Practice Questions for the ECE 661 Spring 2025 Final Exam

Introduction to Machine Learning for Engineers
Prof. Gauri Joshi and Prof. Carlee Joe-Wong

The number of practice questions included in this document may not be representative of the length of the exam. These are solely intended for you to gain more experience in answering questions similar to what you will see on the exam.

1 True or False

Problem 1: *[1 points]* The reason why Naïve Bayes is naïve is that it assumes a prior on the label.
○ True

○ False

Solution: False. Naïve Bayes is called naïve because it assumes conditional feature independence.

Problem 2: *[1 points]* Training a logistic regression model for binary classification is equivalent to training a two-layer neural network with one linear layer and one softmax layer.
○ True

○ False

Solution: True. Logistic regression models the probability of a data point belonging to each class as the sigmoid function of a linear combination of the input features, which is exactly what a linear layer plus softmax layer does for binary classification.

Problem 3: *[1 points]* In decision tree algorithms, including attributes that have no direct effect on the target variable can be beneficial for the robustness of the learned model because they add complexity that can capture unforeseen patterns.
○ True

○ False

Solution: False. Including attributes that do not affect the target variable tends to add unnecessary complexity to the model, which can lead to over-fitting by capturing noise rather than meaningful patterns. The robustness of a decision tree model is improved by including relevant attributes that contribute to information gain, not by the indiscriminate addition of complexity.

Problem 4: *[1 points]* The K-means clustering algorithm is guaranteed to converge if it does not encounter any ties in the distance of a point from the cluster centers.
○ True

○ False

Solution: True.

Problem 5: *[1 points]* Suppose we want to use a Gaussian Mixture Model (GMM) to solve a clustering problem and that we decide to use the EM (Expectation-Maximization) algorithm to solve for the GMM parameters. It is guaranteed that the solution of the EM algorithm converges to parameters that maximize the GMM likelihood function.

○ True

○ False

Solution: False. EM is only guaranteed to converge to a local, not global, maximum.

Problem 6: *[1 points]* Consider using synchronous distributed SGD (stochastic gradient descent) to train a linear regression model, where the data is shuffled and uniformly distributed across m workers. The expected error (i.e., ℓ_2 loss) after t iterations of synchronous distributed SGD with m workers and a batch size of b per worker is equivalent to the expected error of single-worker mini-batch SGD with a batch size of mb after t iterations.

○ True

○ False

Solution: True. In both scenarios, we are effectively performing gradient descent on mb data points chosen uniformly at random from the entire training dataset.

Problem 7: *[1 points]* A Markov Decision Process (MDP) becomes equivalent to the multi-armed bandit formulation when we set the discount factor to $\gamma = 0$, there is a single state without any state transitions, and the reward function is only action dependent (i.e. the reward is given by $r(a)$ where a represents the action).

○ True

○ False

Solution: True.

2 Multiple Choice

Problem 8: *[2 points]* You are building a k-nearest neighbors model to predict students' final grades based on their study hours, attendance rate, midterm grades, and assignment grades. From earlier knowledge, you know that assignment grades are the most important indicator. After training your model with a certain dataset, you realize that it performs well on the training set but poorly on the test set. Which of the following steps is most likely to improve the generalization performance of your k-nearest neighbors model?

⬡ Increase the value of k.

⬡ Decrease the value of k.

⬡ Remove the assignment grades feature.

⬡ Replace the Euclidean distance (L2-norm) with the Manhattan distance (L1-norm).

Solution: A. Increasing k will decrease the variance, helping to prevent overfitting to the training dataset.

Problem 9: *[2 points]* Suppose you train a neural network on a training dataset $\mathcal{D}_T$, using a validation dataset $\mathcal{D}_V$ to determine the optimal value of the model hyperparameters. However, you observe that while the trained neural network has low training error, it has high testing error on the test dataset. Which of the following mistakes in your training procedure might explain these results?

⬡ You accidentally omitted half of the training data.

⬡ You accidentally used the same training and validation datasets (i.e., $\mathcal{D}_V$ was mistakenly set to be the same as $\mathcal{D}_T$).

⬡ You used a neural network with too many layers.

⬡ All of the above.

Solution: D. Omitting half of the training data makes the model more likely to overfit to the training dataset. Using the training data as validation also makes the model prone to overfitting. Using a neural network that is too deep can also lead to overfitting.

