CS 337 Autumn 2023 | Endsem Exam

Instructor: Preethi Jyothi Date/Time: Nov 17, 2023, 8.30 am to 11.30 am

Total points: **50**

Name: _____

Roll number: _____

Instructions

- This is an open notes exam which should be completed individually. No form of collaboration or discussion is allowed.
- Use of laptops or cell phones are not allowed during the exam.
- Write your name and roll number on the top of this page.
- This exam consists of 5 problems with sub-parts. The maximum possible score is 50.
- Work efficiently. The questions are not sorted in any order of difficulty. Try and attempt the easier ones first.
- Write your answers legibly with a pen (not a pencil) in the space provided on the exam sheet. If necessary, use/ask for extra sheets to work out your solutions. These sheets will not be graded, so please make sure your final answers appear on the exam sheet.
- Good luck!

Question	Score
Mixed Bag (I)	/11
Mixed Bag (II)	/13
Search	/9
Boosting	/9
Pushy Perceptron	/8
Total	/50

Problem 1: Mixed Bag (I) (11 points)

(A) Let π be some probability distribution over integers. For integers $a \leq b$, let $\pi_{a,b}$ denote the following probability distribution:

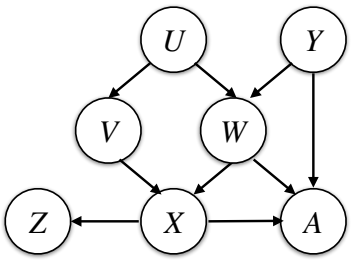
$$\pi_{a,b}(x) = \begin{cases} \frac{\pi(x)}{Z(a,b)} & \text{if } a \leq x \leq b \\ 0 & \text{otherwise} \end{cases}$$

where $Z(a,b) = \sum_{j=a}^{b} \pi(j)$. What is the MLE estimate of (a,b), given a set of n i.i.d. training points $x_1, \ldots, x_n$ drawn from the distribution $\pi_{a,b}$? Justify your answer. **[2 pts]**

Solution: Choose a and b such that 1) all the training points lie within this range 2) the probability of each point is maximized i.e., $Z(a,b)$ is as small as possible. The tightest such range would be $a = \min(x_1, \ldots, x_n)$ and $b = \max(x, \ldots, x_n)$.

Grading guide
1 point for getting a correct and 1 point for getting b correct.

(B) Consider the following Bayesian network. Using the principles of d-separation, explain whether the following conditional independence statements are guaranteed to be true or not. **[2 pts]**



i. $V \perp\!\!\!\perp Y \mid A$

Solution: False. $V \to X \to A \leftarrow Y$ is an active path.

Grading guide
No point for the correct answer without explanation.

ii. $V \perp\!\!\!\perp A \mid Y, W$

Solution: False. $V \to X \to A$ is an active path.

Grading guide
No point for the correct answer without explanation.

(C) Express the following natural language statements using first-order logic (FOL) operators and quantifiers, and use the specified predicates. If a sentence is ambiguous, give all logical forms. **[2 pts]**

i. "Whoever has a baby, has a cradle".
Predicates:

1. baby-of(x, y) holds iff person x is a baby of person y
2. belongs-to(x, y) holds iff thing x belongs to person y
3. is-cradle(x) holds iff thing x is a cradle

Solution: $\forall$ x ($\exists$ y baby-of(y, x)) $\implies$ ($\exists$ y is-cradle(y) $\wedge$ belongs-to(y, x))

Grading guide

1 point for any logically equivalent expression of the abovementioned answer.

ii. "Every contest has a CSE student as the winner".
Predicates:

- is-contest(x) holds iff event x is a contest
- CSE-student(x) holds iff person x is a CSE student
- won(x, y) holds iff event x has a person y as the winner

Solution: Ambiguous sentence. Could be interpreted as either:
$\forall$ c $\exists$ s is-contest(c) $\implies$ (CSE-student(s) $\wedge$ won(c,s)) OR
$\exists$ s $\forall$ c is-contest(c) $\implies$ (CSE-student(s) $\wedge$ won(c,s)).

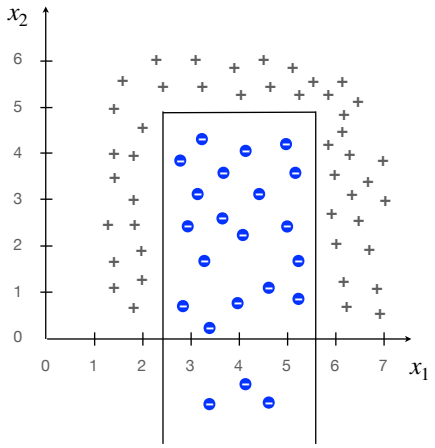
Grading guide

$\frac{1}{2}$ point for each logical form.

