ECE 661 Spring 2025 Final Exam

Introduction to Machine Learning for Engineers
Prof. Gauri Joshi and Prof. Carlee Joe-Wong
Friday, May 2nd, 2025
10:00am - 1:00pm PT/1:00pm - 4:00pm ET/7:00pm - 10:00pm CAT

Andrew ID ______________________ **Name** ____________________________

Instructions

a. If a problem asks you which of its choices is TRUE, you should treat choices that may be either true or false as FALSE.

b. **Unless otherwise stated, only one option is correct in each multiple-choice question.** No partial credit will be given for multiple-choice with only one option correct, or for true/false questions.

c. For descriptive questions, make sure to explain your answers and reasoning. We will give partial credit for wrong answers if portions of your reasoning are correct. Conversely, correct answers accompanied by incomplete or incorrect explanations may not receive full credit.

d. You are allowed two **physical** US-letter or A4 sized cheat sheets (two-sided). No other notes or material (aside from blank pieces of scratch paper) may be used.

e. You may only use a pen/pencil, eraser, and scratch paper. The backside of each sheet in the exam can also be used as scratch paper. If you do not wish for us to grade your scratch work, please clearly indicate which parts of your work we should ignore.

f. Calculators are not necessary and are not permitted.

g. If you would like to ask a clarification question during the exam, raise your hand and an instructor or TA will come over. Note that we will not help you answer the questions but can give clarifications.

Problem	**Type**	**Points**
1	Honor Pledge	0
2-9	True/False	8 (1 pt each)
10-19	Multiple Choice	20 (2 pts each)
20	Descriptive	6
21	Descriptive	6
22	Descriptive	8
23	Descriptive	7
24	Descriptive	6
25	Descriptive	8
26	Descriptive	6
Total		75 points

Problem 1: To affirm that you did not cheat on the exam, please write out the below statement. Sign your name beneath it. **Failure to do so will be taken as a sign that you have cheated on the exam.**

I pledge my honor that I neither gave nor received unauthorized assistance on this examination.

1 True or False (8 questions, 1 pt each)

Problem 2: *[1 points]* Consider the linear regression problem

$$\min_{\mathbf{w}\in\mathbb{R}^d} \|\mathbf{y}-\mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2 ,$$

where $\mathbf{X} \in \mathbb{R}^{n\times d}$ is the design matrix of input features, $\mathbf{y} \in \mathbb{R}^n$ is the vector of target outputs, $\mathbf{w} \in \mathbb{R}^d$ is the parameter vector to be learned, and $\lambda \geq 0$ is a regularization parameter. Suppose that one of the features in $\mathbf{X}$ is an identical copy of another feature. Taking $\lambda > 0$ ensures that the solution is given by $\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$.

◯ True

◯ False

Solution: False. If regularization is used (ridge regression), the solution is given by

$$\mathbf{w}^* = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$$

not by the unregularized formula. Regularization is required when features are linearly dependent to make the matrix invertible, but the closed-form solution changes accordingly.

Problem 3: *[1 points]* Random Forests subsample the original dataset for every learned decision tree but **do not** subsample the features when deciding each split of each tree.

◯ True

◯ False

Solution: False. Random Forests subsample both the data and the features during tree generation.

Problem 4: *[1 points]* A deep neural network with multiple fully connected layers, but without activation functions acting on any neuron, has the same representational power as a single-layer linear model with the same input features.

◯ True

◯ False

Solution: True. Without non-linear activation functions, each layer performs a linear transformation. The composition of linear transformations is itself linear, so the entire network collapses to a single linear map. This makes it equivalent to a one-layer linear model in terms of representational power.

Problem 5: *[1 points]* Mini-batch Stochastic Gradient Descent (SGD) has slower per-iteration runtime and has more noisy updates relative to full-batch Gradient Descent (GD).

◯ True

◯ False

Solution: False. mini-batch SGD will be faster per-iteration than vanilla GD, but it does introduce more noise in each gradient step.

Problem 6: *[1 points]* When using the EM algorithm to learn a Gaussian mixture model (GMM), the E-step computes the probability of each data point belonging to all clusters. When using the EM algorithm to learn a K-means model, the E-step assigns each data point to exactly one cluster.

