CS24307 • MODULE 5

Module 5: Machine Learning & Neural Networks

Topics 30 to 38 • Decision Trees ID3, Perceptrons & Backpropagation Math

Module 5 Table of Contents (Topics 30 to 38)

- **Topic 30:** Forms of Learning (Supervised, Unsupervised, RL)
- **Topic 31:** Decision Tree Learning & ID3 Algorithm
- **Topic 32:** Entropy & Information Gain Calculations
- **Topic 33:** Gini Index, Overfitting & Tree Pruning
- **Topic 34:** Biological & McCulloch-Pitts Artificial Neurons
- **Topic 35:** Single-Layer Perceptron & Linear Separability (XOR Problem)
- **Topic 36:** Multi-Layer Perceptrons (MLP) Architecture
- **Topic 37:** Mathematical Derivation of Backpropagation
- **Topic 38:** Activation Functions & Solved University Exam Bank

Topic 30 to 33: Decision Tree Induction & Information Gain Mathematics

In **Decision Tree Learning**, candidate attributes are recursively evaluated at each split to maximize the purity of child nodes:

Shannon Entropy & Information Gain Formulas:

$$\mathbf{H(S) = -\sum_{i=1}^{c} p_i \log_2(p_i)}$$

$$\mathbf{Gain(S, A) = H(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} H(S_v)}$$

$$\mathbf{\text{Gini Index: } Gini(S) = 1 - \sum_{i=1}^{c} p_i^2}$$

🏛️ Step-by-Step Solved Problem: ID3 Decision Tree Root Attribute Selection

A dataset has 14 instances (9 Yes, 5 No). Initial parent entropy is:

$$H(S) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = -(0.643)(-0.637) - (0.357)(-1.485) = \mathbf{0.940} \text{ bits}$$

Evaluate Attribute **Windy** ($\text{Weak: } 6 \text{ Yes}, 2 \text{ No};\ \text{Strong: } 3 \text{ Yes}, 3 \text{ No}$):

$$H(S_{\text{Weak}}) = -\frac{6}{8}\log_2\left(\frac{6}{8}\right) - \frac{2}{8}\log_2\left(\frac{2}{8}\right) = \mathbf{0.811} \text{ bits}$$

$$H(S_{\text{Strong}}) = -\frac{3}{6}\log_2\left(\frac{3}{6}\right) - \frac{3}{6}\log_2\left(\frac{3}{6}\right) = \mathbf{1.000} \text{ bits}$$

$$\text{Gain}(S, \text{Windy}) = 0.940 - \left[\frac{8}{14}(0.811) + \frac{6}{14}(1.000)\right] = 0.940 - (0.463 + 0.429) = \mathbf{0.048} \text{ bits}$$

Topic 34 to 38: Neural Networks, Backpropagation & Non-Linear Activations

🏛️ Why Single-Layer Perceptrons Cannot Solve XOR (Minsky & Papert 1969)

A single perceptron $y = \text{step}(w_1x_1 + w_2x_2 - \theta)$ represents a single straight hyper-plane linear decision boundary. The XOR function requires:

$$\text{For } (0,0) \to 0: \ 0 < \theta \implies \theta > 0$$

$$\text{For } (1,0) \to 1: \ w_1 \geq \theta$$

$$\text{For } (0,1) \to 1: \ w_2 \geq \theta$$

$$\text{For } (1,1) \to 0: \ w_1 + w_2 < \theta$$

$$\text{Adding } (1,0) \text{ and } (0,1) \implies w_1 + w_2 \geq 2\theta > \theta \implies \text{CONTRADICTION! (Non-linearly separable)}$$

Complete Mathematical Derivation of Backpropagation Gradient: Let squared loss be $E = \frac{1}{2}\sum_k (y_k - \hat{y}_k)^2$. Using the Multivariate Chain Rule for output layer weight w_{jk}:

$$\frac{\partial \mathbf{E}}{\partial \mathbf{w_{jk}}} = \frac{\partial \mathbf{E}}{\partial \hat{\mathbf{y}}_\mathbf{k}} \cdot \frac{\partial \hat{\mathbf{y}}_\mathbf{k}}{\partial \mathbf{z_k}} \cdot \frac{\partial \mathbf{z_k}}{\partial \mathbf{w_{jk}}} = -(\mathbf{y_k} - \hat{\mathbf{y}}_\mathbf{k}) \cdot \sigma'(\mathbf{z_k}) \cdot \mathbf{a_j} = -\delta_\mathbf{k} \cdot \mathbf{a_j}$$

$$\text{Weight Update: } \mathbf{w_{jk}} \leftarrow \mathbf{w_{jk}} - \eta \frac{\partial \mathbf{E}}{\partial \mathbf{w_{jk}}} = \mathbf{w_{jk}} + \eta \cdot \delta_\mathbf{k} \cdot \mathbf{a_j}$$

