CSCI 567 Fall 2022, Quiz 2

Instructor: Vatsal Sharan

Time limit: 2 hour and 40 minutes

Problem	1	2	3	4	5	6	7	Total
Max	8	8	15	12	17	10	30	100

Please read the following instructions carefully:

- Please go ahead and fill in your name and Student ID in the space above.
- The exam has 20 pages (including this cover page), numbered 1-20. Once you are permitted to open your exam (and not before), you should make sure you are not missing any pages.
- Answers should be concise and written down legibly. You must answer each question on the space provided below the question. You likely won't need all the provided space to answer most of the questions.
- There are 4 pages provided for scratch work at the end. These pages will not be graded.
- This is a closed-book/notes exam. Consulting any resource is NOT permitted. Do not use your phones during the exam.
- Any kind of cheating will lead to score 0 for the entire exam and be reported to SJACS.

1 k-means clustering (8 points)

Recall the k-means objective function $F(\{\gamma_{ij}\}, \{\boldsymbol{\mu}_j\})$ (Eq 1), and the k-means algorithm (in the box below):

$$F(\{\gamma_{ij}\}, \{\mu_j\}) = \sum_{i=1}^{n} \sum_{j=1}^{k} \gamma_{ij} \|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|_2^2. \tag{1}$$

Step 0 Initialize $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$
Step 1 For the centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$ being fixed, assign each point to the closest center:

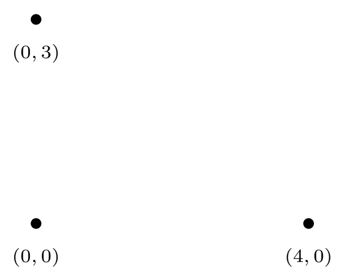
$$\gamma_{ij} = \mathbb{I}\left[j == \underset{c}{\operatorname{argmin}} \|\boldsymbol{x}_i - \boldsymbol{\mu}_c\|_2^2 \right]$$

Step 2 For the assignments $\{\gamma_{ij}\}$ being fixed, update the centers

$$\boldsymbol{\mu}_j = \frac{\sum_i \gamma_{ij} \boldsymbol{x}_i}{\sum_i \gamma_{ij}}$$

Step 3 Return to Step 1 if not converged (convergence means that all the assigments γ_{ij} are unchanged in Step 1).

Now consider the following data points in a 2-D plane:



(a) Suppose you run k-means with $k = 2$ and the initial cluster centers being $(4, 0)$ and $(0, 3)$. What are the final cluster centers when k-means converges? (2 points)

$(4, 0)$ and $(0, 1.5)$. In the first round, we will group $(0, 3)$ and $(0, 0)$ together, and $(4, 0)$ as the other group. Thus, we'll get the new center $(4, 0)$ and $(0, 1.5)$, and k-means converges.

(b) Prove that the above initialization converges to the global minimum for $k = 2$. (Hint: Enumerate over all possible cluster assignments.) (4 points)

The objective for k-means is :

$$F(\{\gamma_{ij}\}, \{\mu_j\}) = \sum_{i=1}^{3}\sum_{j=1}^{2} \gamma_{ij}\|x_i - \mu_j\|_2^2 \tag{2}$$

There are three possible groupings:

(1.) $(4, 0)$ with $(0, 0)$; $(0, 3)$

(2.) $(4, 0)$ with $(0, 3)$; $(0, 0)$

(3.) $(0, 3)$ with $(0, 0)$; $(4, 0)$

The objective for (1.) is: 4; for (2.) is: 5; for (3.) is: 3. Using our initialization in part (a) can lead us to group $(0, 3)$ and $(0, 0)$, and $(4, 0)$ as the other group. This assignment also has a loss of 3. Hence, it's a global minimum.

(c) Suppose we uniformly randomly pick two distinct data points (out of the three datapoints) as the cluster centers at initialization. What is the probability that k-means converges to the global minimum? (2 points)

Three cases of the initialization choices:

(1.) Selecting $(4, 0)$ and $(0, 0)$

(2.) Selecting $(4, 0)$ and $(0, 3)$

(3.) Selecting $(0, 3)$ and $(0, 0)$

For (1.), we will end up grouping $(0, 3)$ with $(0, 0)$. Hence, this is an optimal global assignment. For (3.), we will end up grouping $(4, 0)$ with $(0, 0)$. According to our discussion of (b), this is not an optimal assignment.

As a result, we have a $2/3$ probability to get a global minimum if we uniformly randomly pick two distinct data points (out of the three datapoints) as the cluster centers at initialization.

2 Linear discriminant analysis (LDA) in 1-D (8 points)

In this question we will explore a simple generative model for classifying 1-D data with a Gaussian assumption (the generalization of the approach to higher dimensions is known as linear discriminant analysis (LDA), hence the name of the question).

Consider a binary classification task on one dimensional data with labels $\{1, 2\}$. We have n_1 datapoints with label 1 and these are denoted as $\{x_1^{(1)}, \dots, x_{n_1}^{(1)}\}$. Similarly, we have n_2 datapoints $\{x_1^{(2)}, \dots, x_{n_2}^{(2)}\}$ with label 2. We will use a generative model to predict the labels of the datapoints. Recall that a generative model will output the label given by $\operatorname{argmax}_{j \in [2]} p(x, y = j)$. For $j \in \{1, 2\}$, we model $p(x|y = j)$ as

$$p(x|y=j) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu_j)^2}{2\sigma^2}\right),$$

for parameters $\mu_j \in \mathbb{R}$. We assume $\sigma \in \mathbb{R}^+$ is some appropriately chosen constant and is not a parameter of the model (and is also the same for all j).