Problem 10: *[2 points]* Suppose you are given a dataset with two classes and are asked to train a classification model on this dataset. You are told to expect that the classification boundary will be a linear one, so you are trying to decide between using logistic regression or support vector machines (SVMs). Which of the following characteristics of the dataset would indicate that SVM will train a more accurate model using fewer computational resources?

○ Your dataset has a very high number of features.

○ The data points corresponding to the two classes are linearly separable.

○ One of the features has a range of values from -1000000 to 1000000, but another has a range of values from -1 to 1.

○ Your dataset has a very high number of data points.

Solution: A. Through duality, SVM scales better to a large number of features.

Problem 11: *[2 points]* You want to perform K-means clustering on a dataset of N social network users to identify groups of similar users. You are not sure what is the right value of K, so you decide to plot the final value of the cost function J when using K clusters versus K for all K in the range 1 to N. Recall that

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|^2.$$

where $\boldsymbol{x}_n$ is the feature vector of the n-th sample, $\boldsymbol{\mu}_k$ is the k-th cluster mean and r_{nk} is 1 if sample n is assigned to cluster k and 0 otherwise.
Which of the following statements is TRUE?

○ J decreases or stays the same as K increases. The optimal K is the smallest possible value.

○ J first decreases and then increases with K. The optimal K is the one that minimizes J.

○ J first increases and then decreases with K. The optimal K is the one that maximizes J.

○ None of the above.

Solution: D. J decreases or stays the same as K increases, but the best K is at the 'elbow' of the function.

Problem 12: *[2 points]*   Assume you are given a sample training dataset which consists of real numbers $(x_i, y_i) \in \mathbb{R}^2$, for $i = 1, 2, ..., N$. To understand the data you plotted those points with x values corresponding to the horizontal axis and y values corresponding to the vertical axis, as seen in Figure 1. You will fit a curve to those points using a nonlinear regression model. Which of the following basis functions would result in a model with the least training error?
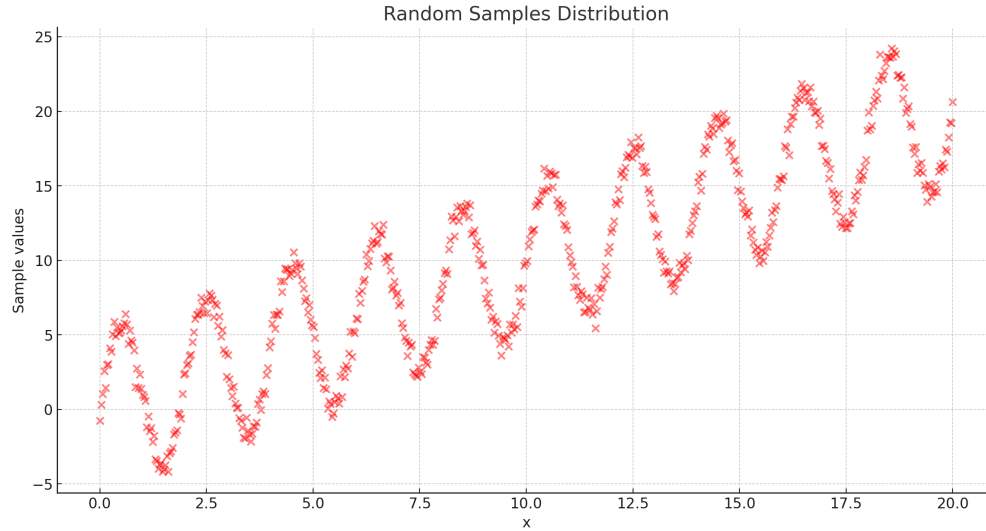

Figure 1: Random samples plot

○ $\phi(x) = \begin{bmatrix} x & \cos(2\pi x) \end{bmatrix}^T$

○ $\phi(x) = \begin{bmatrix} 1 & \cos(\pi x) \end{bmatrix}^T$

○ $\phi(x) = \begin{bmatrix} x & \sin(\pi x) \end{bmatrix}^T$

○ $\phi(x) = \begin{bmatrix} 1 & \sin(2\pi x) \end{bmatrix}^T$

Solution: C: $\phi(x) = \begin{bmatrix} x & \sin(\pi x) \end{bmatrix}^T$. Note that the period of the oscillations is approximately 2, which corresponds to either $\sin(\pi x)$ or $\cos(\pi x)$. We also want to capture the linear increase in y values, so we need an x in the basis function.