(D) Briefly explain why we use weak learners for boosting. **[1 pts]**

Solution: To prevent overfitting; the boosted classifier grows in complexity with each training iteration. So that it is not too-variant in unseen data.

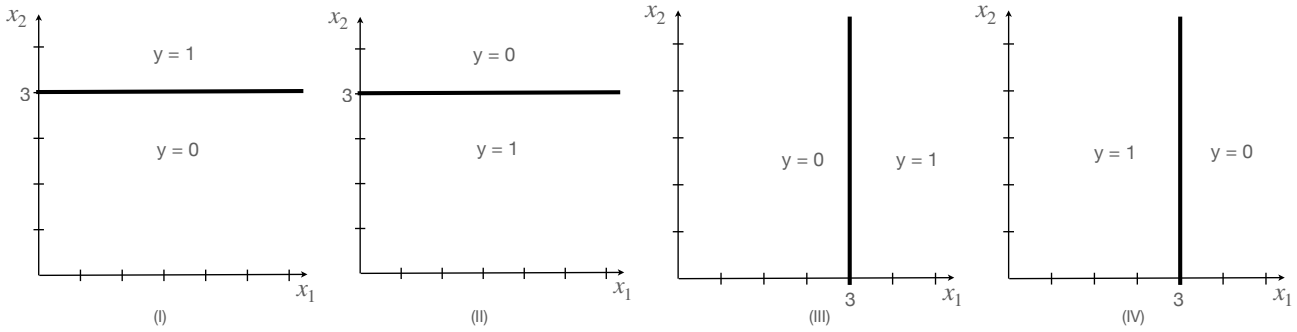
(E) The depth of a decision tree is the number of edges in a longest path from the root to a leaf. Draw a dataset with 2-dimensional points and binary labels where decision trees of depth 3 will perfectly separate the datapoints but decision trees of depth 2 will not. (Consider decision trees with continuous attributes, where you can split on an attribute x based on a threshold τ, i.e. $x \leq \tau$ and $x > \tau$. Note that the same attribute can be used multiple times with different thresholds.) **[2 pts]**



(F) Say you train a logistic regression classifier on 2-dimensional data and the final hypothesis function is:

$$h_{\mathbf{w}}(x) = \sigma(w_0 + w_1 x_1 + w_2 x_2)$$

where $w_0 = 3$, $w_1 = 0$, $w_2 = -1$. Which of the following is the correct decision boundary for $h_{\mathbf{w}}(x)$? Justify. Write sigmoid expression and put x2=0 and see the probability. **[1 pts]**



Solution: Answer is (II). Cannot be (III) or (IV) because the decision boundary does not depend on x_1. $h_{\mathbf{w}}(x)$ is lesser than 0.5 when $x_2 > 3$ (and hence $y = 0$). This rules out (I).

(G) Consider the following two claims about CNNs:

i. CNNs are capable of identifying an object in an image, regardless of how it is translated (i.e., shifted vertically or horizontally), even when the training set only contains images where that object remains in a single position.

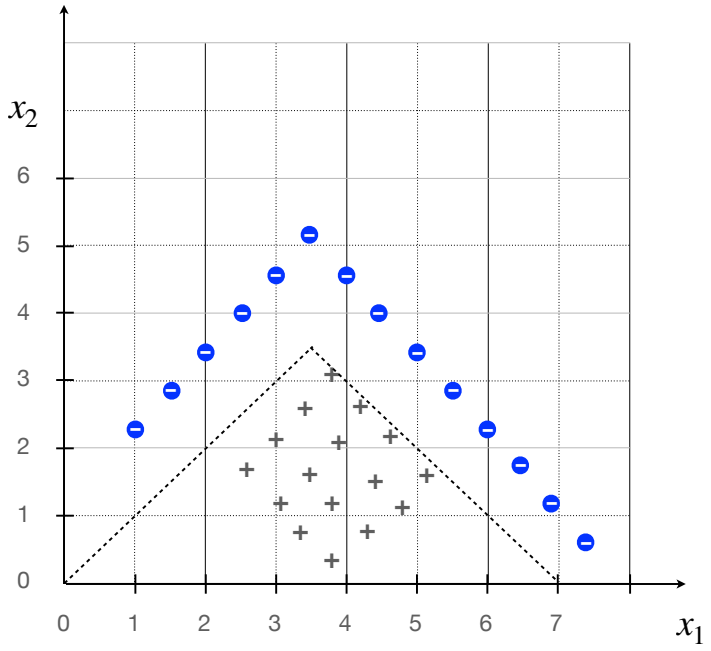
ii. CNNs are capable of identifying an object in an image, regardless of how it is rotated, even when the training set only contains images where that object remains in a single orientation.