(a) What is the maximum likelihood estimate (MLE) of $\{\mu_j\}_{j=\{1,2\}}$ given the data? Write down the expression for the log-likelihood, and show your derivation. (4 points)

Let $X_j = \{x_1^{(j)}, ..., x_{n_j}^{(j)}\}$. Define

$$\begin{aligned} F(\mu_1, \mu_2) &= \prod_{j \in [2]} \prod_{x \in X_j} p(x|y=j) p(y=j) \\ G(\mu_1, \mu_2) = \ln F(\mu_1, \mu_2) &= \sum_{j \in [2]} \sum_{x \in X_j} \ln p(x|y=j) + \ln p(y=j) \\ &= \text{some constant} + \sum_{j \in [2]} \sum_{x \in X_j} -\frac{(x-\mu_j)^2}{2\sigma^2} + \ln p(y=j) \end{aligned} \tag{3}$$

To maximize the log-likelihood, we can take the derivative and set it to 0, since we have a convex function. Therefore,

$$\begin{aligned} \frac{\partial G}{\partial \mu_j} &= -2 \sum_{x \in X_i} (x - \mu_j) = 0 \\ \implies \sum_{x \in X_j} x &= n_j \times \mu_j \\ \implies \mu_j &= \frac{1}{n_j} \sum_{x \in X_j} x \end{aligned} \tag{4}$$

(b) Suppose $n_1 = n_2$. What is the MLE estimate for $p(y)$? Use this to show that the final predictor has the following form, (4 points)

$$\operatorname*{argmax}_{j\in[2]} p(x, y = j) = \begin{cases} 1, & \text{if } |x-\mu_1| \le |x-\mu_2| \\ 2, & \text{otw.} \end{cases}$$

$$p(y = j) = \frac{n_j}{\sum_{j=[2]} n_j} = \frac{1}{2} \tag{5}$$

$$p(x, y = j) = p(x|y = j)p(y = j) = \frac{1}{2}\frac{1}{\sqrt{2\pi}\sigma}\exp\left(\frac{-(x-\mu_j)^2}{2\sigma^2}\right) \tag{6}$$

From above, we get

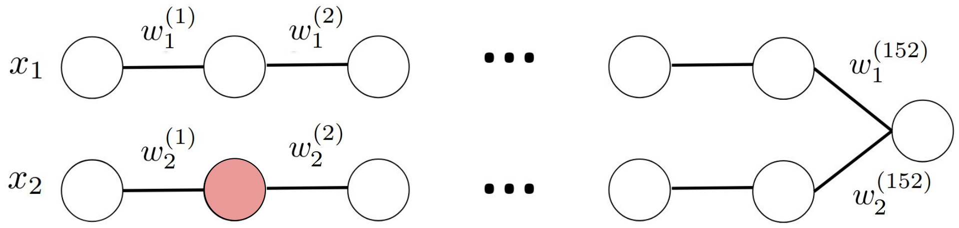
$$\operatorname*{argmax}_{j\in[2]}\ p(x, y = j) \tag{7}$$

$$= 1 \cdot I\big[p(x, y = 1) \le p(x, y = 2)\big] + 2 \cdot I\big[p(x, y = 1) > p(x, y = 2)\big] \tag{8}$$

$$= 1 \cdot I\big[|x-\mu_1| \le |x-\mu_2|\big] + 2 \cdot I\big[|x-\mu_1| > |x-\mu_2|\big] \tag{9}$$

3 Backpropagation and Vanishing Gradients (15 points)

Consider the 152-layer neural network in the figure below. Throughout the question, we assume that the training example is $(\boldsymbol{x}, y)$ where $\boldsymbol{x} = (x_1, x_2) \in \mathbb{R}^2$ and $y \in \mathbb{R}$ (all activations and gradients are with respect to this training example). The network has no bias terms, and it is fully-connected only for the final output node (i.e. the input x_1 is not connected to the red node in the next layer and so on, except for the last output node).



Notation: The notation for the weights is as described in the figure above. The notation for the inputs and outputs to the nodes is analogous. For e.g., the red node has input $a_2^{(1)} = w_2^{(1)} x_2$ and output $o_2^{(1)} = h(a_2^{(1)})$. Similarly, the final output node has input $a_1^{(152)} = w_1^{(152)} o_1^{(151)} + w_2^{(152)} o_2^{(151)}$ and output $o_1^{(152)} = h(a_1^{(152)})$.

Activation functions: The network uses some activation function $h(\cdot)$ for all nodes in all layers.

Loss function: The network is trained on the standard squared loss, so that $L(\boldsymbol{w}) = \frac{1}{2}(o_1^{(152)} - y)^2$ for the example $(\boldsymbol{x}, y)$, where $\boldsymbol{w}$ is the vector of all the weights in the network.

For questions (a) and (b), you can use $h'(a)$ to represent the derivative of the activation function $h(\cdot)$ with respect to some input a in your answer.

(a) Derive expressions for $\frac{\partial L(\boldsymbol{w})}{\partial o_1^{(152)}}$ and $\frac{\partial L(\boldsymbol{w})}{\partial w_1^{(152)}}$. (4 points)

$$\begin{aligned}
\frac{\partial L(\boldsymbol{w})}{\partial o_1^{(152)}} &= \frac{\partial}{\partial o_1^{(152)}} \left(\frac{1}{2} (o_1^{(152)} - y)^2 \right) \\
&= (o_1^{(152)} - y) \\
\frac{\partial L(\boldsymbol{w})}{\partial w_1^{(152)}} &= \frac{\partial L(\boldsymbol{w})}{\partial o_1^{(152)}} \cdot \frac{\partial o_1^{(152)}}{\partial a_1^{(152)}} \cdot \frac{\partial a_1^{(152)}}{\partial w_1^{(152)}} \\
&= (o_1^{(152)} - y) \cdot \frac{\partial}{\partial a_1^{(152)}} (h(a_1^{(152)})) \cdot \frac{\partial}{\partial w_1^{(152)}} (w_1^{(152)} o_1^{(151)} + w_2^{(152)} o_2^{(151)}) \\
&= (o_1^{(152)} - y) \cdot h'(a_1^{152}) \cdot o_1^{(151)}
\end{aligned}$$

(b) Derive expressions for $\frac{\partial L(\boldsymbol{w})}{\partial o_1^{(l)}}$ and $\frac{\partial L(\boldsymbol{w})}{\partial w_1^{(l)}}$ for all hidden layers $l = 1, \ldots, 151$ in the network.