For hidden layer weight w_{ij}:

$$\delta_\mathbf{j} = \sigma'(\mathbf{z_j}) \sum_\mathbf{k} \delta_\mathbf{k} \mathbf{w_{jk}} \implies \mathbf{w_{ij}} \leftarrow \mathbf{w_{ij}} + \eta \cdot \delta_\mathbf{j} \cdot \mathbf{a_i}$$

Activation Function	Mathematical Formula	Output Range	Derivative $\sigma'(z)$
Sigmoid (Logistic)	$\sigma(z) = \frac{1}{1+e^{-z}}$	$(0, 1)$	$\sigma(z)(1-\sigma(z))$
Hyperbolic Tangent (Tanh)	$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$(-1, 1)$ (Zero-centered)	$1 - \tanh^2(z)$
ReLU (Rectified Linear)	$f(z) = \max(0, z)$	$[0, \infty)$	1 if $z > 0$ else 0
Softmax	$\sigma(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$	$(0, 1)$ with $\sum p_i = 1$	$p_i(\delta_{ij} - p_j)$

Topic 38.2: Master University Examination Solved Question Bank (10 Solved Questions)

Q1. State the Perceptron Convergence Theorem. What happens if the training data is not linearly separable? (8 Marks)

The **Perceptron Convergence Theorem** (Rosenblatt 1962) proves that if a dataset is linearly separable (there exists a hyperplane with margin $\gamma > 0$), the perceptron learning rule is guaranteed to find a separating hyperplane in a finite number of weight updates $k \leq \left(\frac{R}{\gamma}\right)^2$ (where $R = \max \|x\|$). If the dataset is *non-linearly separable*, the perceptron weights oscillate indefinitely and never converge.

Q2. Complete Step-by-Step Backpropagation Numerical Walkthrough with explicit calculations. (12 Marks)

Consider a network: $x_1 = 0.05, x_2 = 0.10$, target $y = 0.01$, $\eta = 0.5$. Hidden weights: $w_1 = 0.15, w_2 = 0.20, w_3 = 0.25, w_4 = 0.30$, bias $b_1 = 0.35$. Output weights: $w_5 = 0.40, w_6 = 0.45$, bias $b_2 = 0.60$.
1. Forward Pass: $z_{h1} = (0.05)(0.15) + (0.10)(0.20) + 0.35 = 0.3775 \implies a_{h1} = \sigma(0.3775) = 0.59327$. $z_{h2} = (0.05)(0.25) + (0.10)(0.30) + 0.35 = 0.3925 \implies a_{h2} = \sigma(0.3925) = 0.59688$. Output: $z_{o1} = (0.59327)(0.40) + (0.59688)(0.45) + 0.60 = 1.1059 \implies a_{o1} = \sigma(1.1059) = \mathbf{0.75137}$. Error $E = \frac{1}{2}(0.01 - 0.75137)^2 = \mathbf{0.2748}$.
2. Backward Pass: $\delta_{o1} = (y - a_{o1})a_{o1}(1 - a_{o1}) = (0.01 - 0.75137)(0.75137)(0.24863) = \mathbf{-0.1385}$. Weight gradient: $\frac{\partial E}{\partial w_5} = -\delta_{o1} a_{h1} = -(-0.1385)(0.59327) = \mathbf{+0.08217}$. New weight: $\mathbf{w_5^{new} = 0.40 - 0.5(0.08217) = 0.3589}$.