Comment on whether claims (i) and (ii) are true or false and briefly justify. **[1 pts]**

Solution: Claim (i) is true because of shared kernel weights, but claim (ii) is false since kernels are not rotation-invariant.

Problem 2: Mixed Bag (II) [13 points]

(A) Consider the following binary-labeled dataset with positive and negative labels marked as pluses and minuses (within circles), respectively. You train a feedforward neural network on this data, with one hidden layer consisting of two nodes with ReLU activations, and one output node with the sign activation function. Draw this network along with the weights and biases needed to perfectly classify the given data. **[3 pts]**



Solution: Consider a network with the specified structure defined by the following function:

$$f(x_1, x_2) = \text{sign}(-\text{ReLU}(x_1 + x_2 - 7) - \text{ReLU}(x_2 - x_1))$$

If $f(x_1, x_2) = 0$, then we classify the point as a "+", and "-" otherwise. The decision boundary from this neural network is given by the thick, dashed lines in the figure above.

(B) Consider $\mathbf{X} \in \mathbb{R}^{d \times n}$ to be a design matrix consisting of n mean-centered datapoints. Recall that the principal components in PCA can be computed using an eigendecomposition of the covariance matrix of the data, $\mathbf{S} = \frac{1}{n}\mathbf{X}\mathbf{X}^T$. $\mathbf{S}$ can also be decomposed as:

$$\mathbf{S} = \mathbf{Q}\Lambda\mathbf{Q}^T$$

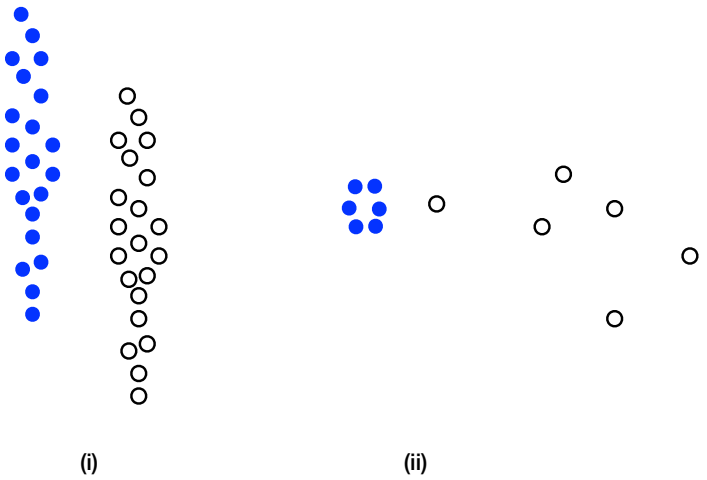
where $\mathbf{Q}$ is an orthogonal matrix and Λ is a diagonal matrix. The projection matrix $\mathbf{U} \in \mathbb{R}^{d \times k}$ for PCA will then be the first k columns of $\mathbf{Q}$. Then, the low-dimensional projection of the data would be $\mathbf{Z} = \mathbf{U}^T\mathbf{X}$. Show that the coordinates of the projected datapoints are uncorrelated with each other; that is, the covariance matrix of $\mathbf{Z}$ is a diagonal matrix. **[2 pts]**

Solution: The covariance matrix of $\mathbf{Z}$: $\frac{1}{n}\mathbf{Z}\mathbf{Z}^T = \mathbf{U}^T\left(\frac{1}{n}\mathbf{X}\mathbf{X}^T\right)\mathbf{U} = \mathbf{U}^T\mathbf{S}\mathbf{U} = \mathbf{U}^T\mathbf{Q}\Lambda\mathbf{Q}^T\mathbf{U} == \begin{pmatrix}\mathbf{I} & \mathbf{0}\end{pmatrix}\Lambda\begin{pmatrix}\mathbf{I} \\ \mathbf{0}\end{pmatrix}$. The top left $k \times k$ block of this covariance matrix is diagonal, which means that the coordinates of the projected datapoints are uncorrelated with each other.