Problem 13: *[2 points]* Support vector machines (SVMs) are a historically popular classification algorithm due to their ability to learn nonlinear decision boundaries. Consider the dataset shown in the figure below. Each data point has two-dimensional input $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, and the labels are indicated by blue circles (label 0) and red squares (label 1).
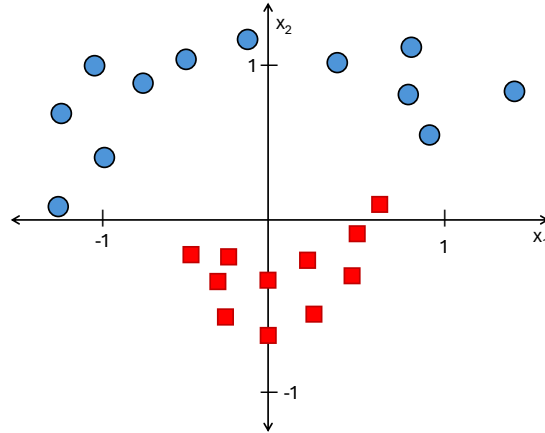
Which of the following nonlinear basis functions $\phi(\mathbf{x})$, when used to learn a SVM model, will NOT learn a decision boundary that perfectly separates the two classes?

○ $\phi(\mathbf{x}) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

○ $\phi(\mathbf{x}) = \begin{bmatrix} x_1^2 \\ x_2^2 \\ 1 \end{bmatrix}$

○ $\phi(\mathbf{x}) = \begin{bmatrix} x_1 x_2 \\ x_1^2 x_2^2 \end{bmatrix}$

○ This is a trick question. Learning nonlinear decision boundaries in SVM requires specifying a kernel function. We cannot specify a nonlinear basis function.

Solution: C. A and B will learn a line and circle, which both separate the dataset. D is incorrect since, while it is often convenient to specify a kernel instead of a basis function, it is not required.

6

Problem 14: *[2 points]* Consider a distributed SGD implementation with m workers with a mini-batch size b per worker. Suppose that each worker takes exponential time rate μ (and mean $1/\mu$) to finish its mini-batch gradient computation, and that these times are independent across workers and mini-batches. What are the expected times taken by synchronous and asynchronous SGD to finish processing N mini-batches of data?

[Hint:] The expected minimum and maximum of n independent exponential random variables with rate μ are $\frac{1}{n\mu}$ and $\frac{\log n}{\mu}$ respectively.

○ Sync SGD $= \frac{N \log m}{\mu}$, Async SGD $= \frac{N}{m\mu}$

○ Sync SGD $= \frac{N\mu}{\log m}$, Async SGD $= \frac{Nm}{\mu}$

○ Sync SGD $= \frac{N \log m}{\mu}$, Async SGD $= \frac{Nm}{\mu}$

○ Sync SGD $= \frac{Nm \log m}{\mu}$, Async SGD $= \frac{N}{m\mu}$

Solution: A. The time taken to complete one iteration of synchronous SGD in which we process m mini-batches is $\log m/\mu$, and thus the time to process N mini-batches is $N \log m/\mu$. The time taken to complete one iteration of asynchronous SGD in which we process 1 mini-batch is $1/m\mu$, and thus the time to process N mini-batches is $N/m\mu$.

3   Descriptive

Problem 15: *[5 points]*   Consider a dataset containing the following three 2-dimensional points along with their corresponding labels (y-values) as shown in Figure 2:

- Point A: $(x_1 = 1, x_2 = 1)$, $y = 1$

- Point B: $(x_1 = 4, x_2 = 4)$, $y = 2$

- Point C: $(x_1 = 7, x_2 = 1)$, $y = 3$

Suppose that you perform k-Nearest Neighbors regression for predicting the y-value, $\hat{y}$, with $k = 1$ and with the Euclidean distance (i.e., ℓ_2 distance) metric. For any new point (x_1, x_2) we want to find the decision regions for all possible $\hat{y}$ in terms of x_1 and x_2.