Q3. Contrast ID3, C4.5, and CART Decision Tree Algorithms. (8 Marks)

• **ID3 (Quinlan):** Uses Information Gain (Entropy). Restricted to discrete categorical attributes; biased toward attributes with many values.
• **C4.5 (Quinlan):** Uses *Gain Ratio* $(\text{Gain}/\text{SplitInfo})$ to eliminate multi-value bias. Handles continuous attributes (dynamic thresholds) and missing values. Employs post-pruning.
• **CART (Breiman):** Uses *Gini Impurity*. Builds strictly binary decision trees and supports both classification and regression.

Q4. Explain the Bias-Variance Trade-off in Machine Learning with an error decomposition derivation. (8 Marks)

Expected prediction error on unseen data decomposes into: $\mathbb{E}[(\mathbf{y} - \hat{\mathbf{f}}(\mathbf{x}))^2] = \text{Bias}^2(\hat{\mathbf{f}}(\mathbf{x})) + \text{Var}(\hat{\mathbf{f}}(\mathbf{x})) + \sigma^2$ (where σ^2 is irreducible noise).
• **High Bias (Underfitting):** Model is overly simplistic and fails to capture underlying true relationships (e.g. linear model on quadratic data).
• **High Variance (Overfitting):** Model is overly complex and fits random training noise, failing to generalize to test data.

Q5. Explain the Vanishing Gradient Problem in Deep Neural Networks and how ReLU resolves it. (8 Marks)

For Sigmoid activation $\sigma(z) = \frac{1}{1+e^{-z}}$, the derivative is $\sigma'(z) = \sigma(z)(1-\sigma(z)) \leq 0.25$. In a network with L layers, backpropagation multiplies gradients across layers: $\prod_{l=1}^{L} \sigma'(z_l) \leq (0.25)^L$. For $L = 10$, $(0.25)^{10} \approx 10^{-6}$, causing earlier layer weights to freeze.
ReLU Resolution: $f(z) = \max(0, z) \implies f'(z) = 1$ for all $z > 0$. Gradients pass through unchanged without diminishing!

Q6. Explain Reinforcement Learning (RL), the Markov Decision Process (MDP), and the Bellman Equation. (10 Marks)

In RL, an agent learns optimal behavior through trial-and-error interactions with an environment by receiving scalar rewards $R(s, a, s')$. An MDP is defined by $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$.
The Bellman Optimality Equation:

$$\mathbf{V}^*(\mathbf{s}) = \max_{\mathbf{a} \in \mathcal{A}} \sum_{\mathbf{s}'} \mathbf{P}(\mathbf{s}' \mid \mathbf{s}, \mathbf{a}) \left[\mathbf{R}(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma \mathbf{V}^*(\mathbf{s}') \right]$$

$$\mathbf{Q}^*(\mathbf{s}, \mathbf{a}) = \sum_{\mathbf{s}'} \mathbf{P}(\mathbf{s}' \mid \mathbf{s}, \mathbf{a}) \left[\mathbf{R}(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma \max_{\mathbf{a}'} \mathbf{Q}^*(\mathbf{s}', \mathbf{a}') \right]$$

Q7. Explain Q-Learning (Off-Policy Temporal Difference Control). (8 Marks)

Q-learning directly learns the optimal action-value function $Q^*(s, a)$ independent of the agent's behavioral exploration policy:

$$\mathbf{Q}(\mathbf{s}, \mathbf{a}) \leftarrow \mathbf{Q}(\mathbf{s}, \mathbf{a}) + \alpha \left[\mathbf{r} + \gamma \max_{\mathbf{a}'} \mathbf{Q}(\mathbf{s}', \mathbf{a}') - \mathbf{Q}(\mathbf{s}, \mathbf{a}) \right]$$

Where $\alpha \in (0, 1]$ is the learning rate, $\gamma \in [0, 1)$ is the discount factor, and $[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$ is the TD Error.

Q8. Explain Ensemble Learning: Bagging (Random Forests) vs. Boosting (AdaBoost/XGBoost). (8 Marks)

• **Bagging (Bootstrap Aggregating):** Trains multiple independent base models (e.g., deep decision trees) in parallel on bootstrap sample subsets with replacement and averages predictions. Primary goal: *Reduces Variance*.
• **Boosting:** Trains base learners (e.g. shallow decision stumps) sequentially, where each new learner focuses on correcting the errors (higher instance weights) of previous models. Primary goal: *Reduces Bias*.