◯ True

◯ False

Solution: True. r_{nk} in GMM are soft, but are hard in K-Means.

Problem 7: *[1 points]* In a hard-margin SVM (support vector machine) model, the margin size is determined solely by the distance between the decision boundary and the support vectors, regardless of how many support vectors there are.

◯ True

◯ False

Solution: True. In a hard-margin SVM, the margin is defined by the support vectors, the closest points to the decision boundary. The number of support vectors can vary depending on the data distribution, but the margin size is determined by their distance from the hyperplane, not by their count.

Problem 8: *[1 points]* Increasing the discount factor γ in a Markov Decision Process causes the agent to value long-term rewards more.

◯ True

◯ False

Solution: True.

Problem 9: *[1 points]* In Naïve Bayes without smoothing, if a feature value is never observed in the training data for a class c, the model will assign a probability of zero to any test sample containing that feature value for belonging to class c.

◯ True

◯ False

Solution: True. Without smoothing (like Laplace smoothing), Naïve Bayes assigns zero probability to unseen feature values, which zeroes out the entire class probability.

2 Multiple Choice (10 questions, 2 pts each)

Problem 10: *[2 points]* Which of the following statements correctly distinguishes Maximum Likelihood Estimation (MLE) and Maximum A Posteriori (MAP) estimation? **More than one option can be correct.**

☐ MAP estimation incorporates prior beliefs about the parameters, whereas MLE does not.

☐ MLE can be viewed as a special case of MAP when the prior is uniform.

☐ MAP always yields more accurate estimates than MLE, especially on large datasets.

☐ In MAP estimation, we minimize the sum of the negative log-likelihood and an additional term derived from the prior distribution.

Solution: The correct answers are (A), (B), and (D).

☐ True; MAP uses a prior $P(\theta)$, while MLE does not.
☐ True; When the prior $P(\theta)$ is uniform, MAP reduces to MLE.

☐ False; The accuracy of the MAP estimate depends on the accuracy of the prior, and it does not guarantee higher accuracy than MLE.

☐ True; MAP maximizes posterior, which translates to minimizing $-\log$ likelihood plus a regularization term based on $-\log$ prior.

Problem 11: *[2 points]* Consider a Naïve Bayes classifier f and a ham/spam dataset $\mathcal{D}$ consisting of both spam and ham documents, where each document is a sequence of words. We will use the bag-of-words method to calculate the required parameters for the Naïve Bayes Classifier that distinguishes ham from spam documents, and we will not use any smoothing. Which of the below changes to the dataset would also change the learned Naïve Bayes parameters? **More than one option can be correct.**

☐ Duplicating each document in the dataset exactly one time.

☐ Changing the order of the words in each document.

☐ Duplicating every word in each document exactly three times.

☐ Adding the word 'start' to the beginning of every document when 'start' already existed in the bag-of-words vocabulary.

Solution: D

☐ False; Duplicating each document in the dataset exactly one time only changes the objective function by a constant scaling factor
☐ False; Changing the order of the words in each document does not change the objective function

☐ False; Duplicating every word in each document exactly three times only changes the objective function by a constant scaling factor

☐ True: Adding the word 'start' to the beginning of every document when 'start' already existed in the vocabulary changes the objective function

Problem 12: *[2 points]* The self-attention mechanism used in transformers calculates pairs of interactions between tokens using the formula

$$\text{softmax}\left(\frac{\mathbf{QK}^\top}{\sqrt{d}}\right) \tag{1}$$

where $\mathbf{Q} \in \mathbb{R}^{n \times d}, \mathbf{K} \in \mathbb{R}^{n \times d}$ and n is the number of tokens in the input sequence and d is the embedding dimension. What is the computational complexity of performing self-attention with an input sequence n tokens long?

◯ $O(n)$

◯ $O(n \log(n))$

◯ $O(1)$

◯ $O(n^2)$

Solution: D. self-attention is quadratic operation since it has to calculate $O(n^2)$ token interactions from the input sequence.