When answering the questions below, **show all your work** and explicitly state each decision region for each possible $\hat{y}$ (there are three possible values, $\hat{y} = 1, 2, 3$) using x_1 and x_2 values and '$=, <, >, +, -$'. For example, the decision rule for declaring $\hat{y} = 1$ can be of the form $\{ax_1 + bx_2 \geq c \text{ and } dx_1 + ex_2 \geq f\}$ for some scalars a, b, c, d, e, f. You can break ties arbitrarily.

a. *[2 points]* Draw the decision boundaries on Figure 2.

b. *[1 points]* Specify the decision region in which we predict label $\hat{y} = 1$ for a point (x_1, x_2).

c. *[1 points]* Specify the decision region in which we predict label $\hat{y} = 2$ for a point (x_1, x_2).

d. *[1 points]* Specify the decision region in which we predict label $\hat{y} = 3$ for a point (x_1, x_2).

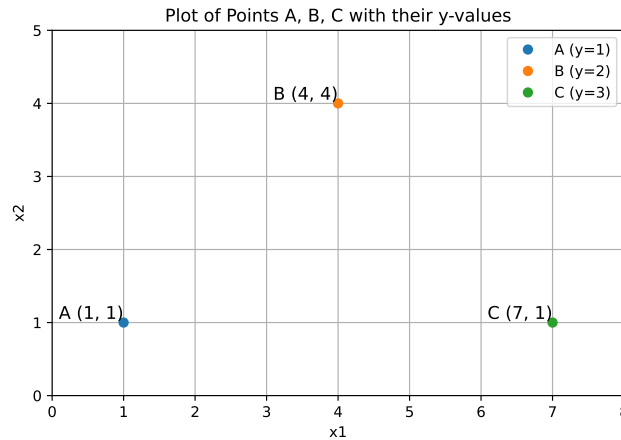Figure 2: Dataset for Problem 17.

Solution: For any test point (x_1, x_2),

(1.) If $x_1 < 4$, the point is closer to A than C.

(2.) If $x_1 + x_2 > 5$, the point is closer to B than A.

(3.) If $x_1 - x_2 < 3$, the point is closer to B than C.

It results in Voronoi tessellation in Figure 3.
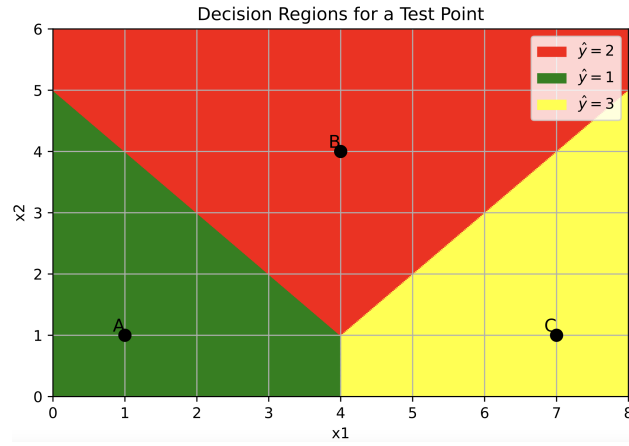Then, we can have the following decision rules

8

Figure 3: Problem 5 Solution Figure

· $\hat{y} = 2$ if $\{x_1 + x_2 > 5$ and $x_1 - x_2 < 3\}$ by using (2.) and (3.)

· $\hat{y} = 1$ if $\{x_1 < 4$ and $x_1 + x_2 \leq 5\}$ by using (1.) and (2.)

· $\hat{y} = 3$ if $\{x_1 \geq 4$ and $x_1 - x_2 \geq 3\}$ by using (2.) and (3.)

Problem 16: *[5 points]* Suppose you are given a dataset $\mathcal{D}$ of N data points, where N is an even number and $\mathcal{D} = \left\{ (-1)^j \lceil \frac{j}{2} \rceil \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}, \ j = 1, 2, \ldots, N \right\}$. Each data point therefore has three input features.

Note that the notation $\lceil \alpha \rceil$ refers to the smallest integer greater than or equal to α. For example, $\lceil \frac{1}{2} \rceil = 1$.

a. *[2 points]* Let $N = 2$, so that $\mathcal{D} = \left\{ \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix} \right\}$. Suppose you run PCA (principal component analysis) to find the top principal component on this dataset. What principal component do you find? Show your work or explain your answer.

Solution: The top principal component is $\frac{\sqrt{2}}{2} \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}$. One can see this directly from the dataset, as this principal component leads to zero reconstruction loss. Alternatively, one can see that all variance in this dataset occurs along this vector, which therefore should be the principal component. Full points should be given even if the principal component is not normalized to magnitude 1.

