress and potential overfitting.

49) Types of Models in Keras: Keras provides Sequential models, Functional API models, and Subclassing models for building neural networks.

50) Differences in Keras Models: Sequential models are simple, Functional API models allow for more complex architectures, and Subclassing models offer full flexibility but require more coding