Grading guide

Correct expression for the covariance matrix of $\mathbf{Z}$ is $\frac{1}{2}$ point.

(C) Consider the 2D points below grouped into two clusters ($k = 2$) denoted by filled and unfilled points. For the clusterings shown in figures (i) and (ii) below, comment on whether k-means clustering or Gaussian mixture models or neither could have resulted in these cluster assignments. Briefly justify. **[2 pts]**



Solution: k-means could not resulted in either of the cluster assignments. k-means results in roughly spherical clusters, and there's no cluster mean initialization that would yield the given cluster assignments. GMMs could result in both the cluster assignments. Picking a mixture of two Gaussians with the correct means and covariance matrices can yield the given cluster assignments.

Grading guide

$\frac{1}{2}$ point each for correct answers wrt k-means/GMMs for figures (i) and (ii).

(D) Recall that the soft-margin SVM can also be written as minimizing an L2-regularized hinge loss. That is,

$$\hat{y} = \mathbf{w}^T\mathbf{x} + b$$
$$\mathcal{L}_h(\hat{y}, y) = \max(0, 1 - y\hat{y})$$
$$\mathcal{L}(\mathbf{w}, b) = \frac{\|\mathbf{w}\|^2}{2} + \frac{C}{N}\sum_{i=1}^{N}\mathcal{L}_h(\hat{y}_i, y_i)$$

where $y \in \{-1, +1\}$. What is $\frac{\partial \mathcal{L}}{\partial \mathbf{w}}$? Show your work. Your final expression should not contain any unexpanded (partial) derivatives. **[2 pts]**

Solution:

On derivative application, coefficient matrix gets transposed.

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} + \frac{C}{N}\sum_{i=1}^{N}\frac{\partial \mathcal{L}_h^i}{\partial \mathbf{w}}$$
$$\frac{\partial \mathcal{L}_h^i}{\partial \mathbf{w}} = \frac{\partial \mathcal{L}_h^i}{\partial \hat{y}}\mathbf{x}$$
$$\frac{\partial \mathcal{L}_h^i}{\partial \hat{y}} = \begin{cases} -y_i & \text{if } 1 - y_i\hat{y}_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

Grading guide

1 point for the correct expression of $\frac{\partial \mathcal{L}}{\partial \mathbf{w}}$ using $\frac{\partial \mathcal{L}_h^i}{\partial \mathbf{w}}$, and 1 point for the correct expansion of $\frac{\partial \mathcal{L}_h^i}{\partial \mathbf{w}}$.

(E) Consider a linear regression model in one dimension with no bias terms, $y = wx$ where y and x are scalars. Weight w is initialized to 0. Consider a dataset with a single point $x = 1$, $y = 1$ and the loss function is mean squared error. Compute the value of the weight w for two steps of gradient descent with momentum. Show your work. Recall the expressions for gradient descent with momentum:

$$\mathbf{v}_t \leftarrow \beta\mathbf{v}_{t-1} + \eta\mathbf{g}_t$$
$$\mathbf{w}_t \leftarrow \mathbf{w}_{t-1} - \mathbf{v}_t$$

where $\mathbf{v}$, $\mathbf{g}$ and $\mathbf{w}$ are momentum, gradient and weight terms, respectively. Set $\beta = 0.99$, $\eta = 0.5$ in your computations. **[2 pts]**

Solution:

1. $\frac{\partial(wx-y)^2}{\partial w} = 2(wx - y)x = -2.$
   $\mathbf{v}_0 = 0, \mathbf{v}_1 = 0.99 \times 0 + 0.5 \times -2 = -1$
   $\mathbf{w}_1 = 1$
2. $\mathbf{g}_1 = 2(w_1x - y)x = 0$
   $\mathbf{v}_2 = 0.99\mathbf{v}_1 = -0.99$
   $\mathbf{w}_2 = 1 + 0.99 = 1.99$

Grading guide

1 point each for correct computations in the two steps outlined in the solution. Not assuming $\mathbf{v}_0 = 0$, but rest being correct, will also fetch you full points.