(Hint: Use the chain rule, for example, $\frac{\partial o_1^{(l+1)}}{\partial o_1^{(l)}} = \frac{\partial o_1^{(l+1)}}{\partial a_1^{(l+1)}} \frac{\partial a_1^{(l+1)}}{\partial o_1^{(l)}}$.) (5 points)

$$\begin{aligned}\frac{\partial L(\boldsymbol{w})}{\partial o_1^{(l)}} &= \frac{\partial L(\boldsymbol{w})}{\partial o_1^{(152)}} \cdot \frac{\partial o_1^{(152)}}{\partial o_1^{(151)}} \cdots \frac{\partial o_1^{(l+1)}}{\partial o_1^{(l)}} \\ &= \frac{\partial L(\boldsymbol{w})}{\partial o_1^{(152)}} \cdot \prod_{l'=l}^{151} \frac{\partial o_1^{(l'+1)}}{\partial o_1^{(l')}}\end{aligned}$$

For $l = 1, \cdots, 151$, $o_1^{(l+1)} = h(a_1^{(l+1)})$. For $l = 1, \cdots, 150$, $a_1^{(l+1)} = w_1^{(l+1)} o_1^{(l)}$, and $a_1^{(152)} = w_1^{(152)} o_1^{(151)} + w_2^{(152)} o_2^{(151)}$. Using these, we get:

$$\begin{aligned}\frac{\partial o_1^{(l+1)}}{\partial o_1^{(l)}} &= \frac{\partial o_1^{(l+1)}}{\partial a_1^{(l+1)}} \cdot \frac{\partial a_1^{(l+1)}}{\partial o_1^{(l)}} \\ &= h'(a_1^{(l+1)}) \cdot w_1^{(l+1)}\end{aligned}$$

for $l = 1, \cdots, 151$. Using this and (a), we get:

$$\frac{\partial L(\boldsymbol{w})}{\partial o_1^{(l)}} = (o_1^{(152)} - y) \cdot \prod_{l'=l}^{151} h'(a_1^{l'+1}) w_1^{(l'+1)}$$

$$\begin{aligned}\frac{\partial L(\boldsymbol{w})}{\partial w_1^{(l)}} &= \frac{\partial L(\boldsymbol{w})}{\partial o_1^{(152)}} \cdot \frac{\partial o_1^{(152)}}{\partial o_1^{(151)}} \cdots \frac{\partial o_1^{(l+2)}}{\partial o_1^{(l+1)}} \cdot \frac{\partial o_1^{(l+1)}}{\partial w_1^{(l)}} \\ &= \frac{\partial L(\boldsymbol{w})}{\partial w_1^{(152)}} \cdot \prod_{l'=l+1}^{151} \frac{\partial o_1^{(l'+1)}}{\partial o_1^{(l')}} \cdot \frac{\partial o_1^{(l+1)}}{\partial w_1^{(l)}} \\ \frac{\partial o_1^{(l+1)}}{\partial w_1^{(l)}} &= \frac{\partial o_1^{(l+1)}}{\partial a_1^{(l+1)}} \cdot \frac{\partial a_1^{(l+1)}}{\partial w_1^{(l)}} \\ &= h'(a_1^{(l+1)}) \cdot o_1^{(l)}\end{aligned}$$

Using this and (a), we get:

$$\frac{\partial L(\boldsymbol{w})}{\partial w_1^{(l)}} = (o_1^{(152)} - y) \cdot o_1^{(l)} \cdot \prod_{l'=l+1}^{151} h'(a_1^{(l'+1)}) w_1^{(l'+1)}$$

(c) Suppose we use the sigmoid activation $h(a) = \frac{1}{1+e^{-a}}$ for all nodes in this neural network. The gradient of $h(a)$ is $h'(a) = h(a)(1-h(a))$. Using your expression from the previous part, describe what the gradients $\frac{\partial L(\boldsymbol{w})}{\partial w_1^{(l)}}$ for the weights in the earlier layers of the network would be like, and explain why (earlier layers refers to small l). (Hint: You don't need to actually use the expression for the sigmoid activation, just think about the range of values the gradient of the activation takes. The range of $h(a)$ is $[0,1]$. What is the range of $h'(a)$?) (3 points)

When $h(a)$ is the sigmoid activation, $h'(a) \in [0, 0.25]$. The number of terms in the product (for the expression of $\frac{\partial L(\boldsymbol{w})}{\partial w_1^{(l)}}$), which depend on $h'(a)$ and hence, take fractional values, is given by $151 - l$. For earlier layers, l is small (e.g. < 50), so the gradient contains a product of a large number (> 100) of fractional values. Thus, the gradients for the earlier layers will be negligibly small.

(d) Suppose we use ReLU as the activation function for all nodes, i.e. $h(a) = \max(0, a)$. When would the gradients $\frac{\partial L(\boldsymbol{w})}{\partial w_1^{(l)}}$ for the weights for some layer of the network be zero? If we continue with the ReLU activation function but modify the network to be fully connected for all layers (i.e. add all cross-connections by connecting x_1 and the red node and so on), when would the gradients of the weights be zero? Describing one case where the gradient is zero for each of these questions is enough. (3 points)

When using ReLU activation, we can write:

$$\frac{\partial L(\boldsymbol{w})}{\partial w_1^{(l)}} = (o_1^{(152)} - y) \cdot \max(0, a_1^{(l)}) \cdot \prod_{l'=l+1}^{151} \mathbb{1}[a_1^{(l'+1)} > 0] w_1^{(l'+1)}$$

Thus, if $a_1^{(l')} \leq 0$ or $w_1^{(l'+1)} = 0$ for even one value of $l' \geq l$, then the gradient for $w_1^{(l)}$ will be 0. Further, since $a_1^{(l')} = w_1^{(l')} o_1^{(l'-1)}$ and $o_1^{(l'-1)} \geq 0$, this means that the gradient will be 0 when $w_1^{(l')} \leq 0$, for any $l' \geq l$.

When the layers are converted to fully connected ones, $a_1^{(l+1)} = w_1^{(l+1)} o_1^{(l)} + w_3^{(l+1)} o_2^{(l)}$ for $l = 1, \cdots, 150$, but the expressions for the partial derivatives remain the same. In this case, the gradients will be 0 when $w_1^{(l')} o_1^{(l'-1)} + w_3^{(l')} o_2^{(l'-1)} \leq 0$ for any $l' \geq l$.