Q9. Explain Support Vector Machines (SVM), Maximum Margin Hyperplane, and the Kernel Trick. (8 Marks)

SVM finds the separating hyperplane $\mathbf{w}^T\mathbf{x} + b = 0$ that maximizes the geometric margin $\frac{2}{\|\mathbf{w}\|}$ between support vectors (quadratic optimization). The **Kernel Trick** $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ (e.g. RBF Gaussian $e^{-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2}$, Polynomial) implicitly projects data into infinite-dimensional Hilbert space to achieve linear separability without explicitly computing high-dimensional coordinates.

Q10. Explain k-Nearest Neighbors (k-NN) Classification and Distance Metrics. (6 Marks)

k-NN is a non-parametric, lazy instance-based learning algorithm that stores all training samples. Given a test query $\mathbf{x}_q$, it calculates distances (Euclidean $d = \sqrt{\sum (x_i - y_i)^2}$, Manhattan $d = \sum |x_i - y_i|$) to all points, identifies the k nearest neighbors, and assigns the majority class label (or distance-weighted vote).

Topic 38.5: Modern Deep Generative Models, Transformers & LLM Architectures

Modern Artificial Intelligence is driven by large-scale deep learning architectures capable of representation learning across raw multimodal sensory data.

Generative Model	Training Objective & Loss Function	Key AI Applications
Variational Autoencoders (VAE)	Maximizes the Evidence Lower Bound (ELBO): $$\mathcal{L}_{\text{ELBO}} = \mathbb{E}_{q_\phi(z\|x)}[\log p_\theta(x \mid z)] - D_{\text{KL}}(q_\phi(z \mid x) \parallel p(z))$$	Latent space interpolation, image generation, anomaly detection.
Generative Adversarial Networks (GAN)	Minimax Two-Player Game: $$\min_G \max_D V(D,G) = \mathbb{E}_x[\log D(x)] + \mathbb{E}_z[\log(1 - D(G(z)))]$$	Photorealistic image synthesis, deepfakes, super-resolution.
Diffusion Models (DDPM)	Forward Markov chain adds Gaussian noise ($q(x_t \mid x_{t-1})$); reverse neural network $\epsilon_\theta(x_t, t)$ learns to denoise step-by-step.	State-of-the-art text-to-image (Stable Diffusion, Midjourney, DALL-E 3).
Autoregressive Transformers (GPT)	Causal Language Modeling (Next-token cross-entropy loss): $$\mathcal{L}_{\text{CLM}} = -\sum_{t=1}^{T} \log P(w_t \mid w_1, \ldots, w_{t-1}; \theta)$$	Large Language Models, code generation, reasoning agents.

🏛️ The Transformer Scaled Dot-Product Self-Attention Mathematical Formulation

Given an input sequence of token embeddings $\mathbf{X} \in \mathbb{R}^{N \times d_{\text{model}}}$, linear projection matrices produce Queries $\mathbf{Q}$, Keys $\mathbf{K}$, and Values $\mathbf{V}$:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_\mathbf{Q}, \quad \mathbf{K} = \mathbf{X}\mathbf{W}_\mathbf{K}, \quad \mathbf{V} = \mathbf{X}\mathbf{W}_\mathbf{V} \qquad (\mathbf{W}_\mathbf{Q}, \mathbf{W}_\mathbf{K} \in \mathbb{R}^{\mathbf{d}_{\text{model}} \times \mathbf{d}_\mathbf{k}},\ \mathbf{W}_\mathbf{V} \in \mathbb{R}^{\mathbf{d}_{\text{model}} \times \mathbf{d}_\mathbf{v}})$$

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\mathbf{T}}{\sqrt{\mathbf{d}_\mathbf{k}}}\right)\mathbf{V}$$

Why Divide by $\sqrt{d_k}$? For large dimensional vectors d_k, the dot products grow large in magnitude, pushing the softmax function into regions with extremely small gradients (vanishing gradients). Scaling by $\frac{1}{\sqrt{d_k}}$ preserves unit variance and stabilizes gradient flow!