Problem 13: *[2 points]* Suppose that you wish to train a regularized neural network model on a dataset $\mathcal{D}$, i.e., you wish to find the parameters that minimize

$$\sum_{(\mathbf{x},y)\in\mathcal{D}} \ell(f(\mathbf{w},\mathbf{x}),y) + \lambda \left\|\mathbf{w}\right\|_2^2,$$

where $f(\mathbf{w},\mathbf{x})$ denotes the neural network model's output for parameters $\mathbf{w}$ and input features $\mathbf{x}$, and $\ell(\cdot)$ is the loss function. You have three candidate values of λ, which are given by $\lambda_1, \lambda_2, \lambda_3$.
You use 5-fold cross-validation to find the optimal value of λ. Let $l_j^{(k)}$ denote the **training** loss when optimizing over the kth fold using regularization weight λ_j, i.e.,

$$l_j^{(k)} = \min_{\mathbf{w}} \sum_{(\mathbf{x},y)\in\mathcal{D}_{\neq k}} \ell(f(\mathbf{w},\mathbf{x}),y),$$

where $\mathcal{D}_{\neq k}$ denotes all folds except the kth fold. You observe that the average training loss across the folds, $\frac{1}{5}\sum_{k=1}^{5} l_j^{(k)}$, is lowest when $j=1$, but that the variance $\frac{1}{5}\sum_{k=1}^{5}\left(l_j^{(k)} - \frac{1}{5}\sum_{m=1}^{5} l_j^{(m)}\right)^2$ is the lowest when $j=2$. Which of the following statements gives the most likely order of $\lambda_1, \lambda_2, \lambda_3$?

- ◯ $\lambda_1 < \lambda_2 < \lambda_3$
- ◯ $\lambda_2 < \lambda_3 < \lambda_1$
- ◯ $\lambda_1 < \lambda_3 < \lambda_2$
- ◯ $\lambda_3 < \lambda_1 < \lambda_2$

Solution: C. A lower value of λ will lead to lower training loss on each fold (i.e., the bias will be lower, since we regularize less), so λ_1 must be the smallest. The variance of the training losses across the folds, however, will be lower when the regularization weight is higher (i.e., more regularization leads to lower variance), so λ_2 must be the largest.

Problem 14: *[2 points]* Suppose that you wish to train a soft-margin support vector machine (SVM) model to solve a binary classification task on a dataset $\mathcal{D}$. Let $\mathbf{w}, b$ denote the parameters of the trained SVM model.
After training the SVM model, you decide to remove outlier data points and retrain the model without them. Which of the following types of data points can be removed without changing the learned model parameters $\mathbf{w}, b$?

- ◯ Misclassified data points.
- ◯ Correctly classified data points $(\mathbf{x}_n, y_n)$ that are far from the decision boundary, i.e., for which $y_n\left[\mathbf{w}^T\mathbf{x}_n + b\right] \geq 1$.
- ◯ Correctly classified data points $(\mathbf{x}_n, y_n)$ that are close to the decision boundary, i.e., for which $0 \leq y_n\left[\mathbf{w}^T\mathbf{x}_n + b\right] \leq 1$.
- ◯ None of the above.

Solution: B. Both A and C represent support vectors, whose removal may change the learned decision boundary.

Problem 15: *[2 points]* Which of the following statements correctly identifies one of the differences between training a fully connected neural network and training a neural network with at least one convolutional layer? **More than one option may be correct.**

☐ We use backpropagation to find the gradients of the fully connected neural network, but we cannot use backpropagation to find the gradients of convolutional layers.

☐ Convolutional layers have fewer parameters than fully connected layers with the same number of input and output neurons, so storing the gradient of each layer requires less memory for CNNs.

☐ Full batch gradient descent is typically used in training fully connected neural networks, in order to minimize noise in the gradient updates, but stochastic gradient descent (SGD) is typically used for convolutional neural networks (CNNs).

☐ Batch normalization can be used in both fully connected neural networks and CNNs to reduce overfitting.

Solution:

☐ False; we use backpropagation for both types of networks.

☐ True; convolutional layers have fewer parameters than fully connected ones of the same size.

☐ False; SGD is typically used to train both types of neural networks.

☐ True; batch normalization can be used for both types of networks.