(F) Consider an MDP with five states $\{0, 1, 2, 3, 4\}$ where 4 is the start state and 0 is the goal state. The reward $R(s, a, s') = (s' - s)^2$ for all (s, a, s'), and the discount factor $\gamma = 0.5$. In states $i \geq 1$, you can take action NEXT (N) and deterministically land in the state just one below. That is, the transition probability $T(i-1|N, i) = 1$. In states $i \geq 2$, you can also take an action SKIP (S) with the following transitions, $T(i-2|S, i) = T(i|S, i) = \frac{1}{2}$. Compute the value of $V^*(2)$. Show your work. **[2 pts]**

Recall the Bellman equation for $V^*(s)$:

$$V^*(s) = \max_a \sum_{s'} T(s'|s, a) \left(R(s, a, s') + \gamma V^*(s')\right)$$

Solution:

$$\begin{aligned}
V^*(0) &= 0 \\
V^*(1) &= \max\{1 + \gamma V^*(0)\} \\
&= 1 \quad \text{either one step down or took two steps down (SKIP), if chose one step down, no further recursion possible, but if V*(2) can be calculated if checked condition wise.} \\
V^*(2) &= \max\{1 + \gamma V^*(1), \frac{1}{2}(4 + \gamma V^*(0)) + \frac{1}{2}\gamma V^*(2)\} \\
&= \max\{\frac{3}{2}, 2 + \frac{1}{4}V^*(2)\} \\
&= \max\{\frac{3}{2}, 2 + \frac{2}{3}\} \quad \text{Comment: get value } \frac{2}{3} \text{ by solving for } V^*(2) \text{ in } V^*(2) = 2 + \frac{1}{4}V^*(2) \\
&= \frac{8}{3}
\end{aligned}$$

Grading guide

1 point each for correct computations of $V^*(1)$ and $V^*(2)$.

Problem 3: Search [9 points]

This problem involves four knights (♘), two black and two white, placed on a standard chessboard with 64 squares. Note that the two white knights are indistinguishable from each other, and so are the two black knights. The goal is, given an initial configuration (position of the four knights) and a final configuration of the board, to find a sequence of legal moves that will take the board from the initial configuration to the final one. In a legal move a single knight can be moved to any of the squares it can reach according to the rules of chess (the exact rules are not important for this problem), as long as that square is not already occupied. A white piece moves first, and then moves should alternate between black and white pieces.

Consider using the A* algorithm to find a short sequence of legal moves that takes the board from the initial configuration to the final configuration.

(A) Clearly describe the state space for this search problem. Also describe the actions (edges) and their costs.

How large is your state space? (You can leave your answer as a product of integers.) What is the maximum number of actions available at a state? (The maximum number of squares a knight can reach in a single move is 8.) **[5 pts]**

Solution:

- State space is a set $(\{W_1, W_2\}, \{B_1, B_2\}, b)$ where $b \in \{0, 1\}$. Here, W_i and B_i are (x,y) coordinates on the board where the knights lie. The bit indicates who moves next.
- Actions are all the squares a single (white or black) knight can move to.
- Cost for each action should be 1.
- Size of the state space: $\binom{64}{2} \times \binom{62}{2} \times 2$.
- Depending on the bit, there can be at max 16 actions available at a state.

Grading guide

1 point for each item above. If one of the points above is subsumed in another answer, full points will be granted.

(B) Let $K(S_1, S_2)$ denote the smallest number of moves needed by a knight to move from a square S_1 to a square S_2.

Below, (W_1, W_2) denotes the white knights' squares in the current configuration (sorted according to some predetermined scheme), and (B_1, B_2) denotes those of the black knights; similarly, $(W_1^*, W_2^*, B_1^*, B_2^*)$ denotes the knights' positions in the final configuration. For each of the following heuristic costs, state if it is guaranteed to be consistent or not (irrespective of the sorting scheme). Justify your answer.

i. $K(W_1, W_1^*) + K(W_2, W_2^*) + K(B_1, B_1^*) + K(B_2, B_2^*)$ **[2 pts]**

Solution: $K(W_1, W_2^*) + K(W_2, W_1^*)$ could be less than $K(W_1, W_1^*) + K(W_2, W_2^*)$, since the white and black pieces are indistinguishable. So, not admissible. Since consistency implies admissibility, not admissible implies not consistent.

ii. $\min\big(K(W_1, W_1^*) + K(W_2, W_2^*), K(W_2, W_1^*) + K(W_1, W_2^*)\big)$
$+ \min\big(K(B_1, B_1^*) + K(B_2, B_2^*), K(B_2, B_1^*) + K(B_1, B_2^*)\big)$ **[2 pts]**

Solution: This is the exact solution to a relaxed problem where pieces can overlap. So, it is admissible. To prove it's consistent, consider an $h(s)$ is the best among all paths from s to the goal state in the relaxed problem. Say we add some additional constraints that you should reach an s' and further till you reach s' you cannot use the relaxation. Then the best path would be $h(s') + \text{cost}(s, s')$ where $\text{cost}(s, s')$ is the actual cost in the original problem to take you from s to s', and $h(s')$ is the best path from s' to the goal state in the relaxed problem. Then, $h(s') + \text{cost}(s, s') \geq h(s)$ since every path in the relaxed set with constraints is also allowed in the relaxed set without constraints. Thus, the heuristic is consistent.