Q11. Explain Convolutional Neural Networks (CNNs) and the significance of Parameter Sharing and Equivariance. (8 Marks)

• **Parameter Sharing:** The same kernel filter weights are applied across every spatial receptive field of the input tensor, drastically reducing total learnable weights compared to fully-connected layers.
• **Translational Equivariance:** If an input object shifts by (dx, dy), the output feature map activates at the corresponding shifted location ($(f * g)(x - dx) = f(x - dx) * g$). This allows CNNs to recognize visual features anywhere in an image!

Q12. Explain Reinforcement Learning from Human Feedback (RLHF) for Large Language Models. (8 Marks)

RLHF aligns raw pre-trained LLMs with human values (Helpful, Honest, Harmless) via a 3-stage pipeline:
1. **Supervised Fine-Tuning (SFT):** Fine-tune base LLM on high-quality human prompt-response demonstrations.
2. **Reward Model (RM) Training:** Prompt the SFT model to generate multiple candidate outputs for a prompt; human evaluators rank them. Train a scalar reward model $r_\theta(x, y)$ using Bradley-Terry preference loss: $\mathcal{L} = -\mathbb{E}[\log \sigma(r_\theta(x, y_w) - r_\theta(x, y_l))]$.
3. **PPO Policy Optimization:** Optimize the LLM policy using Proximal Policy Optimization (PPO) to maximize reward $r_\theta(x, y)$ while penalizing KL-divergence $D_{\text{KL}}(\pi_\theta \parallel \pi_{\text{SFT}})$ from the reference model to prevent reward hacking.

Topic 38.6: Advanced Statistical Learning Theory & Optimization Algorithms

Optimization Algorithm	Mathematical Update Rule	Key Advantages in Deep Learning
Stochastic Gradient Descent (SGD)	$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}_i(\theta_t)$	Fast per-iteration computation; noisy gradients escape shallow saddle points.
Momentum SGD	$v_{t+1} = \gamma v_t + \eta \nabla_\theta \mathcal{L}(\theta_t), \quad \theta_{t+1} = \theta_t - v_{t+1}$	Dampens oscillations across high-curvature ravines; accelerates progress along flat valleys.
Adam (Adaptive Moment Estimation)	$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \quad v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$ $\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \ \hat{v}_t = \frac{v_t}{1-\beta_2^t} \implies \theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t}+\epsilon}\hat{m}_t$	Combines momentum with adaptive per-parameter learning rates; the universal standard for Transformers and Deep Neural Networks.

🏛️ **Step-by-Step Solved Problem: Support Vector Machine Dual Formulation & Support Vector Identification**

Given 3 training data points in 2D space: $\mathbf{x}_1 = (1,1)^T, y_1 = -1; \mathbf{x}_2 = (2,0)^T, y_2 = -1; \mathbf{x}_3 = (2,3)^T, y_3 = +1$.

1. SVM Dual Formulation:

$$\max_{\alpha} \sum_{\mathbf{i=1}}^{\mathbf{3}} \alpha_\mathbf{i} - \frac{\mathbf{1}}{\mathbf{2}} \sum_{\mathbf{i=1}}^{\mathbf{3}} \sum_{\mathbf{j=1}}^{\mathbf{3}} \alpha_\mathbf{i}\alpha_\mathbf{j}\mathbf{y_i}\mathbf{y_j}(\mathbf{x_i^T x_j}) \quad \text{subject to } \sum \alpha_\mathbf{i}\mathbf{y_i} = \mathbf{0},\ \alpha_\mathbf{i} \geq \mathbf{0}$$

2. Optimal Solution: $\alpha_1 = 0,\ \alpha_2 = 0.5,\ \alpha_3 = 0.5$.

$$\mathbf{w} = \sum_{\mathbf{i=1}}^{\mathbf{3}} \alpha_\mathbf{i}\mathbf{y_i}\mathbf{x_i} = \mathbf{0.5}(-1)(\mathbf{2},\mathbf{0})^\mathbf{T} + \mathbf{0.5}(+1)(\mathbf{2},\mathbf{3})^\mathbf{T} = (-\mathbf{1},\mathbf{0})^\mathbf{T} + (\mathbf{1},\mathbf{1.5})^\mathbf{T} = (\mathbf{0},\mathbf{1.5})^\mathbf{T}$$

3. Hyperplane Bias b: For support vector $\mathbf{x}_3$: $y_3(\mathbf{w}^T\mathbf{x}_3 + b) = 1 \implies 1(0(2) + 1.5(3) + b) = 1 \implies 4.5 + b = 1 \implies \mathbf{b = -3.5}$.