Problem 16: *[2 points]* An asynchronous SGD (stochastic gradient descent) system has 12 workers. The time each worker takes to compute and send a gradient follows an exponential distribution with a mean of 3 seconds. What is the expected time between two successive gradients received by the parameter server (assuming there is no communication delay)?

◯ 3 seconds

◯ 3/12 seconds

◯ 36 seconds

◯ 12/3 seconds

Solution: Let X be the time to compute and send a gradient. $X \sim \text{Exponential}(\lambda)$. Since we're told $\text{E}(X) = 3$ seconds, then $\lambda = 1/3$. The expected time between two successive gradients received for asynchronous SGD is $\text{E}(X_{1:m})$ where m is the number of workers. Therefore, $\text{E}(X_{1:12}) = 1/12\lambda = 3/12$

Problem 17: *[2 points]* Consider a stochastic K-armed bandit across time horizon T where pulling an arm $i \in [K]$ yields i.i.d. (independent and identically distributed) rewards $r(i)$, which are random variables with means μ_i. Denote the expected reward of the best arm $i^\star$ by $\mu^\star = \max_i \mu_i$ and the number of times arm i is pulled until time-step t by $N(i,t)$. If i_t represents the arm pulled at time-step t, which of the following expressions represents the expected cumulative regret of the bandit, which is defined as $R_T = \sum_{t=1}^{T} (\mathbb{E}[r(i^\star)] - \mathbb{E}[r(i_t)])$? **More than one option can be correct.**

☐ $R_T = T\mu^\star - \sum_{t=1}^{T} \mathbb{E}[r(i_t)]$

☐ $R_T = \sum_{i \neq i^\star} (\mu^\star - \mu_i)$

☐ $R_T = \sum_{i=1}^{K} (\mu^\star - \mu_i)\mathbb{E}[N(i,T)]$

☐ $R_T = \sum_{i=1}^{K} (\mu_i - \mu^\star)N(i,T)$

Solution: A, C. Follows from definition.

Problem 18: *[2 points]* Consider a Markov Decision Process $\mathcal{M}(\mathcal{S}, \mathcal{A}, r, P, \gamma)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, r is the reward function, P is the state transition matrix, and γ is the discount factor. Let π represent a policy, $V^\pi(s)$ the state-value function and $Q^\pi(s, a)$ the state-action value function for policy π. Which of the following statements about $V^\pi(s)$ and $Q^\pi(s, a)$ are TRUE? **More than one option can be correct.**

- ☐ $V^\pi(s) = \gamma \mathbb{E}_{a\sim\pi(\cdot|s)} [Q^\pi(s, a)]$
- ☐ $Q^\pi(s, a) = \mathbb{E} [r(s, a)] + \gamma \mathbb{E}_{s'\sim P(\cdot|s,a)} [V^\pi(s')]$
- ☐ $V^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)} [Q^\pi(s, a)]$
- ☐ $Q^\pi(s, a) = \mathbb{E} [r(s, a)] + \mathbb{E}_{s'\sim P(\cdot|s,a)} [V^\pi(s')]$

Solution: B, C. Follows from definition.

Problem 19: *[2 points]* Which of the following statements about the k-Nearest Neighbours (KNN) algorithm are correct? **More than one option can be correct.**

- ☐ KNN does not require a training phase.
- ☐ KNN requires the value of k to be optimized to avoid overfitting or underfitting.
- ☐ KNN can only be used for classification tasks, not regression.
- ☐ The choice of distance metric (e.g., Euclidean, Manhattan) can significantly impact KNN's performance.

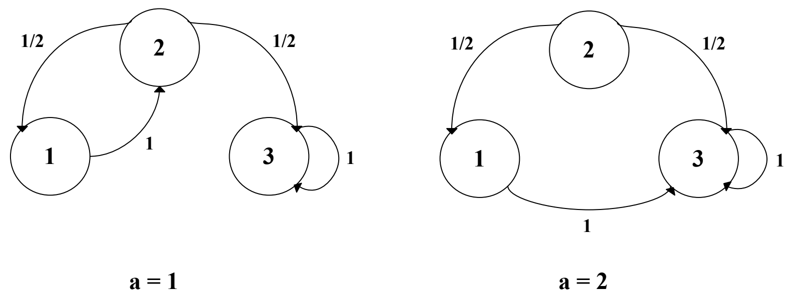
Solution: A, C, D

- ☐ True; it stores the data and makes decisions during prediction.
- ☐ True; small k can overfit, large k can underfit.
- ☐ False; KNN can be used for both.
- ☐ True; different metrics affect neighbor selection.