Mathematically, our data is zero-centered, so the covariance matrix $X^T X = \begin{bmatrix} 2 & -2 & 0 \\ -2 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$; it is easy to see that the only nonzero eigenvalue of this matrix is 2, with associated eigenvector $\begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}$.

b. *[2 points]* Now suppose that $N = 4$, and that you again run PCA to find the top principal component on $\mathcal{D}$. What principal component do you find? Is it the same as the one you found in part (a) for $N = 2$? Briefly explain your answer.

Solution: We obtain the same answer as in part (a). All variation in our dataset still occurs along the same vector $\begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}$, as each vector is a multiple of this vector, and therefore the same reasoning as in part (a) holds.

c. *[1 points]* You are now asked to find the top *two* principal components for the dataset in part (b). Will this lead to a decrease in the average reconstruction error over the data points in $\mathcal{D}$, compared to the reconstruction error with the single principal component you found in part (b)? Briefly explain your answer.

Recall that the *reconstruction error* of a data point $\mathbf{x}$ is the Euclidean distance between $\mathbf{x}$ and its reconstructed data point $\tilde{\mathbf{x}}$, where $\tilde{\mathbf{x}}$ is the representation of $\mathbf{x}$ in terms of the principal components.

Solution: No, it will not decrease the reconstruction error, which was already 0 from part (b).

Problem 17: *[6 points]*   Suppose we have a neural network for performing nonlinear regression which consists of three linear layers, where all but the last layer have a sigmoid activation and the last layer uses a linear activation. Letting $\mathbf{x}_0 \in \mathbb{R}^{d_0}$ denote the d_0-dimensional input to the network and $\hat{y}$ the scalar output, we can write

$$
\begin{aligned}
\mathbf{x}_1 &= A_1\mathbf{x}_0 + b_1\mathbf{1}, \quad \tilde{\mathbf{x}}_1 = \sigma(\mathbf{x}_1) \\
\mathbf{x}_2 &= A_2\tilde{\mathbf{x}}_1 + b_2\mathbf{1}, \quad \tilde{\mathbf{x}}_2 = \sigma(\mathbf{x}_2) \\
\hat{y} = x_3 &= A_3\tilde{\mathbf{x}}_2 + b_3.
\end{aligned}
$$

Here our model parameters are the bias terms $b_1, b_2, b_3 \in \mathbb{R}$ and the matrices $A_1 \in \mathbb{R}^{d_1 \times d_0}$, $A_2 \in \mathbb{R}^{d_2 \times d_1}$, $A_3 \in \mathbb{R}^{1 \times d_2}$, where the hidden states $\mathbf{x}_1 \in \mathbb{R}^{d_1}$ and $\mathbf{x}_2 \in \mathbb{R}^{d_2}$. We use $\mathbf{1}$ to denote a vector of all ones; thus, the bias term in each layer is added to each entry of the output vector of this layer.

Recall that the sigmoid function $\sigma(a) = (1 + e^{-a})^{-1}$ and that $\sigma'(a) = \sigma(a)(1 - \sigma(a))$; when we write $\sigma(\mathbf{a})$ for a vector $\mathbf{a}$, we apply the sigmoid function to each element of $\mathbf{a}$. For example, $\sigma\left(\begin{bmatrix} a_1 \\ a_2 \end{bmatrix}\right) = \begin{bmatrix} \sigma(a_1) \\ \sigma(a_2) \end{bmatrix}$.

Normally, we optimize the parameters of our neural network $A_1, A_2, A_3, b_1, b_2, b_3$ in order to minimize a given loss function. Now, suppose we are instead trying to attack the neural network in order to cause it to obtain the *wrong* answer by choosing the input $\mathbf{x}_0$ that *maximizes* the loss.

a. *[3 points]* Suppose we use the l_2 loss function $\mathcal{L}(\hat{y}, y^*) = ||\hat{y} - y^*||_2^2$, where y^* is the ground truth label. Express $\frac{\partial \mathcal{L}}{\partial \mathbf{x}_0}$ in terms of y^*, parameters $A_1, A_2, A_3, b_1, b_2, b_3$, and intermediate values $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, x_3, \tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2$. Remember to show your work.