Problem 4: Boosting [9 points]

Recall that in a boosting algorithm we run several iterations, such that in the i^{th} iteration we 1) train a weak learner on a weighted dataset (initialized uniformly) to obtain a hypothesis h_i 2) compute the weighted error using the weak learner 3) increase the weights of those training examples that were misclassified. The hypothesis output by the boosted classifier after n iterations is given by $h^*(\mathbf{x}) = \text{sign}(\sum_{i=1}^{n} \alpha_i h_i(\mathbf{x}))$ for some non-negative weights α_i. For this problem, the exact reweighting scheme and how α_i are defined are not important.

For $S \subseteq \{1, \ldots, d\}$ and $b \in \{0, 1\}$, let $h_{S,b}(\mathbf{x}) = (-1)^b \prod_{i \in S} (-1)^{x_i}$ where $\mathbf{x} = (x_1, \ldots, x_d) \in \{0, 1\}^d$. Suppose you have access to a weak learner which returns a hypothesis of the form $h_{S,b}$ when given a set of labelled points $(\mathbf{x}_i, y_i)$, $y_i \in \{+1, -1\}$ and weights $D(i)$, which minimizes the weighted error.

Consider the following toy dataset:

0	0	0	+1
0	0	1	−1
0	1	0	−1
1	0	0	−1
0	1	1	+1
1	1	0	−1
1	1	1	+1

(A) Show that there is no function $h_{S,b}$ such that $h_{S,b}(\mathbf{x}_i) = y_i$ for all $(\mathbf{x}_i, y_i)$ in the above dataset. **[3 pts]**

Solution: A function $h_{S,b}$ that correctly predicts all $(\mathbf{x}_i, y_i)$ in the given dataset should predict the first four datapoints correctly. The hypothesis $h_{S,b}$ with $S = \{1, 2, 3\}, b = 0$ is the *only one* that correctly predicts the first four datapoints. However, it errs on the last two datapoints. Thus, there is no function $h_{S,b}$ such that $h_{S,b}(\mathbf{x}_i) = y_i$ for all $(\mathbf{x}_i, y_i)$.

(B) Show that when a boosting algorithm using the given weak learner is run on the above dataset with uniform weighting, after the first iteration, it will return a hypothesis which is wrong on only one datapoint. (In fact, this hypothesis is unique. You can use this fact for the next part, without proof.) **[2 pts]**

Solution: The fact that there is no function $h_{S,b}$ such that $h_{S,b}(\mathbf{x}_i) = y_i$ has already been shown in part A. The hypothesis $h_{S,b}$ with $S = \{2, 3\}, b = 0$ only errs on $(1, 0, 0)$.

Grading guide

Full points for identifying the correct (unique) hypothesis that is wrong on only one datapoint.

(C) Show that when the above boosting algorithm is continued, it will not have zero training error after two iterations. **[4 pts]**

Solution: Suppose the training error becomes 0 in two iterations. That is, there are weights α, β and hypotheses $h_{S,b}$ (from the first round) and $h_{S',b'}$ (from the second round) such that $h^* = \alpha h_{S,b} + \beta h_{S',b'}$ correctly classify the given 7 datapoints. We will use the fact that $h_{S,b}$ is wrong at one datapoint $(1,0,0)$ and there the correct solution is -1.

We need the following observation: For any two distinct sets S, S', and any bits b, b', $h_{S,b}$ and $h_{S',b'}$ differ on exactly 4 of the 8 points.

We consider 3 cases:

- $\alpha > \beta$: In this case, $\text{sign}(h^*(x)) = \text{sign}(h_{S,b}(x))$ in all 8 points. This errs on one datapoint.
- $\alpha < \beta$: In this case $\text{sign}(h^*(x)) = \text{sign}(h_{S',b'}(x))$ in all 8 points. But this differs from $h_{S,b}$ in at least 3 of the given 7 datapoints, and hence will err on at least two datapoints.
- $\alpha = \beta$: In this case, $\text{sign}(h^*(x))$ matches $\text{sign}(h_{S,b}(x))$ in all the points where $h_{S,b}(x) = h_{S',b'}(x)$, and becomes $+1$ in the other points. That is, $\text{sign}(h^*(x))$ can flip -1 to $+1$ in $\text{sign}(h_{S,b}(x))$, but not the other way around. However, the correct solution differs from $h_{S,b}$ in one point (namely $x = (1,0,0)$) where it is -1. Hence $\text{sign}(h^*(x))$ cannot equal the correct solution.