3 Descriptive (7 questions, 6-8 pts each)

Problem 20: *[6 points]* Consider a Markov Decision Process $\mathcal{M}(\mathcal{S}, \mathcal{A}, r, P, \gamma)$ with three states $\mathcal{S} = \{1, 2, 3\}$, two actions $\mathcal{A} = \{1, 2\}$, and discount factor $\gamma = \frac{1}{2}$. The rewards $\mathbf{r}$ and state transition probabilities $\mathbf{P}$ are given in the matrices below, where we define the (i, j) entry of $\mathbf{r}$, $[\mathbf{r}]_{i,j} = r(s = i, a = j)$, and the (i, j) entry of $[\mathbf{P}_{a=k}]_{i,j} = P(s' = j | s = i, a = k)$. We have also drawn the state transition diagrams coresponding to these transition probabilities.

$$\mathbf{r} = \begin{bmatrix} -1 & -1 \\ 0 & 0 \\ 1/2 & 1/2 \end{bmatrix}, \mathbf{P}_{a=1} = \begin{bmatrix} 0 & 1 & 0 \\ 1/2 & 0 & 1/2 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{P}_{a=2} = \begin{bmatrix} 0 & 0 & 1 \\ 1/2 & 0 & 1/2 \\ 0 & 0 & 1 \end{bmatrix}.$$



(a) *[3 points]* Compute the value function V^π for the policy π where for all $s \in \{1, 2, 3\}$.

$$\pi(\cdot|s) = \begin{cases} 1/2 & \text{if } a = 1 \\ 1/2 & \text{if } a = 2. \end{cases}$$

Hint: Observe that state 3 is an absorbing/terminal state with a probability 1 self-transition. Evaluate the value function of this state first, and use it to compute the value functions of other states.

(b) *[3 points]* Compute the optimal policy for the above MDP $\mathcal{M}$ if the starting state probabilities are

$$P(s_0) = \begin{cases} 1/2 & \text{if } s_0 = 1 \\ 1/4 & \text{if } s_0 = 2 \\ 1/4 & \text{if } s_0 = 3. \end{cases}$$

Solution:

(a) Observe that $V^\pi(3) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \cdot 1/2] = 1/2 + \gamma \cdot 1/2 + \gamma^2 \cdot 1/2 + \cdots = 1$ because $s = 3$ is an absorbing state. Further $V^\pi(s) = r(s, a) + \gamma \sum_{s',a} \pi(a|s) P(s'|s, a) V(s')$,

$$V^\pi(2) = \frac{1}{2}\left(\frac{1}{2} \cdot \frac{1}{2} \cdot V^\pi(1) + \frac{1}{2} \cdot \frac{1}{2} \cdot V^\pi(3) + \frac{1}{2} \cdot \frac{1}{2} \cdot V^\pi(1) + \frac{1}{2} \cdot \frac{1}{2} \cdot V^\pi(3)\right) = \frac{V^\pi(1)}{4} + \frac{1}{4}$$

and

$$V^\pi(1) = -1 + \frac{1}{2}\left(\frac{1}{2} \cdot 1 \cdot V^\pi(2) + \frac{1}{2} \cdot 1 \cdot V^\pi(3)\right) = \frac{V^\pi(2)}{4} - \frac{3}{4}.$$

Solving the above two equations results in $V^\pi(1) = -11/15$ and $V^\pi(2) = 1/15$. Hence the solution is $V^\pi = [-11/15, 1/15, 1]$.