4 PCA (12 points)

Consider the usual PCA setup. There is a dataset with n datapoints $\{\boldsymbol{x}_i, i \in [n]\} \in \mathbb{R}^d$. Assume that the data is 0 mean. The data is represented as a matrix $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ as usual, with each row being the (transpose) of the corresponding datapoint. Let the covariance matrix $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ have the eigendecomposition $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} = \boldsymbol{QDQ}^{\mathrm{T}}$. Here $\boldsymbol{D}$ is a diagonal matrix containing the eigenvalues $(\lambda_1, \dots, \lambda_d)$ as diagonal entries, sorted in decreasing order (assume all the eigenvalues are unique). $\boldsymbol{Q}$ is an orthogonal matrix ($\mathbf{Q}^T\mathbf{Q} = \mathbf{Q}\mathbf{Q}^T = \mathbf{I}$), and consists of the eigenvectors. We proved in class that the j^{th} principal component $\boldsymbol{v}_j$ is equal to the eigenvector corresponding to the j^{th} eigenvalue, which is the j^{th} column $\boldsymbol{q}_j$ of $\boldsymbol{Q}$.

In this question we will prove some properties of PCA for data compression which we have already implicitly stated in class. We first note that we can represent any datapoint $\boldsymbol{x}_i$ by using all the d principal components,

$$\boldsymbol{x}_i = \sum_{j=1}^{d} \alpha_{i,j}\boldsymbol{v}_j \tag{10}$$

where $\alpha_{i,j}$ are some coefficients. If we only use the top k principal components (for some $k \leq d$), we can get an approximation $\hat{\boldsymbol{x}}_i$ of the i^{th} datapoint $\boldsymbol{x}_i$ as follows:

$$\hat{\boldsymbol{x}}_i = \sum_{j=1}^{k} \alpha_{i,j}\boldsymbol{v}_j. \tag{11}$$

Consider the overall reconstruction error if we use the top k principal components:

$$E = \sum_{i=1}^{n} \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|_2^2 = \sum_{i=1}^{n} (\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i)^{\mathrm{T}}(\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i). \tag{12}$$

(a) Show that for all $i \in [n]$ and $j \in [k]$, choosing the coefficients $\alpha_{i,j} = \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{v}_j$ in Eq. 11 minimizes the reconstruction error E in Eq. 12. To do this, follow the following steps,

- Step 1: First consider the case when $k = 1$. Expand out Eq. 12 using Eq. 11, and then find the best value of $\alpha_{i,j}$. (3 points)
- Step 2: Then, consider the case of general k. Here, use the fact that $\boldsymbol{v}_j = \boldsymbol{q}_j$ and $\boldsymbol{Q}$ is an orthogonal matrix. (2 points)

Expanding out Eq. 12 using Eq. 11 with $k = 1$, we have

$$E = \sum_{i=1}^{n} (x_i - \alpha_{i,1}v_1)^\top (x_i - \alpha_{i,1}v_1).$$

Taking derivative w.r.t $\alpha_{i,1}$ for each i and set it to zero, we have

$$\frac{\partial E}{\partial \alpha_{i,1}} = 2(x_i - \alpha_{i,1}v_1)^\top v_1 = 0.$$

Solving this gives $\alpha_{i,1} = x_{i,1}^\top v_1$ by $\|v_1\|_2^2 = 1$. For general k, we have

$$\frac{\partial E}{\partial \alpha_{i,j}} = 2(x_i - \sum_{l=1}^{k} \alpha_{i,l}v_l)^\top v_j = 2(x_i^\top v_j - \alpha_{i,j}) = 0,$$

where the last step is by the fact that $\{v_j\}_j$ are orthoganal unit vectors. This gives $\alpha_{i,j} = x_i^\top v_j$.

(b) Prove that $E = \sum_{j=k+1}^{d} \lambda_j$. To show this, you should follow the following three steps.[1] If you get stuck on some step, feel free to try out the next one using the conclusion of the previous step.

- Step 1: Plug Eq. 10 and 11 into the expression for E in Eq. 12. Next, use the fact that $\boldsymbol{v}_j = \boldsymbol{q}_j$ and $\boldsymbol{Q}$ is orthogonal to simpify the expression. You should be able to show that $E = \sum_{i=1}^{n} \sum_{j=k+1}^{d} \alpha_{i,j}^2$. (3 points)
- Step 2: Use the fact that $\alpha_{i,j} = \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{v}_j$ for all $i \in [n]$ and $j \in [d]$ to simplify the expression from Step 1. Finally, you should be able to show that $E = \sum_{j=k+1}^{d} \|\boldsymbol{X}\boldsymbol{v}_j\|_2^2$. (2 points)
- Step 3: Prove that $\|\boldsymbol{X}\boldsymbol{v}_j\|_2^2 = \lambda_j$ to finish the proof (use the fact that $\boldsymbol{v}_j = \boldsymbol{q}_j$, and that $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} = \boldsymbol{Q}\boldsymbol{D}\boldsymbol{Q}^{\mathrm{T}}$). (2 points)

Plugging Eq. 10 and 11 into the expression for E in Eq. 12, we have

$$E = \sum_{i=1}^{n} (\sum_{j=k+1}^{d} \alpha_{i,j} v_j)^\top (\sum_{j=k+1}^{d} \alpha_{i,j} v_j) = \sum_{i=1}^{n} \sum_{j=k+1}^{d} \alpha_{i,j}^2,$$

where the last step is by the fact that $\{v_j\}_j$ are orthogonal. Now by $\alpha_{i,j} = x_i^\top v_j$, we have

$$E = \sum_{i=1}^{n} \sum_{j=k+1}^{d} v_j^\top x_i x_i^\top v_j = \sum_{j=k+1}^{d} v_j^\top (\sum_{i=1}^{n} x_i x_i^\top) v_j = \sum_{j=k+1}^{d} v_j^\top X^\top X v_j = \sum_{j=k+1}^{d} \|X v_j\|_2^2$$

Finally, note that $\|Xv_j\|_2^2 = \lambda_j$ since

$$\|Xv_j\|_2^2 = v_j^\top X^\top X v_j = v_j^\top Q D Q^\top v_j = e_j^\top D e_j = \lambda_j.$$

Plugging this back completes the proof.