Problem 5: Pushy Perceptron [8 points]

Recall the online perceptron algorithm for binary classification:

- At each round t, given a training instance $(\mathbf{x}_t, y_t)$, where $\mathbf{x}_t \in \mathbb{R}^d$ and the label $y_t \in \{-1, 1\}$, the weight vector weight vector $\mathbf{w}^{t+1}$ is computed from $\mathbf{w}^t$ using the following update rule:

$$\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t + y_t \mathbf{x}_t \qquad \text{if } y_t(\mathbf{w}^t \cdot \mathbf{x}_t) \leq 0$$

- If the final learned weight is $\mathbf{w}^*$, for a test example $\mathbf{x}$, $\text{sign}(\mathbf{w}^* \cdot \mathbf{x})$ is the final prediction.

Consider the following variant of the online perceptron algorithm that we'll call the Pushy Perceptron (PP). PP has the following weight update rule:

$$\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t + \alpha_t y_t \mathbf{x}_t, \qquad \alpha_t = \frac{\ell_t}{\|\mathbf{x}_t\|^2}$$

where $\ell_t = \mathcal{L}_{\text{hinge}}(\mathbf{x}_t, y_t; \mathbf{w}^t)$, in turn defined as

$$\mathcal{L}_{\text{hinge}}(\mathbf{x}, y; \mathbf{w}) = \begin{cases} 0 & y(\mathbf{w} \cdot \mathbf{x}) \geq 1 \\ 1 - y(\mathbf{w} \cdot \mathbf{x}) & \text{otherwise} \end{cases} \tag{1}$$

(A) Describe how we can easily construct a kernelized version of PP. That is, given a valid kernel function $K(\mathbf{x}, \mathbf{x}')$, write down the kernelized prediction rule for a test instance $\mathbf{x}$ using K. **[1 pts]**

Solution: $\mathbf{w}^t = \sum_{i=1}^{t-1} \alpha_t y_t \mathbf{x}_i$. Therefore, for a test instance $\mathbf{x}$, kernelized version can be written as $\mathbf{w}^t \cdot \mathbf{x} = \sum_{i=1}^{t-1} \alpha_t y_t (\mathbf{x}_i \cdot \mathbf{x}) = \sum_{i=1}^{t-1} \alpha_t y_t K(\mathbf{x}_i, \mathbf{x})$. Using the kernel trick, inner products $\mathbf{x}_i \cdot \mathbf{x}$ can be replaced by a kernel invocation $K(\mathbf{x}_i, \mathbf{x})$.

(B) Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ be a sequence of training instances presented to PP as inputs, where $\mathbf{x}_t \in \mathbb{R}^d$, $y_t \in \{-1, 1\}$ in round t. Prove the following bound for any $\mathbf{u} \in \mathbb{R}^d$:

$$\sum_{t=1}^{T} \alpha_t \left(2\ell_t - \alpha_t \|\mathbf{x}_t\|^2 - 2\ell_t^{\mathbf{u}}\right) \leq \|\mathbf{u}\|^2 \tag{2}$$

where $\alpha_t = \frac{\ell_t}{\|\mathbf{x}_t\|^2}$, $\ell_t = \mathcal{L}_{\text{hinge}}(\mathbf{x}_t, y_t; \mathbf{w}^t)$ and $\ell_t^{\mathbf{u}} = \mathcal{L}_{\text{hinge}}(\mathbf{x}_t, y_t; \mathbf{u})$. Assume that $\mathbf{w}_1$ is initialized to the all-zeros vector.