Solution: We first write the Jacobians of each "block" in our network:

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial x_3} &= \frac{\partial}{\partial \hat{y}}(x_3 - y^*)(x_3 - y^*) = 2(x_3 - y^*) \\
\frac{\partial \mathbf{x}_i}{\partial \tilde{\mathbf{x}}_{i-1}} &= \frac{\partial}{\partial \tilde{\mathbf{x}}_{i-1}}\left(A_i\tilde{\mathbf{x}}_{i-1} + b_i\right) = A_i \\
\frac{\partial \tilde{\mathbf{x}}_i}{\partial \mathbf{x}_i} &= \frac{\partial}{\partial \mathbf{x}_i}\sigma(\mathbf{x}_i) = \mathrm{diag}(\sigma(\mathbf{x}_i)(1 - \sigma(\mathbf{x}_i)))
\end{aligned}
$$

Here, we use the notation that $\mathrm{diag}(\sigma(\mathbf{x}_i)(1 - \sigma(\mathbf{x}_i)))$ is a square diagonal matrix where the diagonal elements are $\sigma(\mathbf{x}_i)(1 - \sigma(\mathbf{x}_i))$ applied element-wise to $\mathbf{x}_i$.

Putting it all together, we have

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \mathbf{x}_0} &= \frac{\partial \mathcal{L}}{\partial x_3}\frac{\partial x_3}{\partial \tilde{\mathbf{x}}_2}\frac{\partial \tilde{\mathbf{x}}_2}{\partial \mathbf{x}_2}\frac{\partial \mathbf{x}_2}{\partial \tilde{\mathbf{x}}_1}\frac{\partial \tilde{\mathbf{x}}_1}{\partial \mathbf{x}_1}\frac{\partial \mathbf{x}_1}{\partial \mathbf{x}_0} \\
&= 2(x_3 - y^*)A_3\mathrm{diag}(\sigma(\mathbf{x}_2)(1 - \sigma(\mathbf{x}_2)))A_2\mathrm{diag}(\sigma(\mathbf{x}_1)(1 - \sigma(\mathbf{x}_1)))A_1
\end{aligned}
$$

b. *[1 points]* Use your answer to part (a) to write the expression for a gradient update with a learning rate of λ for $\mathbf{x}_0$ that will *maximize* the loss $\mathcal{L}(\hat{y}, y^*) = ||\hat{y} - y^*||_2^2$ when performed repeatedly.

Solution: Normally, we perform gradient *descent* by subtracting the learning rate times the gradient. In this case, we are trying to maximize the loss, so must instead perform gradient *ascent*:

$$
\mathbf{x}_0 \leftarrow \mathbf{x}_0 + \lambda\frac{\partial \mathcal{L}}{\partial \mathbf{x}_0}
$$

11

c. *[2 points]* Assuming a constant learning rate that is appropriately small, is the loss value associated with the gradient-based optimization procedure you described in part (b) guaranteed to converge? Why or why not?

Solution: The loss is not guaranteed to converge. Normally, we optimize a loss metric which is bounded in the direction of optimization, e.g. minimizing the l_2 error with the constraint that the l_2 error cannot be negative. So long as the loss decreases monotonically, this guarantees convergence eventually.

However, in this case, the l_2 loss is not bounded above. As such, it can, if not constrained in some way, diverge to $+\infty$. This is commonly done by restricting $||x' - x||_p < \varepsilon$ for some p-norm and distance ε; once the gradient ascent algorithm above is modified to take this into account, this results in a procedure commonly known as *projected gradient descent.*

Problem 18: *[6 points]* Recall that ensemble methods aim to train multiple decision trees and combine their results to produce a classifier. In this problem, we will apply ensemble methods to the dataset of six points shown in the figure below. Each dataset has a two-dimensional input $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, with coordinates shown as (x_1, x_2) in the figure. Each data point's label is indicated by a blue circle or red square.
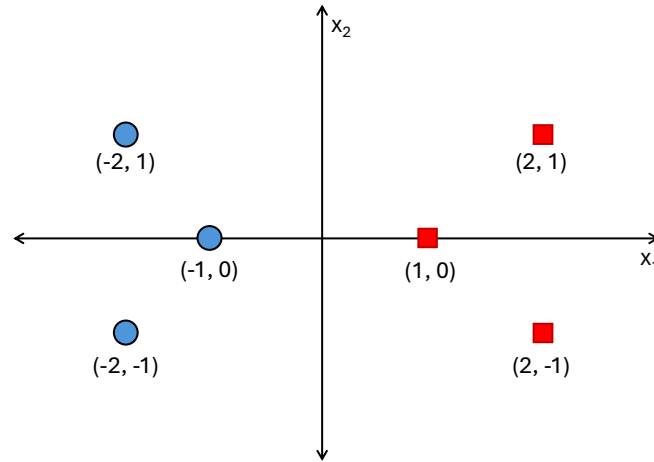