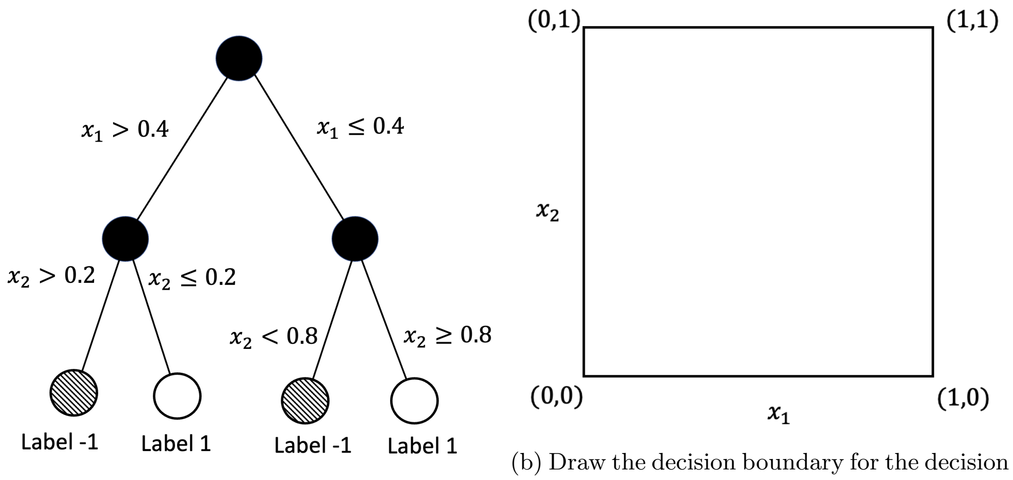
[1]You can also attempt a completely different proof, but will get limited partial credits for incomplete or incorrect attempts.

(you can use this page to finish the previous PCA question, just in case you run out of space)

5 Short answer questions (17 points)

5.1 Decision boundary for decision trees (3 points)

Consider the decision tree in Fig. 1a. The tree takes $\boldsymbol{x} = (x_1, x_2)$ as input, and outputs a label $y \in \{-1, +1\}$. The internal nodes are solid black, the leaves are shaded if the label is -1, and white if the label is 1. Draw the decision boundary for this decision tree in Fig. 1b. You can represent -1 as shaded, and 1 as solid white.



(a) Decision tree

(b) Draw the decision boundary for the decision tree on the left

Figure 1: Decision boundary for decision trees

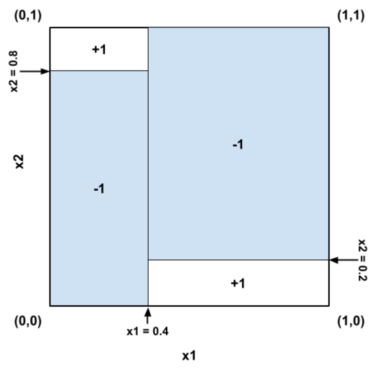


Figure 2: Decision boundary for the above decision tree

5.2 Boosting decision stumps (6 points)

Consider the following dataset with 2-dimensional inputs (x_1, x_2) and binary outputs $\{-1, +1\}$.

Consider running AdaBoost with a decision stump as the base classifier on this dataset. Recall that a decision stump is just a decision tree with a root and two leaves as children. More formally, it is of the form

$$f(\boldsymbol{x}) = \begin{cases} b_1, & \text{if } x_i \geq \theta, \\ b_2, & \text{if } x_i < \theta. \end{cases} \tag{13}$$

Datapoint	Label
(-2,-1)	$+1$
(0,0)	-1
(1,-1)	$+1$
(3,3)	$+1$
(-2,3)	$+1$

where $b_1, b_2 \in \{-1, +1\}$, $i \in \{1, 2\}$ and $\theta \in \mathbb{R}$. Recall that the boosted classifier at the end of T iterations be given by $f_{boost}^{(T)}(\boldsymbol{x}) = \text{sign}(\sum_{t=1}^{T} \beta_t f_t(\boldsymbol{x}))$. Here $f_t(\boldsymbol{x})$ is the weak classifier found by the AdaBoost algorithm at the t^{th} iteration.

(a) Which datapoint(s) will have their weights increased at the end of the first iteration of AdaBoost ($T = 1$) for this dataset? (Hint: You might find it helpful to draw out the dataset. Is it possible to classify all datapoints correctly for $T = 1$? Is it possible to classify all but one correctly for $T = 1$?). (3 points)

One point will always be misclassified. The best classifier in the first iteration will predict every point as class $+1$. Thus, after the first iteration, the weight of the class -1 point, i.e. (0, 0) will increase. Any other stump will make more errors and incur a higher loss.

(b) Is it possible for the algorithm to classify all datapoints correctly after $T = 2$ iterations? Explain. (Hint: You can reason about it directly, and don't need the formulae for AdaBoost). (3 points)

No. We have seen that the first stump will misclassify (0, 0). In the second iteration, the second stump has to correctly classify point (0, 0) since it incurs the highest penalty when misclassified. In the process, it will misclassify at least 2 points of class $+1$ as -1. Thus, neither stump 1 nor stump 2 can differentiate between the class -1 point, i.e. (0, 0), and at least one class $+1$ point. So, the 2 stumps will jointly misclassify one of these two points.

5.3 n-gram models for language (4 points)

Recall that a 3-gram language model uses a second-order Markov assumption to model the data:

$$P(X_{t+1}|X_t, \ldots, X_1) = P(X_{t+1}|X_t, X_{t-1}).$$

Suppose we train the second-order Markov model on the following text corpus. (For the purpose of this question, assume that any punctuation marks are part of the word, so *"Jude,"* is one word.)

> Hey Jude, don't make it bad. Take a sad song and make it better. Hey Jude, don't be afraid. You were made to go out and get her. Hey Jude, refrain. Don't carry the world upon your shoulders.

We can now use the trained Markov model to predict the distribution of the next word following any sequence of words. What is the distribution of the next word in the following text (to fill in the ______), as predicted by the trained Markov model?

And don't you know that it's just you, hey Jude, you'll do, the movement you need is on your shoulder. Hey Jude, ______

Since we are using a 3-gram model, to find the continuation, we only need to consider what words follow *"Hey Jude,"* in the training data and calculate the distribution.

Word	Count	Probability
don't	2	$\frac{2}{3}$
refrain	1	$\frac{1}{3}$
Total	3	

The distribution assigns 0 probability to any other word.

5.4 Neural networks for language modelling (4 points)

Continuing with the text corpus from the previous question, suppose we now want to train a standard fully-connected neural network for the task. The network should take the previous two words as input and predict the distribution of the next word.