(b) Observe that in this instance, the starting state probabilities do not make a difference to the optimal policy. If you are state 3, no matter what action you take, you will continue to be in state 3 and continue to receive a reward of 1/2 and hence for any $x \in [0, 1]$

$$\pi^\star(a|s = 3) = \begin{cases} x & \text{if } a = 1 \\ 1 - x & \text{if } a = 2. \end{cases}$$

Again for state 2, no matter what action you take, you will end up in state 1 or state 3 with probability 1/2 each. Hence, for any $y \in [0, 1]$

$$\pi^\star(a|s = 2) = \begin{cases} y & \text{if } a = 1 \\ 1 - y & \text{if } a = 2. \end{cases}$$

For state 1, action 2 is the fastest path to the positive reward earning state 3. Hence

$$\pi^\star(a|s = 1) = \begin{cases} 0 & \text{if } a = 1 \\ 1 & \text{if } a = 2. \end{cases}$$

WRITE YOUR ANSWER BELOW

WRITE YOUR ANSWER BELOW

WRITE YOUR ANSWER BELOW

Problem 21: *[6 points]* In K-means clustering with the Euclidean distance measure, we seek to minimize the following objective function:

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

where $r_{nk} \in \{0, 1\}$ indicates whether the n-th point $\boldsymbol{x}_n$ is assigned to the k-th cluster whose center is $\boldsymbol{\mu}_k$, for $n = 1, \ldots, N$ and $k = 1, \ldots K$.
The K-means clustering algorithm provides an iterative way to optimize this objective function, but the solution may converge to a local minimum rather than the global minimum. In this problem, we want to find the *optimal* clustering assignment that achieves the *global minimum* of the objective function.

(a) *[2 points]* Suppose we have N points in $\mathbb{R}^d$, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathbb{R}^d$, where d is large, and the number of clusters is $K = 2$. Show that the computational complexity of finding the optimal clustering assignment is $O(Nd2^N)$, and provide an outline of the algorithm that achieves this complexity.

Solution: For $K = 2$, every point could belong to one of two clusters. Therefore, we have to test $O(2^N)$ possible clustering assignments. For each clustering assignment, we have to calculate the objective function and distances from the cluster center for N points which is an additional $O(Nd)$ operation. The overall time complexity is then $O(Nd2^N)$.

(b) *[2 points]* Now suppose the N points are in $\mathbb{R}$ instead, i.e., $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathbb{R}$. Provide a method to find the optimal clustering in $O(N^2)$ complexity. This can be further optimized, but for the sake of this question, only an $O(N^2)$ solution is required.

Solution: We can begin by sorting the points in ascending order $\boldsymbol{x}_1 \leq \boldsymbol{x}_2 \leq \ldots \leq \boldsymbol{x}_N$ which is $O(N \log(N))$ operation. However, we only pay this cost once at the beginning. For this sorted set of points we know that the optimal clustering will be two contiguous set of points with a boundary somewhere in the middle. We test each point as the boundary point so there are N possible boundary points. For each boundary we calculate the objective values which is an $O(N)$ operation. So the overall time complexity will be $O(N \cdot N) = O(N^2)$

(c) *[2 points]* What is the computational complexity of performing T iterations of the K-means clustering algorithm? Compare it with the computational complexities given in parts (a) and (b).

Solution: The complexity of K-means clustering is $O(NKdT)$. For a small T, the complexity is much smaller than the above two computational complexities, however, we are not guaranteed to find the optimal clustering assignment.

WRITE YOUR ANSWER BELOW

WRITE YOUR ANSWER BELOW

Problem 22: *[8 points]* Recall that training a logistic regression model on a dataset $\mathcal{D}$ minimizes the cross-entropy loss

$$\sum_{(\mathbf{x},y)\in\mathcal{D}} \left[y\log\left(\sigma\left(\mathbf{w}^T\mathbf{x}\right)\right) + (1-y)\log\left(1-\sigma\left(\mathbf{w}^T\mathbf{x}\right)\right)\right],$$

where $\sigma(\cdot)$ denotes the sigmoid function. Suppose that you train a logistic regression model on a dataset $\mathcal{D}$ for which each data point n has four-dimensional input features $\mathbf{x}_n \in \mathbb{R}^4$ and a binary label $y_n \in \{0,1\}$. You find that the optimal model parameters are $\mathbf{w} = \begin{bmatrix}1 & 0 & -1 & 0\end{bmatrix}^T$.

