Inroduction to Deep Learning.

1. What is deep learning and its significance in AI

Ans: Definition: Deep learning is a subset of machine learning that uses artificial neural networks with multiple processing layers to learn hierarchical representations of data. These layered architectures enable the automatic learning of features at increasing levels of abstraction.

Significance in AI:

-Feature Learning: Eliminates need for manual feature engineering -Scalability: Performance improves with more data and computation

-Versatility: Successful across diverse domains (vision, speech, text)

-State-of-the-art Results: Dominates benchmarks in many AI tasks

-End-to-end Learning: Can map raw inputs directly to outputs

Impact Areas:

-Computer vision (image recognition, object detection)

-Natural language processing (translation, summarization)

-Speech recognition and generation

-Game playing (AlphaGo, OpenAI Five)

-Scientific discovery (protein folding, drug discovery)

2. Fundamental components of artificial neural networks

Ans:

Neurons: Basic computational units that receive inputs, apply transformations, and produce outputs. Mathematically: $y = f(\sum(w_i*x_i) + b)$

Connections: Pathways between neurons that transmit information. Each has an associated weight determining its importance.

Weights: Numerical parameters that scale the influence of inputs. Learned during training to minimize error.

Biases: Additional parameters that allow shifting the activation function. Help the model fit the data better.

Layers:

-Input layer: Receives raw data -Hidden layers: Intermediate processing -Output layer: Produces final prediction

Activation Functions: Introduce non-linearity (ReLU, sigmoid, tanh)

Loss Function: Measures prediction error

Optimizer: Algorithm for updating weights (SGD, Adam)

3. Roles of neurons, connections, weights, and biases

Ans:

Neurons: -Receive weighted inputs from previous layer -Compute weighted sum plus bias -Apply activation function -Pass output to next layer

Connections:

-Define network topology -Determine information flow paths -Enable hierarchical feature learning

Weights: -Determine connection strength -Store learned knowledge -Adjusted during backpropagation -Initialized randomly then optimized

Biases: -Allow shifting decision boundaries -Provide additional flexibility -Help neurons fire when inputs are zero -Learned parameters like weights

4. ANN architecture and information flow example

Ans:

Typical Architecture: Input Layer → Hidden Layer 1 → ... → Hidden Layer N → Output Layer

Information Flow Example (XOR Problem):

Input layer receives two binary values (x1, x2)

First hidden layer computes: h1 = ReLU(w11x1 + w12x2 + b1) h2 = ReLU(w21x1 + w22x2 + b2)

Output layer computes: y = σ(w3h1 + w4h2 + b3)

Final output (0 or 1) indicates XOR result

Visualization: x1 →(w11)→ h1 →(w3)→ y x2 →(w12)↗ ↗ x1 →(w21)→ h2 →(w4)→ y x2 →(w22)↗

5. Perceptron learning algorithm

Ans:

Algorithm Steps:

Initialize weights randomly (small values)

For each training example (x, target): a. Compute output: y = step_function(w·x + b) b. Calculate error: δ = target - y c. Update weights: w_i ← w_i + η·δ·x_i d. Update bias: b ← b + η·δ

Repeat until convergence or max epochs

Weight Adjustment:

η is learning rate ($0 < \eta \leq 1$)

Updates proportional to input magnitude

Only occurs for misclassified samples

Changes decision boundary orientation

Key Properties:

Guaranteed convergence for linearly separable data

Only works with binary outputs

Foundation for more complex algorithms

6. Activation functions in hidden layers Importance:

Introduce non-linearity → enables complex function approximation

Control neuron output range

Affect gradient flow during backpropagation

Determine if neuron activates ("fires")

Enable learning hierarchical features

Common Activation Functions:

ReLU: max(0,x)

Pros: Simple, avoids vanishing gradient

Cons: "Dying ReLU" problem

Sigmoid: 1/(1+e^-x)

Pros: Smooth gradient, [0,1] output

Cons: Vanishing gradients, not zero-centered

Tanh: (e^x-e^-x)/(e^x+e^-x)

Pros: Zero-centered, [-1,1] range

Cons: Vanishing gradients

Leaky ReLU: max(αx,x) (α≈0.01)

Pros: Fixes dying ReLU problem

Cons: Extra parameter

Softmax: (e^x_i)/∑(e^x_j)

Used in output layer for classification

Neural Network Architectures

1. Feedforward Neural Network (FNN) Basic Structure:

Acyclic connections between layers

Information flows input → output (no loops)

Typically fully connected between layers

Minimum: 1 input, 1 hidden, 1 output layer

Activation Function Purpose:

Without it, FNN could only learn linear transformations

Multiple linear layers = single linear layer

Enables learning non-linear decision boundaries

Allows stacking multiple layers effectively

2. Convolutional and Pooling Layers in CNN Convolutional Layers:

Apply learned filters/kernels to input

Detect local patterns (edges, textures)

Share weights across spatial positions

Preserve spatial relationships

Output feature maps represent learned features

Pooling Layers:

Types: Max, Average, L2-norm pooling

Achieves:

Spatial invariance (translation tolerance)

Dimensionality reduction

Computational efficiency

Control overfitting

Typically uses 2×2 windows with stride 2

3. RNNs and Sequential Data Handling Key Differentiator:

Recurrent connections create internal memory

Can process variable-length sequences

Shares parameters across time steps

Sequential Data Processing:

Maintains hidden state $h_t = f(h_{t-1}, x_t)$

Processes one time step at a time

Can learn temporal dependencies

Output depends on entire history

Challenges:

Vanishing/exploding gradients

Difficulty learning long-range dependencies

Computationally sequential

4. LSTM Components and Vanishing Gradient Solution Components:

Cell State: Main memory conveyor belt

Forget Gate: Decides what to discard

Input Gate: Decides what new info to store

Output Gate: Decides what to output

Vanishing Gradient Solution:

Constant error carousel in cell state

Gated mechanisms control information flow

Additive updates to cell state (not multiplicative)

Selective remembering/forgetting

Gradient can backpropagate unchanged

5. GAN: Generator and Discriminator Roles Generator:

Learns data distribution

Maps random noise to synthetic samples

Training objective: Fool discriminator

Typically uses transposed convolutions

Discriminator:

Distinguishes real vs. fake samples

Acts as learned loss function

Training objective: Correct classification

Typically convolutional architecture

Training Process:

Train D on real + generated samples

Train G to maximize D's mistakes

Alternating min-max game: $\min_G \max_D V(D,G) = E[\log D(x)] + E[\log(1-D(G(z)))]$