$$\text{Optimal Decision Boundary: } \mathbf{1.5x_2 - 3.5 = 0} \iff \mathbf{x_2 = 2.333}$$

Topic 38.7: Advanced Deep Reinforcement Learning (DQN & Policy Gradients)

Deep RL Algorithm	Core Loss Function & Update Mechanism	Key Innovations
Deep Q-Networks (DQN)	$\mathcal{L}(\theta) = \mathbb{E}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta)\right)^2\right]$	Experience Replay Buffer (breaks temporal correlation), Target Network θ^- (stabilizes targets).
REINFORCE (Policy Gradient)	$\nabla_\theta J(\theta) = \mathbb{E}\left[\sum_{t=0}^{T} \nabla_\theta \ln \pi_\theta(a_t \mid s_t) G_t\right]$	Direct optimization of parameterized policy $\pi_\theta(a \mid s)$ without action-value discretization.
Actor-Critic (A2C / PPO)	Actor updates $\pi_\theta(a \mid s)$ using Advantage $A(s,a) = Q(s,a) - V(s)$; Critic updates value function $V_\phi(s)$ via MSE loss.	Significantly reduces gradient variance while retaining high sample efficiency.

🏛️ **Step-by-Step Solved Problem: Policy Gradient (REINFORCE) Parameter Update Calculation**

A policy network outputs action probabilities $\pi_\theta(a_1 \mid s) = 0.70, \pi_\theta(a_2 \mid s) = 0.30$ using Softmax over linear logits $z_1 = \theta_1 x, z_2 = \theta_2 x$ with $x = 1.0$. The agent executes action a_1 and receives return $G_t = +5.0$. Learning rate $\eta = 0.1$.

1. Log-Likelihood Gradient for Softmax:

$$\nabla_{\theta_1} \ln \pi_\theta(a_1 \mid s) = x(1 - \pi_\theta(a_1 \mid s)) = 1.0(1 - 0.70) = +\mathbf{0.30}$$

$$\nabla_{\theta_2} \ln \pi_\theta(a_1 \mid s) = -x \cdot \pi_\theta(a_2 \mid s) = -1.0(0.30) = -\mathbf{0.30}$$

2. Policy Parameter Updates:

$$\Delta\theta_1 = \eta \cdot \nabla_{\theta_1} \ln \pi(a_1 \mid s) \cdot G_t = 0.1 \times 0.30 \times 5.0 = +\mathbf{0.15}$$

$$\Delta\theta_2 = \eta \cdot \nabla_{\theta_2} \ln \pi(a_1 \mid s) \cdot G_t = 0.1 \times (-0.30) \times 5.0 = -\mathbf{0.15}$$

Result: θ_1 is boosted by $+0.15$, increasing the probability of choosing successful action a_1 in state s on future episodes!

Topic 38.8: Complete Step-by-Step Solved Problem Bank (Part III)

🏛️ Solved Numerical 4: C4.5 Decision Tree Gain Ratio Calculation

A dataset of 20 samples has target entropy $H(S) = 0.95$. An attribute **Outlook** with values $\{\text{Sunny: } 8, \text{Overcast: } 6, \text{Rain: } 6\}$ achieves Information Gain $\text{Gain}(S, \text{Outlook}) = 0.246$ bits.

1. Calculate Split Information:

$$\text{SplitInfo}(\mathbf{S}, \text{Outlook}) = -\sum_{\mathbf{i=1}}^{\mathbf{3}} \frac{|\mathbf{S_i}|}{|\mathbf{S}|} \log_{\mathbf{2}} \left(\frac{|\mathbf{S_i}|}{|\mathbf{S}|} \right) = -\left[\frac{\mathbf{8}}{\mathbf{20}} \log_{\mathbf{2}} \left(\frac{\mathbf{8}}{\mathbf{20}} \right) + \frac{\mathbf{6}}{\mathbf{20}} \log_{\mathbf{2}} \left(\frac{\mathbf{6}}{\mathbf{20}} \right) + \frac{\mathbf{6}}{\mathbf{20}} \log_{\mathbf{2}} \left(\frac{\mathbf{6}}{\mathbf{20}} \right) \right]$$

$$= -[0.40(-1.3219) + 0.30(-1.737) + 0.30(-1.737)] = -[-0.5288 - 0.5211 - 0.5211] = \mathbf{1.571} \text{ bits}$$