(a) *[2 points]* Find the conditions on $a, b, c, d \in \mathbb{R}$ such that $\mathbf{x} = \begin{bmatrix}a & b & c & d\end{bmatrix}^T$ lies on the decision boundary of this learned model.

Solution: The points on the decision boundary satisfy $\mathbf{w}^T\mathbf{x} = 0$, which requires that $a - c = 0$, i.e., $a = c$.

(b) *[2 points]* Suppose you are given a new data point with features $\mathbf{x} = \begin{bmatrix}1 & 2 & 1 & 0\end{bmatrix}^T$ and label $y = 1$. You then add this new data point to your training data. Find the resulting gradient update of the logistic regression model, starting from $\mathbf{w} = \begin{bmatrix}1 & 0 & -1 & 0\end{bmatrix}^T$, evaluated on this new data point.

Solution: The gradient of the cross-entropy loss function with respect to the parameters $\mathbf{w}$ is $\left[\sigma\left(\mathbf{w}^T\mathbf{x}\right) - y\right]\mathbf{x}$. Plugging in the new data point and the current values of $\mathbf{w}$, we obtain $[\sigma(0)-1]\begin{bmatrix}1 & 2 & 1 & 0\end{bmatrix}^T = \begin{bmatrix}\frac{-1}{2} & -1 & \frac{-1}{2} & 0\end{bmatrix}$.

(c) *[2 points]* Suppose that instead of only one new data point as in part (b), you are given two new data points with features $\mathbf{x}_1 = \begin{bmatrix}1 & 2 & 1 & 0\end{bmatrix}^T$ and $\mathbf{x}_2 = \begin{bmatrix}-1 & -2 & -1 & 0\end{bmatrix}^T$, both with the label $y = 1$. Find the resulting gradient update of the logistic regression model with respect to these two new data points.

Solution: We first note that $\sigma(-a) = 1 - \sigma(a)$. Since $\mathbf{x}_1 = -\mathbf{x}_2$, we then see by symmetry that the gradient should be the zero vector.

Quantitatively, calculating the gradient using the same expression as in part (b), we have $[\sigma(0)-1]\begin{bmatrix}1 & 2 & 1 & 0\end{bmatrix}^T - [\sigma(0)-1]\begin{bmatrix}1 & 2 & 1 & 0\end{bmatrix}^T = \begin{bmatrix}0 & 0 & 0 & 0\end{bmatrix}^T$.

(d) *[2 points]* You had originally limited the dataset used for training your logistic regression model, due to resource constraints during training. However, you decide to move the training to a more powerful device, since your mis-classification rate on the test set is currently quite high. You now have the choice to either (i) add a new feature to all data points, or (ii) add ten data points, each of which is correctly classified under the currently learned model and for which $\left|\mathbf{w}^T\mathbf{x}\right| > 1000$, to your training dataset. Is either choice guaranteed to improve your model's accuracy on a test set? Explain your answer in 1-2 sentences.

Solution: Adding the new data points is unlikely to change the training much, since the data points are quite far from the decision boundary and are correctly classified; thus, it is unlikely to improve the test accuracy. Adding the new feature is more likely to improve the model's test accuracy, although this is not guaranteed (it depends on the distribution of the test points).

WRITE YOUR ANSWER BELOW

WRITE YOUR ANSWER BELOW

WRITE YOUR ANSWER BELOW

Problem 23: *[7 points]* We have three "images" $\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \mathbf{X}^{(3)}$ with labels $y^{(1)}$, $y^{(2)}$, $y^{(3)}$, a 2×2 convolutional filter $\mathbf{K}$, and a fixed linear classifier $\mathbf{w} \in \mathbb{R}^4$. Consider a simplified convolutional neural network whose prediction pipeline is as follows:

- **Convolution**: Apply the convolutional filter $\mathbf{K}$ with stride 1 to the input $\mathbf{X}$
- **Flatten**: Flatten the output of the convolution into a column vector $\mathbf{h}$. Note, we flatten row-wise, so the elements of the first row go first, then the elements of the second row, etc.
- **Score**: Compute $s = \mathbf{w}^\top \mathbf{h}$
- **Predict**: Output $\text{sign}(s) \in \{+1, -1\}$. Additionally, we say that $\text{sign}(0) = -1$

