# 18-661: Introduction to ML for Engineers

## Practice Questions for Mini-Exam 2

Spring 2025

ECE – Carnegie Mellon University

# Decision Trees

**Question:** Decision trees can handle both classification and regression tasks by splitting data into branches based on feature values.

☐ True

☐ False

# Decision Trees

True. Decision trees are capable of handling both discrete (classification) and continuous (regression) target variables. In classification, decision trees predict class labels, while in regression, they predict continuous values by computing the mean value of samples within a leaf node.

# Decision Trees

**Question:** Which of the following datasets has the highest entropy?

☐ A dataset with 2 classes and 100 examples where 50 examples are from the first class and 50 examples are from the second class

☐ A dataset with 2 classes and 100 examples where 99 examples are from the first class and 1 example is from the second class

☐ A dataset with 2 classes and 100 examples where 60 examples are from the first class and 40 examples are from the second class

☐ A dataset with 3 classes and 100 examples where 50 examples are from the first class, 25 examples are from the second class, and 25 examples are from the third class

# Decision Trees

The dataset with a 50-50 class distribution maximizes entropy:

$$H(X) = -\sum_i p_i \log_2 p_i$$

For this case, $H = -(0.5 \log_2 0.5 + 0.5 \log_2 0.5) = 1$, which is the highest.

# Neural Networks

**Question:** A neural network with a single hidden layer of sufficient width and a non-linear activation function can approximate any continuous function up to some target accuracy.

- □ True
- □ False

# Neural Networks

True. This is a direct consequence of the Universal Approximation Theorem, which states that a feedforward neural network with a single hidden layer containing a finite number of neurons can approximate any continuous function on a compact subset of $\mathbb{R}^n$ given a non-linear activation function.

# Neural Networks

**Question:** Which of the following statements are true? **More than one option can be true.**

□ SGD with a sufficiently small learning rate will always converge to the same minimum regardless of the initialization of weights.

□ While the Hessian matrix can provide insights into curvature of the loss surface, it is rarely used directly in gradient descent due to its high computational cost in deep learning.

□ Batch normalization helps improve generalization and reduce overfitting by reducing internal covariate shift.

□ The Adam optimizer can sometimes lead to poorer generalization performance compared to standard SGD as it may encourage rapid convergence to sharp minima rather than flat minima.

# Neural Networks

The correct answers are B, C, and D.

□ The Hessian matrix describes curvature but is computationally expensive for deep learning.

□ Batch normalization helps reduce internal covariate shift, leading to improved generalization.

□ Adam can lead to poorer generalization compared to SGD as it may find sharp rather than flat minima.

# Neural Networks

**Question:** Which of the following functions cannot be represented by a one-layer neural network with a bias term? (More than one option can be correct)

- □ AND function ($x_1 \wedge x_2$)
- □ OR function ($x_1 \vee x_2$)
- □ XOR function ($x_1 \oplus x_2$)
- □ All of the above

# Neural Networks

XOR function. The XOR function is not linearly separable, meaning that a single-layer perceptron cannot represent it. A neural network requires at least one hidden layer to model XOR correctly.

# Optimization for NNs

**Question:** The $\tanh(x)$ activation function suffers from vanishing gradients or saturated gradients.

□ True

□ False

# Optimization for NNs

True. The $\tanh(x)$ function squashes input values into the range $[-1, 1]$. When input values are too large or too small, the derivative of $\tanh(x)$ approaches zero, leading to the vanishing gradient problem. This slows down the training of deep neural networks.

# Optimization for NNs

**Question:** The Adam optimizer empirically achieves faster convergence compared to standard gradient descent by employing per-parameter adaptive learning rates and historical gradient information.

□ True

□ False

# Optimization for NNs

True. Adam (Adaptive Moment Estimation) combines momentum and adaptive learning rates for different parameters, allowing it to converge faster than standard SGD in many deep learning applications.

# Optimization for NN

**Question:** Which of the following statements about optimization methods for training neural networks is true. Only one option is true.

□ In the momentum SGD update in iteration $t$, the gradient from iteration $t-1$ gets multiplied by weight $\gamma^{t-1}$, the gradient from iteration $t-2$ gets multiplied by $\gamma^{t-2}$, and so on.

□ In the momentum SGD update in iteration $t$, the gradient from iteration $t-1$ gets multiplied by weight $\gamma$, the gradient from iteration $t-2$ gets multiplied by $\gamma^2$, and so on.

□ Setting a larger momentum term $\gamma$ will lead to slower convergence.

□ Setting the $\gamma = 1$ reduces the algorithm to standard SGD

# Optimization for NN

The correct answer is:

□ In momentum SGD, the gradient from iteration $t-1$ gets multiplied by $\gamma$, from $t-2$ by $\gamma^2$, and so on. This formulation helps in accelerating convergence by incorporating past gradients.

# RNNs

**Question:** Recurrent neural networks (RNNs) are often used to make predictions on a sequence of data. Which of the following characteristics of RNN architectures allows them to make such sequential predictions? More than one option can be true.

□ RNNs may be autoregressive, i.e., they can take previous outputs as inputs to the next prediction.

□ RNNs learn hidden states that are used as inputs to the next prediction.

□ RNNs concatenate inputs from multiple past timeslots as input to the current prediction.

□ None of the above.

# RNNs

The correct answers are A and B.

□ RNNs can take previous outputs as inputs for future predictions.

□ RNNs maintain hidden states, which store information from past inputs.

# Ensemble Methods

**Question:** Boosting is a method used to reduce the bias of a model.

□ True

□ False

# Ensemble Methods

True. Boosting works by sequentially training weak classifiers on modified datasets, where misclassified examples are given higher weights. This reduces bias by improving weak classifiers and creating a strong classifier through aggregation.

# Ensemble Methods

**Question:** Which of the following statements about Adaboost is **NOT** true?

□ AdaBoost assigns weights to both base classifiers and training data points. Base classifiers that make fewer misclassifications get higher weights than those that make more misclassifications, and data points that are misclassified less often get higher weights than those misclassified more.

□ Adaboost iteratively updates the weights assigned to data points. At iteration step $t$ of Adaboost training, the updated weight of a misclassified data point will always be larger if the current base classifier $h_t$ made 5 misclassifications than if it made 20 misclassifications.

□ Base learners with higher classification error contribute less to the final ensemble output.

□ None of the above.

# Ensemble Methods

**A is NOT true.** While it's true that AdaBoost assigns weights to both training data points and base classifiers, the statement about data point weights is incorrect. Data points that are misclassified more often are assigned higher weights to focus learning on harder examples.

**B is also NOT true.** It misleadingly assumes that the number of misclassifications directly determines the weight update. In reality, the update depends on the base classifier's *weighted error rate*, not the raw number of misclassification. A classifier with 5 misclassifications could have a higher error than one with 20 misclassifications if the misclassified points have larger weights. Therefore, we cannot conclude the updated weight is always larger based solely on the misclassification count.

**C is true.** In AdaBoost, the contribution of each base learner to the final model is proportional to its performance: classifiers with higher error get lower weights (smaller $\alpha_t$ values).