2. Calculate Gain Ratio:

$$\text{GainRatio}(\mathbf{S}, \text{Outlook}) = \frac{\text{Gain}(\mathbf{S}, \text{Outlook})}{\text{SplitInfo}(\mathbf{S}, \text{Outlook})} = \frac{\mathbf{0.246}}{\mathbf{1.571}} = \mathbf{0.1566} = \mathbf{15.66\%}$$

Significance: Gain Ratio penalizes broad multi-valued attributes, preventing overfitting on high-cardinality features (like Customer ID)!

🏛️ Step-by-Step Solved Problem: K-Means Clustering Centroid Convergence Trace

Cluster 4 points $P_1(1,1), P_2(2,1), P_3(4,3), P_4(5,4)$ into $K = 2$ clusters with initial seeds $C_1 = (1,1)$ and $C_2 = (2,1)$:

1. **Distance Computation (Iteration 1):**
   - $P_1(1,1) : d(P_1, C_1) = 0, d(P_1, C_2) = 1 \implies \text{Cluster } 1.$
   - $P_2(2,1) : d(P_2, C_1) = 1, d(P_2, C_2) = 0 \implies \text{Cluster } 2.$
   - $P_3(4,3) : d(P_3, C_1) = \sqrt{3^2 + 2^2} = \sqrt{13} \approx 3.61;\ d(P_3, C_2) = \sqrt{2^2 + 2^2} = \sqrt{8} \approx 2.83 \implies \text{Cluster } 2.$
   - $P_4(5,4) : d(P_4, C_1) = \sqrt{4^2 + 3^2} = 5.0;\ d(P_4, C_2) = \sqrt{3^2 + 3^2} = \sqrt{18} \approx 4.24 \implies \text{Cluster } 2.$
2. **Update Centroids:**

$$C_1^{\text{new}} = (1, 1)$$

$$C_2^{\text{new}} = \left(\frac{2+4+5}{3}, \frac{1+3+4}{3} \right) = \left(\frac{11}{3}, \frac{8}{3} \right) = (\mathbf{3.67}, \mathbf{2.67})$$

3. **Iteration 2:** Assignments remain unchanged. **Converged!**

Q13. Explain AdaBoost Algorithm and Weighted Training Sample Updates. (8 Marks)

AdaBoost trains a sequence of weak learners $h_t(x)$:
1. Initialize sample weights $w_{1,i} = \frac{1}{N}$.
2. In iteration t, compute weighted error $\epsilon_t = \sum_{i: y_i \neq h_t(x_i)} w_{t,i}$.
3. Compute learner weight $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$.
4. Update instance weights: $w_{t+1,i} = \frac{w_{t,i} \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$ (misclassified samples get amplified weights!).
5. Final ensemble: $H(x) = \text{sign} \left(\sum_{t=1}^{T} \alpha_t h_t(x) \right)$.

Topic 38.9: Advanced Statistical Machine Learning & KKT Support Vector Bounds

The Karush-Kuhn-Tucker (KKT) Optimality Conditions for Soft-Margin SVM: For primal objective $\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i$ with constraints $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i,\ \xi_i \geq 0$: 1. **Stationarity:** $\mathbf{w} = \sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i, \quad \sum_{i=1}^{N}\alpha_i y_i = 0, \quad C - \alpha_i - \mu_i = 0.$

2. **Primal Feasibility:** $y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \xi_i \geq 0, \quad \xi_i \geq 0.$

3. **Dual Feasibility:** $0 \leq \alpha_i \leq C, \quad \mu_i \geq 0.$

4. **Complementary Slackness:** $\alpha_i[y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \xi_i] = 0 \quad \text{and} \quad \mu_i\xi_i = (C - \alpha_i)\xi_i = 0.$

🏛️ Step-by-Step Solved Problem: Support Vector Classification Case Analysis

From KKT Complementary Slackness, data points fall into three mathematically distinct regimes based on α_i:

- **Case 1 ($\alpha_i = 0$):** Point is strictly on the correct side of the margin ($y_i(\mathbf{w}^T\mathbf{x}_i + b) > 1, \xi_i = 0$). It has zero influence on the decision boundary!
- **Case 2 ($0 < \alpha_i < C$):** Point is a **Free Support Vector** lying exactly on the margin ($y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1, \xi_i = 0$). These points determine the exact location of the hyperplane bias b!
- **Case 3 ($\alpha_i = C$):** Point is a **Bounded Support Vector** violating the margin ($\xi_i > 0$). It lies inside the margin band or is misclassified.

