Perceptron and Neural Network

May 1, 2025

1 Multiple Choice Questions (MCQs)

1. **Smart Home Assistant Decision**
   Scenario: A smart home assistant decides whether to remind a user to turn off the lights based on the time of night, whether someone is in the room, and light sensor readings. If the combined signal is strong enough, it sends a reminder.
   Question: What AI concept is this scenario modeling?

   A) Gradient descent

   B) Perceptron decision-making

   C) Loss minimization

   D) Data normalization

   Answer: **B**
   Explanation: The assistant combines multiple input features (time, presence, light) with weights to make a binary decision (remind or not), mimicking a perceptron. A perceptron computes a weighted sum ($z = \sum w_i x_i + b$) and applies a threshold or activation function, similar to a neuron firing. Gradient descent and loss minimization are optimization techniques, and normalization is preprocessing, not the decision process.

2. **Autonomous Drone Detection**
   Scenario: An autonomous drone detects whether an object is a cat using features like pointed ears, fur texture, and whisker presence. Despite seeing fur and whiskers, it sometimes fails to detect the cat.

Question: What parameter might be set too high?

A) Learning rate

B) Bias

C) Threshold

D) Activation

Answer: **C**
Explanation: In a perceptron, the threshold determines when the weighted sum ($z = \sum w_i x_i + b$) triggers activation. If the threshold is too high, even strong feature combinations (fur, whiskers) may not exceed it, preventing detection. Lowering the threshold allows valid inputs to activate the neuron. Learning rate affects training, bias shifts the decision boundary, and "activation" is not a parameter.

3. **Neural Network Neuron Failure**
Scenario: In a neural network trained to recognize handwritten digits, one neuron fails to activate regardless of the input image.
Question: What could be a possible cause?

A) Low learning rate

B) Too many features

C) Dying ReLU problem

D) High batch size

Answer: **C**
Explanation: The Dying ReLU problem occurs when a neuron with a ReLU activation ($f(z) = \max(0, z)$) consistently receives negative weighted sums ($z < 0$), outputting zero and halting gradient updates. This "dead" neuron cannot learn, causing inactivity. Low learning rates slow training, many features don't directly cause this, and batch size affects optimization, not neuron activation.

4. **Robot Vacuum Learning**
Scenario: A robot vacuum decides to avoid obstacles based on IR sensors (front and sides). Initially, it fails to avoid walls, but after training, it learns.

Question: What explains this behavior?

A) Static rule-based programming
B) Weight updates through error correction
C) XOR logic implementation
D) Linear regression

Answer: **B**
Explanation: The vacuum improves by adjusting weights based on errors (difference between predicted and actual actions). This error correction, as in perceptron learning ($\Delta w = \alpha(y_{\text{true}} - y_{\text{pred}})x$), refines the decision boundary. Static rules don't adapt, XOR requires multi-layer networks, and linear regression predicts continuous values, not binary actions.

5. **Child's Dog Recognition**
Scenario: A child learns that a "dog" can be big or small, black or white. As they see more dogs, their recognition improves.
Question: In AI, this resembles which process?

A) Model overfitting
B) Gradient explosion
C) Schema formation and learning from experience
D) Backpropagation failure

Answer: **C**
Explanation: The child builds a mental model (schema) of "dog" by refining feature importance through experience, akin to neural networks updating weights based on data. Overfitting fits noise, gradient explosion destabilizes training, and backpropagation failure prevents learning, none of which match this adaptive process.

6. **Smart Assistant Activation**
Scenario: A smart assistant uses a perceptron to decide whether to respond, with inputs like sound level and wake word match. Even when both are strong, it doesn't activate.
Question: What's likely the issue?

A) Input data is biased

B) Threshold is too low

C) Threshold is too high

D) Too many hidden layers

Answer: **C**
Explanation: A high threshold prevents the perceptron from activating ($z = \sum w_i x_i + b \geq$ threshold) even with strong inputs. Lowering it allows valid signals to trigger a response. Biased data affects training, a low threshold causes over-activation, and hidden layers are irrelevant for a single perceptron.

7. **Perceptron for AND Gate**
Scenario: A student designs a perceptron for an AND gate. Initially, weights and bias are zero. After training, it correctly classifies all inputs.
Question: What helped the perceptron learn?

A) Multiple hidden layers

B) Large batch size

C) Error correction rule

D) Activation decay

Answer: **C**
Explanation: The perceptron learns via the error correction rule ($\Delta w = \alpha(y_{\text{true}} - y_{\text{pred}})x$), adjusting weights and bias to minimize classification errors. AND is linearly separable, so a single-layer perceptron suffices. Hidden layers aren't needed, batch size affects optimization, and activation decay is unrelated.

8. **XOR Problem Limitation**
Scenario: You analyze why a single-layer perceptron cannot solve the XOR problem.
Question: Why can't it solve XOR?