a. *[1.5 points]* First, we apply random forests to this dataset. We use $n = 2$ data points for each tree and train $m = 5$ trees. Find an expression for the probability that $\mathbf{x} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$ is used to train the fifth tree. Remember to show your work. You may assume that data points are sampled with replacement. If it is not possible to derive this probability without knowledge of the first four trees, explain why not.

Solution: Each data point has an equal probability of being chosen to train each tree in the ensemble. We choose $n = 2$ out of $N = 6$ data points with replacement, so the probability of being chosen is $1 - \left(\frac{N-1}{N}\right)^n = 1 - \left(\frac{5}{6}\right)^2 = \frac{11}{36}$.

We also accepted answers that assumed data points are sampled with replacement across trees but without replacement for each tree. In this case, five pairs of data points include the point $\begin{bmatrix} 2 \\ 1 \end{bmatrix}$, out of a total of $15 = \binom{6}{2}$ possible pairs, for a probability of $\frac{1}{3}$.

b. *[1.5 points]* Now suppose that you try AdaBoost on this dataset, and you decide to use decision stumps (one-layer decision trees) as your base classifiers. Find the first decision stump that is created by AdaBoost. You can provide your answer by drawing the line in the figure below, or by providing the equation of the decision boundary.
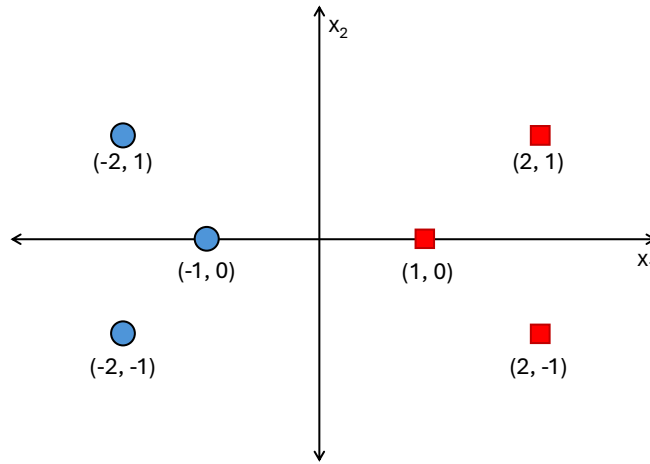
Solution: Any vertical line $x_1 = \alpha$ for $\alpha \in (-1, 1)$ is correct.

c. *[1 points]* Continuing from the decision stump found in part (b), you decide to run AdaBoost for four more iterations, so that your final ensemble has five decision stumps. Find the probability that the data point $\begin{bmatrix} 2 \\ 1 \end{bmatrix}$ has higher weight than the data point $\begin{bmatrix} -2 \\ 1 \end{bmatrix}$ in training the fifth decision stump in AdaBoost. Remember to show your work. If it is not possible to derive this probability without computing the second, third, and fourth trees, explain why not.

Solution: Both data points will have equal weight for all classifiers in the ensemble, since from part (b) the first classifier has zero training error.

In general, this statement may not be true, as the weight of each data point $\mathbf{x}$ depends on the value $h_t(\mathbf{x})$ of each base classifier. However, since we are using decision stumps, the base classifier value is binary, so each correctly classified data point has the same classifier value $h_t(\mathbf{x})$.

d. *[1 points]* How many data points in this dataset will be misclassified by the resulting ensemble of five trees from part (c)? Explain your answer or show your work.

Solution: All data points will be correctly classified. Since all points are correctly classified by the first tree, all subsequent trees will also correctly classify each data point.

e. *[1 points]* Considering again the ensemble learned in part (c), will the mis-classification rate on a test dataset be larger or smaller than the mis-classification rate on the training dataset? Briefly explain your answer. If it is not possible to know the answer without information on the testing dataset, explain why not.

Solution: Since the mis-classification rate on the training dataset is zero, the mis-classification rate on the test dataset must be either equal to or larger than the rate on the training dataset.