🏛️ Step-by-Step Solved Problem: Deep Multilayer Perceptron Matrix Form

Consider a 3-layer neural network with layer dimensions $d_0 \to d_1 \to d_2 \to d_3$:

$$\mathbf{a}^{(0)} = \mathbf{x} \in \mathbb{R}^{\mathbf{d_0}}$$

$$\mathbf{z}^{(1)} = \mathbf{W}^{(1)}\mathbf{a}^{(0)} + \mathbf{b}^{(1)} \in \mathbb{R}^{\mathbf{d_1}}, \quad \mathbf{a}^{(1)} = \sigma(\mathbf{z}^{(1)})$$

$$\mathbf{z}^{(2)} = \mathbf{W}^{(2)}\mathbf{a}^{(1)} + \mathbf{b}^{(2)} \in \mathbb{R}^{\mathbf{d_2}}, \quad \mathbf{a}^{(2)} = \sigma(\mathbf{z}^{(2)})$$

$$\mathbf{z}^{(3)} = \mathbf{W}^{(3)}\mathbf{a}^{(2)} + \mathbf{b}^{(3)} \in \mathbb{R}^{\mathbf{d_3}}, \quad \hat{\mathbf{y}} = \text{softmax}(\mathbf{z}^{(3)})$$

Cross-Entropy Loss with Softmax Output has the remarkably clean gradient:

$$\delta^{(3)} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{(3)}} = \hat{\mathbf{y}} - \mathbf{y}$$

$$\delta^{(2)} = (\mathbf{W}^{(3)})^{\mathbf{T}}\delta^{(3)} \odot \sigma'(\mathbf{z}^{(2)}) \qquad \frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(2)}} = \delta^{(2)}(\mathbf{a}^{(1)})^{\mathbf{T}}$$

Topic 38.10: Master University Exam Problem Bank (Part IV)

🏛️ Solved Numerical 5: Neural Network Multiclass Cross-Entropy Loss Derivation

An image classifier outputs raw unnormalized logits for 3 classes: $\mathbf{z} = [2.0, 1.0, 0.1]^T$. True class label is Class 1 ($\mathbf{y} = [1, 0, 0]^T$).

1. Compute Softmax Probabilities:

$$\sum e^{z_i} = e^{2.0} + e^{1.0} + e^{0.1} = 7.389 + 2.718 + 1.105 = \mathbf{11.212}$$

$$p_1 = \frac{7.389}{11.212} = \mathbf{0.6590} \qquad p_2 = \frac{2.718}{11.212} = \mathbf{0.2424} \qquad p_3 = \frac{1.105}{11.212} = \mathbf{0.0986}$$

2. Compute Categorical Cross-Entropy Loss:

$$\mathcal{L} = -\sum_{\mathbf{i=1}}^{\mathbf{3}} \mathbf{y_i} \ln(\mathbf{p_i}) = -\mathbf{1.0} \ln(\mathbf{0.6590}) = -(-\mathbf{0.417}) = \mathbf{0.417} \text{ nats}$$

Q14. Explain Batch Normalization, Layer Normalization, and Dropout Regularization. (8 Marks)

• **Batch Normalization (Ioffe & Szegedy):** Normalizes neuron activations across mini-batch samples ($\hat{x} = \frac{x - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}, y = \gamma \hat{x} + \beta$). Accelerates convergence and stabilizes internal covariate shift.
• **Layer Normalization (Ba, Kiros, Hinton):** Normalizes across all hidden features of a single sample independent of batch size; the universal standard for Transformer attention blocks.
• **Dropout (Srivastava et al.):** Randomly zeros out hidden neuron activations with probability p during training, preventing complex co-adaptations and acting as an ensemble over 2^N sub-networks.