(a) Describe how to convert the prediction task to a standard supervised ML problem. To do this, describe the input and output which will be used to train the model (you should provide an example of a training datapoint, with the input and desired output). (2 points)

We first define a vocabulary of words from the training data (say of size $|V|$) and assign an integer index to every word. Each training and evaluation instance consists of 2 one-hot vectors (corresponding to words at $(t-1)$ and $(t-2)$) concatenated together as input and the output is a one-hot vector for the word at step t. Each training instance is created from 3 consecutive words from the text. Consider the 3-gram *"Hey Jude, don't"*:
Input: $\{0,1\}^{2|V|}$ e.g. If *"Hey"* is word 20 and *"Jude,"* is word 22, then the input is a vector with 1 at position 20 and $|V|+22$ and zero everywhere else (it is also acceptable to use word embeddings for the input)
Output: $\{0,1\}^{|V|}$ e.g. If *"don't"* is word 24, then the output is a vector with 1 at position 24 and zero everywhere else

(b) Write the name, and the mathematical expression for the loss function that we can use to train the model on this task. You can assume that the model has a softmax layer and outputs a probability distribution for the word (let the predicted probability of word w at the t-th time step be $\hat{y}_w^{(t)}$). (Hint: the answer is one of the following: $-\log \hat{y}_{w_t}^{(t)}, \log \hat{y}_{w_t}^{(t)}, \log(1-\hat{y}_{w_t}^{(t)}), -\log(1-\hat{y}_{w_t}^{(t)})$, where you also need to say what w_t is.) (2 points)

At each time step, the model behaves like a classifier assigning a likelihood to the t-th word given words at position $(t-1)$ and $(t-2)$. This is similar to a multiclass classification problem and hence we can use the multiclass cross-entropy loss (saying cross-entropy loss or log-loss is also acceptable). At any time step t, the loss is

$$-\log \hat{y}_{w_t}^{(t)} \tag{14}$$

where $\hat{y}_{w_t}^{(t)}$ is the model's predicted probability of the true word w_t at time step t.

6 Adventures of a CSCI567 graduate: The Saga Continues (10 points)

In the last quiz, we told a few stories about your adventures in the world of ML. This question continues those chronicles.

6.1 Evaluating ML models (4 points)

A rare kind of blood disorder whose consequences are controllable if diagnosed early affects **0.1**% of the population. The condition has so far been expensive to diagnose, but you are working on a ML-based approach that uses the result of an inexpensive lab test as features and predicts if the person has the disorder. In which of the following scenarios can you conclude that your ML model is doing a good job? Explain each answer in 1-2 sentences.

(a) The model predicts if a person has the disorder with accuracy 99.9%.

The model is bad. A model that always predicts "has no disorder" has the same accuracy.

(b) The model predicts if a person has the disorder with precision 0.1% and recall 100%.
(Hint: Let positive instances refer to the person having the disorder. Then

$$\text{Precision} = \frac{\text{\#true positive instances caught by algorithm (TP)}}{\text{\#instances flagged by algorithm as positive (TP+FP)}},$$

$$\text{Recall} = \frac{\text{\#true positive instances caught by algorithm (TP)}}{\text{\#instances which are positive in the entire data (TP+FN)}}.$$

Here TP, FP and FN refer to true positive, false positive and false negative respectively.)

The model is bad. A model that always predicts "has disorder" has the same precision and recall.

(c) The model predicts if a person has the disorder with precision 10% and recall 90%.

The model is good. The model can detect 90% of the people that have the disorder, and does this with a precision of 10%. Since only 0.1% have the disorder, this is quite useful. It would be a good model to use as an initial filter, and can be followed by further, more expensive diagnosis.

6.2 ML advice (6 points)

As you gain more experience, you start to advise and help other teams with their ML projects. What advice would be provide in the following scenarios? Write your recommendation, along with the explanation (2-3 sentences total).

(a) *"I want to build a ML based model to predict if a customer will return a loan. I have collected several features (whether the person took a loan previously, employment status, income etc.) for this task. I want the model to ideally have some interpretability since the customer may demand an explanation for the decision."*

Decision trees or random forests could be a good choice. Decision trees have good interpretability, and random forests could allow it to obtain more expressive power. Decision-trees and ensembles of decision trees are usually good for structured data, such as in this case.

(b) *"I run a retail website and want to optimize the presentation of items on my website, so that items which are more likely to be purchased draw more attention. I have limited data/features regarding these items at present, but can collect statistics of clicks/purchases of different items. Ideally, the model should improve without manual intervention and get better over time as more users visit the website."*

A bandit algorithm (such as the UCB algorithm) could be a good choice for determining the presentation of items. The algorithm balances exploration vs. exploitation in multi-armed bandits (MAB) problems. The provided problem is a typical MAB, which requires us to learn the best probability of presenting each item in an online fashion with limited feedback, that is, each time we only have the feedback from the presented and clicked items but not all the items on the website.

7 Multiple choice questions (30 points)

IMPORTANT: *Select ALL options among {A,B,C,D} that you think are correct (no justification needed). You get 0.5 point for selecting each correct option and similarly 0.5 point for not selecting each incorrect option. You get 1 additional point for selecting all four options correctly.*

(1) Which of the following are best practices for responsible ML?

(A) It is important to identify the right metrics to quantify the performance of the ML model (such as accuracy, precision, recall, performance on sub-groups etc.)
(B) The model may latch onto spurious correlations to make their predictions, so should be tested extensively in use cases which directly impact large populations.
(C) The model should be tested on diverse data which reflects the use cases in deployment.
(D) The model should not be monitored or updated after deployment.

ABC

(2) Suppose we have a fully-connected neural network with 1 hidden layer with ReLU activations ($h(a) = \max(0, a)$) for a binary classification task. Which of the following statements are true about the behavior of the network at training and test time?

(A) The total training time is usually the fastest for the smallest possible batch size, since each gradient step takes less time.
(B) Early stopping may lead to poorer training accuracy, but can improve test accuracy.
(C) Adding a momentum term can help the network converge faster, but too much momentum can be harmful.
(D) Multiplying all the weights and biases in the network by a factor of 10 after training the network will not change its classification accuracy.

BCD

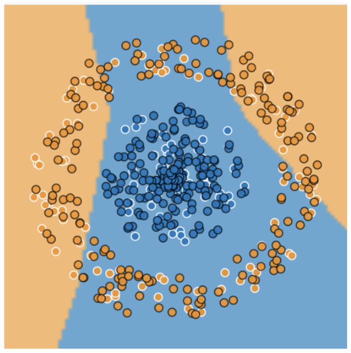
A: Although each gradient update is faster for small batch size, the number of updates is larger. D: The logit values may change, but the predictions remain the same.