Hint: Consider the quantity $\delta_t = \|\mathbf{w}_t - \mathbf{u}\|^2 - \|\mathbf{w}_{t+1} - \mathbf{u}\|^2$. Prove the bound in (2) by showing that the LHS and RHS are a lower bound and an upper bound of $\sum_{t=1}^{T} \delta_t$, respectively. **[5 pts]**

Solution:

$$\begin{aligned}\sum_{t=1}^{T} \delta_t &= \sum_{t=1}^{T} \|\mathbf{w}_t - \mathbf{u}\|^2 - \|\mathbf{w}_{t+1} - \mathbf{u}\|^2 \\ &= \|\mathbf{w}_1 - \mathbf{u}\|^2 - \|\mathbf{w}_{T+1} - \mathbf{u}\|^2\end{aligned}$$

Since $\mathbf{w}_1$ is the all-zeros vector and $\|\mathbf{w}_{T+1} - \mathbf{u}\|^2$ is non-negative, we have the following upper bound:

$$\sum_{t=1}^{T} \delta_t \leq \|\mathbf{u}\|^2 \tag{3}$$

To construct the lower bound, we have:

This is the important step to consider. This equation solves the problem.

$$\begin{aligned}\delta_t &= \|\mathbf{w}_t - \mathbf{u}\|^2 - \|\mathbf{w}_{t+1} - \mathbf{u}\|^2 \\ &= \|\mathbf{w}_t - \mathbf{u}\|^2 - \|\mathbf{w}_t - \mathbf{u} + \alpha_t y_t \mathbf{x}_t\|^2 \\ &= \|\mathbf{w}_t - \mathbf{u}\|^2 - \left(\|\mathbf{w}_t - \mathbf{u}\|^2 + \alpha_t^2 \|\mathbf{x}_t\|^2 + 2\alpha_t y_t (\mathbf{w}_t - \mathbf{u}) \cdot \mathbf{x}_t\right) \\ &= -2\alpha_t y_t (\mathbf{w}_t - \mathbf{u}) - \alpha_t^2 \|\mathbf{x}_t\|^2 \\ &\geq 2\alpha_t \left((1 - \ell_t^{\mathbf{u}}) - (1 - \ell_t)\right) - \alpha_t^2 \|\mathbf{x}_t\|^2 \\ &= \alpha_t \left(2\ell_t - \alpha_t \|\mathbf{x}_t\|^2 - 2\ell_t^{\mathbf{u}}\right)\end{aligned} \tag{4}$$

The inequality above holds because if $\ell_t > 0$ then $\ell_t = 1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)$ i.e., $y_t(\mathbf{w}_t \cdot \mathbf{x}_t) = 1 - \ell_t$. Also from the loss definition, $\ell_t^{\mathbf{u}} \geq 1 - y_t(\mathbf{u}_t \cdot \mathbf{x}_t)$. From Equations (3) and (4), we see that the RHS and LHS of Eqn (2) are an upper bound and a lower bound of $\sum_{t=1}^{T} \delta_t$, thus proving $\sum_{t=1}^{T} \alpha_t \left(2\ell_t - \alpha_t \|\mathbf{x}_t\|^2 - 2\ell_t^{\mathbf{u}}\right) \leq \|\mathbf{u}\|^2$.

Grading guide

1.5 points for the upper bound in Eqn (3), 3.5 points for the lower bound in Eqn (4).

(C) Analogous to the mistake bound for the original Perceptron algorithm, we can prove a loss bound for PP in the fully separable case. Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ be a sequence of training instances presented to PP as inputs, where $\mathbf{x}_t \in \mathbb{R}^d$, $y_t \in \{-1, 1\}$ and $\|\mathbf{x}_t\| \leq R$ for all t. Suppose $\mathbf{u} \in \mathbb{R}^d$ is such that $\ell_t^{\mathbf{u}} = \mathcal{L}_{\text{hinge}}(\mathbf{x}_t, y_t; \mathbf{u}) = 0$ for all t. Then, using Equation (2) from part (B), prove the following loss bound:

$$\sum_{t=1}^{T} \ell_t^2 \leq \|\mathbf{u}\|^2 R^2$$

where $\ell_t = \mathcal{L}_{\text{hinge}}(\mathbf{x}_t, y_t; \mathbf{w}^t)$. **[2 pts]**

Solution: Given that $\ell_t^{\mathbf{u}} = 0$ for all t, using Equation (2) implies:

$$\sum_{t=1}^{T} \alpha_t \left(2\ell_t - \alpha_t \|\mathbf{x}_t\|^2\right) \leq \|\mathbf{u}\|^2$$

Substituting $\alpha_t = \frac{\ell_t}{\|\mathbf{x}_t\|^2}$ in the LHS above, we get:

$$\sum_{t=1}^{T} \frac{\ell_t}{\|\mathbf{x}_t\|^2} \leq \|\mathbf{u}\|^2$$

Given that $\|\mathbf{x}_t\| \leq R$ for all t, we get the desired loss bound for PP: $\sum_{t=1}^{T} \ell_t^2 \leq \|\mathbf{u}\|^2 R^2$.