A) It uses a step function

B) It lacks training data

C) XOR is not linearly separable

D) It doesn't use weights

Answer: **C**
Explanation: XOR's input-output pairs (e.g., (0,0)→0, (0,1)→1, (1,0)→1, (1,1)→0) cannot be separated by a single line, as they are not linearly separable. A single-layer perceptron, limited to linear boundaries, fails. Multi-layer networks with non-linear activations solve this. The step function and weights are standard, and training data isn't the issue.

9. **High Learning Rate Risk**
Scenario: You're training a neural network with a high learning rate.
Question: What's a potential risk?

A) Underfitting

B) Slow training

C) Overshooting the minimum

D) Vanishing gradient

Answer: **C**
Explanation: A high learning rate (η) in gradient descent ($\theta := \theta - \eta \nabla J$) can cause large parameter updates, jumping past the loss function's minimum, leading to oscillations or divergence. Underfitting results from insufficient training, slow training from low learning rates, and vanishing gradients from activation functions like sigmoid.

2 Numeric Type Questions (NTQs)

1. **Perceptron Activation**
Scenario: A perceptron receives inputs [1, 0, 1] and weights [0.4, 0.6, 0.5], with a threshold of 0.9.
Question: Will it fire? (1 = yes, 0 = no)
Note: Perceptron fires on z $\geq$ threshold
Answer: **1**
Explanation: Compute the weighted sum: $z = (1 \times 0.4) + (0 \times 0.6) + (1 \times 0.5) = 0.4 + 0 + 0.5 = 0.9$. Since $z = 0.9 \geq 0.9$ (threshold), the

step function outputs 1, indicating the perceptron fires. The threshold determines the activation boundary.

2. **Weight Update**
   Scenario: A perceptron weight $w = 0.2$ is updated with learning rate $\alpha = 0.1$, error $(y_{\text{true}} - y_{\text{pred}}) = 1$, and input $x = 1$.
   Question: What is the new weight?

   Answer: **0.3**
   Explanation: The perceptron learning rule is $\Delta w = \alpha \times \text{error} \times x$. Here, $\Delta w = 0.1 \times 1 \times 1 = 0.1$. The new weight is $w = 0.2 + 0.1 = 0.3$. This update adjusts the weight to reduce the error in future predictions.

3. **Activation Input Calculation**
   Scenario: Given inputs [1, 1], weights [0.6, 0.4], and bias $b = -0.5$, compute the activation input $z = x \cdot w + b$.
   Question: What is z?
   Answer: **0.5**
   Explanation: Compute $z = (1 \times 0.6) + (1 \times 0.4) + (-0.5) = 0.6 + 0.4 - 0.5 = 1.0 - 0.5 = 0.5$. This z is passed to the activation function (e.g., step or sigmoid) to determine the output. The bias shifts the decision boundary.

3 Multiple Select Questions (MSQs)

1. **ReLU Advantages**
   Scenario: You analyze why ReLU is used in deep neural networks.
   Question: What are reasons why ReLU is commonly used?

   A) It helps avoid vanishing gradients

   B) It is computationally efficient

   C) It always outputs 0

   D) It introduces nonlinearity

   Answers: **A, B, D**
   Explanation:

- A) ReLU ($f(z) = \max(0, z)$) passes positive gradients unchanged, avoiding vanishing gradients unlike sigmoid, true.
- B) ReLU is simple (max operation), making it computationally efficient, true.
- C) ReLU outputs z for $z > 0$, not always 0, false.
- D) ReLU introduces nonlinearity, enabling complex pattern learning, true.

2. **Perceptron Learning Improvements**
Scenario: You design a perceptron to learn a classification task.
Question: Which changes can help it learn correctly?

   A) Adjusting weights based on error

   B) Using a smaller learning rate

   C) Applying the error correction rule

   D) Fixing the number of hidden layers

   Answers: **A, B, C**
   Explanation:
   - A) Weight adjustments via error signals are core to perceptron learning, true.
   - B) A smaller learning rate ensures stable updates, aiding convergence, true.
   - C) The error correction rule ($\Delta w = \alpha(y_{\text{true}} - y_{\text{pred}})x$) drives learning, true.
   - D) Single-layer perceptrons have no hidden layers, so fixing them isn't applicable.

3. **Activation Function Properties**
Scenario: You evaluate different activation functions for neural networks.
Question: Which statements are true about activation functions?

   A) Step function is useful for binary classification but not trainable via gradient descent

   B) Sigmoid outputs are zero-centered

   C) Tanh is zero-centered and helps with gradient flow

D) ReLU may suffer from dying neurons

Answers: **A, C, D**
Explanation:
- A) The step function outputs 0 or 1, ideal for binary classification but non-differentiable, preventing gradient descent, true.
- B) Sigmoid outputs [0, 1], not zero-centered (unlike tanh), false.
- C) Tanh outputs [-1, 1], zero-centered, aiding gradient flow, true.
- D) ReLU can produce dead neurons if $z < 0$ persistently, true.