(3) Consider a convolution layer which takes a $10 \times 10 \times 3$ dimensional input and convolves it with 5 filters of size 3×3 filters with stride 1 and no zero padding. Which of the following statements are true about this layer? (Recall that the output volume as a result of convolving a $N \times N \times 3$ image with one filter of size $F \times F$ with stride S and zero padding P is $D \times D \times 1$ dimensional, where $D = (N + 2P - F)/S + 1$).

(A) The output volume is $8 \times 8 \times 5$ dimensional.
(B) The output volume is $8 \times 8 \times 3$ dimensional.
(C) If the layer was fully connected, it would have more than 10 times as many parameters as currently in the convolution layer.
(D) Without zero-padding the input, we cannot use stride $S = 2$ instead of $S = 1$ in this layer.

ACD. A: Simply using the formula above with $N = 10$, $P = 0$, $F = 3$, and $S = 1$.

(4) Which of the following classifiers could have been used to generate the decision boundary in the binary classification task on the right?



(A) A one hidden layer fully connected neural network with ReLU activation functions ($h(a) = \max(0, a)$).
(B) A one hidden layer fully connected neural network with linear activation functions ($h(a) = a$).
(C) A three hidden layer fully connected neural network with linear activation functions ($h(a) = a$).
(D) A decision stump (decision tree of depth 1).

A: Others are unable to produce a non-linear decision boundary.

(5) Consider a binary dataset with 50 positive examples 50 negative examples. Decision stump $\mathcal{T}_1$ splits this dataset into two children where the left one has 10 positive examples and 50 negative examples, while another decision stump $\mathcal{T}_2$ results in a left child with 0 positive examples and 50 negative examples. Which of the following is correct?

(A) The entropy of the left child of $\mathcal{T}_1$ is $\frac{1}{6}\ln 6 + \frac{5}{6}\ln\frac{6}{5}$. (Hint: Entropy of a random variable X taking value over some domain $\mathcal{X}$ and having density $p(x)$ is $\sum_{x\in\mathcal{X}} p(x)\log(1/p(x))$).
(B) The entropy of the right child of $\mathcal{T}_1$ is 0.
(C) The entropy of either child of $\mathcal{T}_2$ is the minimum possible entropy for two classes.
(D) Based on conditional entropy, $\mathcal{T}_2$ is a better split than $\mathcal{T}_1$.

ABCD.
Entropy of the nodes is defined by

$$\sum_{c=1}^{C} P(Y = c)\log\left(\frac{1}{P(Y = c)}\right)$$

where $P(Y = c)$ is the fraction of nodes with label c in the node. Entropy is always non-negative. Thus, 0 entropy is the minimum possible entropy. Both children of $\mathcal{T}_2$ have 0 entropy while only the right child of $\mathcal{T}_1$ has 0 entropy i.e. conditional entropy of $\mathcal{T}_2$ is 0 while that of $\mathcal{T}_1 > 0$.

(6) Figure 3 shows various training/test classification error curves as parameters for training decision trees are varied. Select all options which represent reasonable relationships between the considered parameters and obtained error(s).

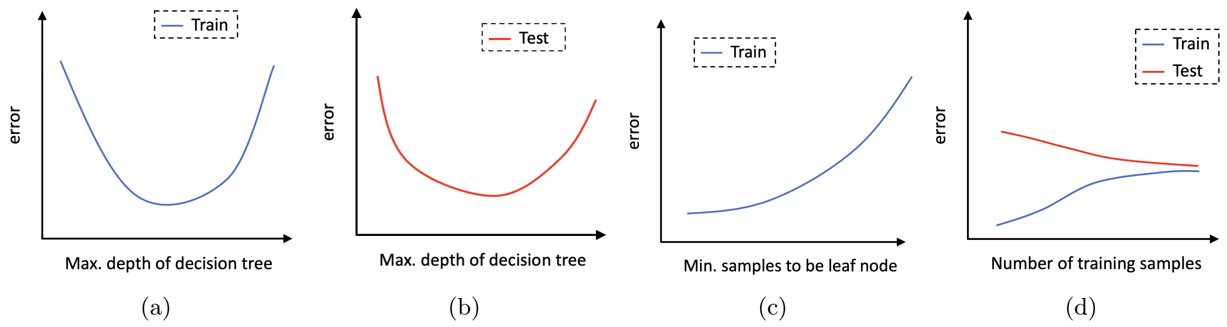


Figure 3: Possible curves for decision trees (select all which are reasonable)

(A) Fig 3a is a reasonable plot of training error as the maximum depth of the decision tree is increased.
(B) Fig 3b is a reasonable plot of test error as the maximum depth of the decision tree is increased.
(C) Fig 3c is a reasonable plot of training error as the minimum number of samples to be a leaf node is increased.
(D) Fig 3d is a reasonable plot of training & test errors as the total size of the training set used to train a decision tree with maximum depth 5 is increased.

BCD.
(A) is should be a decreasing curve because, as depth increases, the decision tree can continue to split nodes leading to lower train error.
(B) is correct because the tree will begin to overfit as its size increases beyond a certain point.
(C) is correct because increasing min. samples in leaf increases, the number of splits in the tree becomes restricted and the tree is more regularized.
(D) is correct because a fixed capacity (here depth) decision tree will become a worse fit for the data as the training set size increases. At the same time, the generalization should improve i.e. the test error decreases.

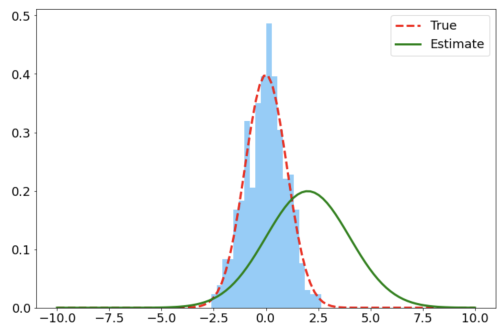
(7) Consider the problem of learning a mixture of $k = 2$ Gaussians on some one-dimensional data. Let the mean, covariance and clusters weight of the two Gaussians be μ_j, σ_j and π_j (respectively), for $j \in \{1, 2\}$. Which of the following statements are true about the problem?

