Slide 1: Introduction and Outline

Artificial Neural Networks (ANN): Computing systems inspired by the biological neural networks in the brain.

Deep Learning: A subset of machine learning focusing on neural networks with many layers.

Linear Classification: The process of classifying data points into categories using a linear decision boundary.

Cost Functions: Mathematical functions used to measure the error of a model’s predictions.

Optimization: Techniques to adjust model parameters to minimize the cost function.

Backpropagation Algorithm: A method for training neural networks by propagating the error backward through layers.

Slide 5: Deep Learning Challenges

Big Data: Large datasets requiring advanced techniques for processing and analysis.

Graphics Processing Units (GPU): Hardware accelerators for parallel processing, crucial for deep learning.

Software Toolboxes: Libraries and frameworks (e.g., TensorFlow, PyTorch) for implementing deep learning.

Slide 6: Linear Regression

Linear Model: A model where the relationship between input and output is linear.

Quadratic Model: A model including squared terms, allowing for curvature in predictions.

Slide 7: Why Deep Learning?

Feature Extraction: The process of identifying and extracting relevant features from raw data.

Low, Mid, and High-Level Features: Features representing increasing abstraction levels in data representation.

Slide 8-10: Neural Network (Perceptron)

Artificial Neuron (Perceptron): A basic computational unit in a neural network, inspired by biological neurons.

Inputs, Weights, Bias, Activation Function: Components of a perceptron contributing to the final output.

Forward Propagation: The process of calculating the output from input through a neural network.

Slide 11-16: Activation Functions

Sigmoid, Tanh, ReLU (Rectified Linear Unit): Common activation functions introducing non-linearities in neural networks.

Gradient Vanishing Problem: A challenge where gradients become too small for effective learning in deep networks.

Batch Normalization: A technique to normalize activations within layers to stabilize and speed up training.

Slide 17-26: Forward Propagation Examples

Dense Layer: A layer where every input is connected to every output via weights.

Weight Update: Adjusting model weights during training to minimize loss.

Weight: Refers to the adjustable parameters of a neural network. These are updated during training to minimize the loss function.

LearningRate (η): A scalar hyperparameter that determines the step size at each iteration during optimization. A smaller value ensures slower but more stable convergence; a larger value speeds up training but risks overshooting.

Gradient: The derivative of the loss function with respect to the weight. It represents the direction and magnitude of change needed to reduce the loss.

Slide 27-33: Loss Functions

Loss Function: Measures the error between predictions and actual outcomes.

Why Are Loss Functions Important?

Guides Model Training: The loss function defines the objective for optimization algorithms (e.g., gradient descent) to minimize.

Evaluate Model Performance: By observing the loss value, you can assess how well the model is learning.

Key Points to Consider When Choosing a Loss Function

Task Type:

Regression → Use MSE or Huber Loss.

Classification → Use Binary or Categorical Cross-Entropy.

Data Characteristics:

If sensitive to outliers → Consider Huber Loss or MAE.

Large class imbalance → Use weighted loss functions.

Output Format:

Probabilities → Cross-Entropy Loss.

Continuous values → MSE.

Mean Squared Error (MSE): A loss function for regression tasks.

Binary Cross Entropy: A loss function for binary classification problems.

Slide 34-49: Optimization Algorithms

Gradient Descent: An algorithm to minimize the loss function by updating weights.

Stochastic Gradient Descent (SGD): A variation of gradient descent using random samples for updates.

Mini-batches: Subsets of data used in SGD for computational efficiency.

Slide 50-55: Overfitting and Regularization

Overfitting: When a model performs well on training data but poorly on unseen data.

Regularization: Techniques like dropout or early stopping to prevent overfitting.

Dropout: Randomly disabling neurons during training to improve generalization.

Why is Dropout Needed?

Overfitting: Neural networks, especially deep networks, tend to overfit when they memorize training data instead of generalizing to unseen data.

Solution: Dropout introduces randomness, forcing the network to learn more robust and generalized features, rather than relying too heavily on specific neurons.

How Does Dropout Work?

Random Neuron Deactivation:

For each forward pass during training, a random subset of neurons in a layer is deactivated (set to zero) based on a predefined probability (p, called the dropout rate).

For example: p=0.3: 30% of neurons are deactivated. p=0.5: 50% of neurons are deactivated.

Scaling During Training:

The remaining active neurons are scaled by 1/(1−p) to maintain the same expected output as the full network. This ensures that the overall magnitude of activations doesn’t change.

Inference Mode (Testing/Prediction):

During testing, no neurons are dropped out. The full network is used, but weights are scaled down by the same 1−p factor to match the expected activations seen during training.

Slide 56-64: Feature Extraction in Deep Learning

Feature Extraction: Identifying meaningful data patterns for machine learning models.

Convolutional Layers: Layers designed to process spatial data like images using filters.

Pooling Layers: Downsampling techniques to reduce data dimensionality while preserving important features.

Slide 65-77: Convolutional Neural Networks (CNNs)

Sliding Window: A technique for feature extraction in image data.

Filters: Matrices used to extract specific features from input data.

Convolutional Neural Network (CNN): A type of neural network well-suited for image recognition tasks.

Slide 78-85: Deep Learning Architectures

AlexNet, VGG, ResNet, YOLO: Popular deep learning architectures for tasks like image classification and object detection.

Slide 86-88: Summary and Applications

Reinforcement Learning with CNNs: Combining reinforcement learning and convolutional networks for tasks requiring spatial feature extraction.

Generalization: Ensuring models perform well on unseen data.