(A) The problem of finding the parameters which maximize the log-likelihood is non-concave.
(B) Suppose EM finds the parameters $\mu_1 = -1, \sigma_1 = 1, \pi_1 = 0.8$ and $\mu_2 = 1, \sigma_2 = 1, \pi_2 = 0.2$. Then for a datapoint with $x = 0$ (equidistant from the two centers), its posterior probability of being in cluster 1 is 0.5.
(C) The learned GMM can be used to flag outliers/anomalies in the data.
(D) The learned GMM can be used to generate synthetic data.

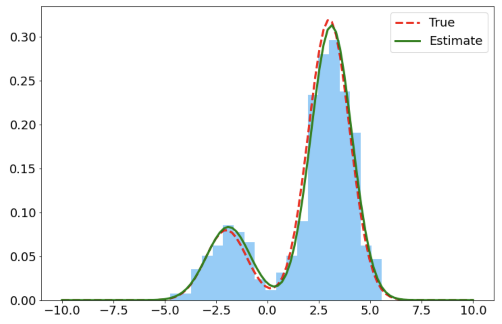
ACD.
B is incorrect because the posterior probability for this data point will be $\pi_1 = 0.8$.
A, C and D are properties of GMMs in general.

(8) Figure 4 shows executions of the EM algorithm with different values of the number of Gaussian components k and different initializations. The code is generated using the same demo as shown in class: the histogram of the data is in blue, the true density is given by the dashed red line and the model learned by EM is given by the solid green line. In which of the figures is it possible that the EM algorithm has converged?

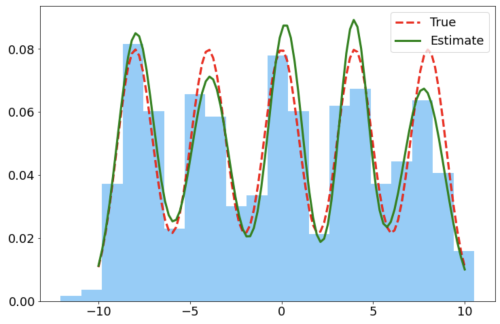
Answer: BCD.
A is clearly not converged, as the single Gaussian doesn't fit the data (means don't match). BC are probably converged, as the estimated lines are close to the true dash lines. D is also possible to have converged if the initialization of four Gaussian components are the same with $\pi_j = 1/4$, which means the Gaussian components will be updated in the same way (essentially, EM can get stuck in local optima).



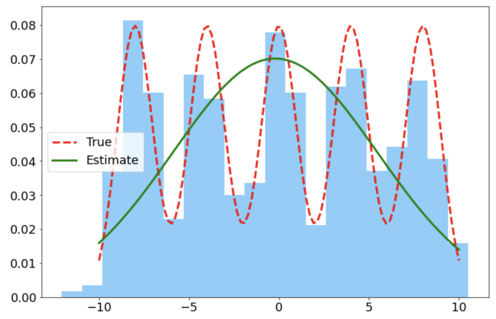
(A) EM execution with $k = 1$



(B) EM execution with $k = 2$



(C) EM execution with $k = 3$



(D) EM execution with $k = 4$

Figure 4: EM algorithm runs

(9) Which of the following statements are true about Principal Component Analysis (PCA)?

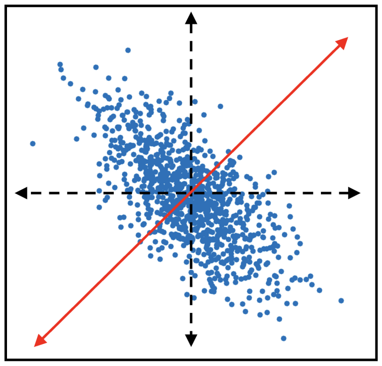
(A) In PCA, the first principal component is the direction where the projection of the dataset has the largest variance.
(B) In Fig. 5a, the red line represents the first principal component of the data.
(C) PCA will not work very well to discover the underlying structure for the dataset in Fig. 5b.
(D) The results of PCA are invariant to re-scaling the coordinates of the data.

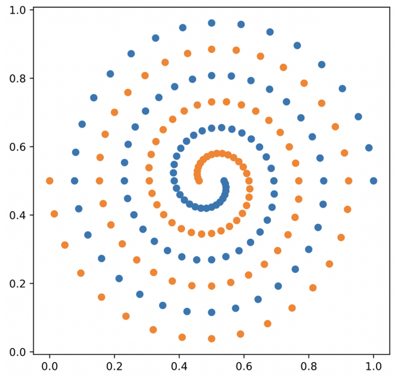
Answer: AC.
A is true by definition. B: the first principal component should be orthogonal to the red line. C has non-linear structure, so PCA will not work well. D: PCA is not scale invariant (think of the ellipse example in the class).

(10) Consider the dataset of 5 examples shown in Table 1. Each example corresponds to an email and has two features "Domain" and "Length", and one label "Label", which denotes or not that email was spam. All columns all have two possible values: "internal" or "external" for Domain, "long" or "short" for Length, and "not spam" or "spam" for Label.

Which of the following statements are true about Naive Bayes applied to this data? (Hint: Naive Bayes predicts using $\text{argmax}_j\, p(\boldsymbol{x}, y = j)$, where the Naive Bayes assumption is then used to expand out the joint probability).

(A) p(L="spam") = 2/5.
(B) p(Domain="internal" | L="spam") = 0.
(C) A new datapoint with Domain="external" and Length="short" is classified as "spam".


(a) Centered data with candidate first principal component in red.



(b) Data with some underlying structure given by the blue and orange points.

Figure 5: Datasets for PCA

i	Domain	Length	Label
1	internal	long	not spam
2	internal	short	not spam
3	external	short	not spam
4	external	long	spam
5	external	long	spam

Table 1: Training set for Naive Bayes

(D) A new datapoint with Domain="internal" and Length="long" is classified as "spam".

Answer: AB.
A is true by direct computation. Using Bayes rule,

$$p(\text{Domain=“internal”} \mid \text{L=“spam”}) = \frac{p(\text{Domain=“internal”, L=“spam”})}{p(\text{L=“spam”})} = 0.$$

B is true. By similar Bayes rules, we compute

$$p([\text{Domain=“external”, Length=“short”}], \text{L=“spam”}) = \frac{2}{5} < \frac{1}{2}$$

and

$$p([\text{Domain=“internal”, Length=“long”}], \text{L=“spam”}) = \frac{2}{5} < \frac{1}{2}.$$

Hence, both CD are false.