The images and associated labels are given as

$$\mathbf{X}^{(1)} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix},\ y^{(1)} = -1, \quad \mathbf{X}^{(2)} = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix},\ y^{(2)} = -1, \quad \mathbf{X}^{(3)} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix},\ y^{(3)} = 1$$

We have initial filter weights, and linear weights:

$$\mathbf{K} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}, \quad \mathbf{w} = \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix}$$

(a) *[2 points]* For which of our 3 images does our convolution neural network with the given initial weights correctly predict the true label?

Solution: For image 1: $\text{conv}(\mathbf{X}^{(1)}, K) = \begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix}$ Flattening and multiplying with our weights we get -2, the sign of which is -1. So the prediction is correct.
For image 2: $\text{conv}(\mathbf{X}^{(2)}, K) = \begin{pmatrix} 2 & 1 \\ 2 & 1 \end{pmatrix}$ Flattening and multiplying with our weights we get 2, the sign of which is 1. So the prediction is incorrect.
For image 3: $\text{conv}(\mathbf{X}^{(3)}, \mathbf{K}) = \begin{pmatrix} 3 & 1 \\ 1 & 1 \end{pmatrix}$ Flattening and multiplying with our weights we get 2, the sign of which is 1. So the prediction is correct. Therefore, the prediction of 2/3 examples is correct

(b) *[2 points]* Suppose we calculate a gradient for the convolutional filter $\mathbf{K}$ which is

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}.$$

We update the convolutional filter using this gradient and learning rate $\eta = 2$. We keep the weights $\mathbf{w}$ fixed. With this change, for which of our 3 images does our convolutional neural network correctly predict the true label?

Solution: The new convolutional filter is

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} - 2 \times \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} -1 & 1 \\ 1 & 0 \end{pmatrix} \tag{2}$$

For image 1: $\text{conv}(\mathbf{X}^{(1)}, \mathbf{K}) = \begin{pmatrix} -1 & 1 \\ 1 & 0 \end{pmatrix}$ Flattening and multiplying with our weights we get -2, the sign of which is -1. So the prediction is correct.

For image 2: $\text{conv}(\mathbf{X}^{(2)}, \mathbf{K}) = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}$ Flattening and multiplying with our weights we get -2, the sign of which is -1. So the prediction is correct.
For image 3: $\text{conv}(\mathbf{X}^{(3)}, \mathbf{K}) = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$ Flattening and multiplying with our weights we get 0, the sign of which is -1. So the prediction is incorrect. Therefore, the prediction of 2/3 examples is correct

(c) *[2 points]* Repeat part (b), but instead using learning rate $\eta = 1$.
Solution: The new convolutional filter is

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} - 1 \times \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{3}$$

For image 1: $\text{conv}(\mathbf{X}^{(1)}, \mathbf{K}) = \begin{pmatrix} 0 & 2 \\ 1 & 1 \end{pmatrix}$ Flattening and multiplying with our weights we get -2, the sign of which is -1. So the prediction is correct.
For image 2: $\text{conv}(\mathbf{X}^{(2)}, \mathbf{K}) = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$ Flattening and multiplying with our weights we get 0, the sign of which is -1. So the prediction is correct.
For image 3: $\text{conv}(\mathbf{X}^{(3)}, \mathbf{K}) = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$ Flattening and multiplying with our weights we get 1, the sign of which is 1. So the prediction is correct. Therefore, the prediction of 3/3 examples is correct

(d) *[1 points]* If part (b) and part (c) result in a different number of correct examples, explain in 1-2 sentences why the choice of learning rate affects the performance. If they are the same, explain why both learning rates result in the same performance.

Solution: Using the large learning rate from part (b) causes you to overshoot the optimal value. Reducing the learning rate helps to classify each example correctly.

WRITE YOUR ANSWER BELOW

WRITE YOUR ANSWER BELOW

WRITE YOUR ANSWER BELOW

Problem 24: *[6 points]* Suppose that you wish to use AdaBoost to train an ensemble of three classifiers for the dataset shown in Figure 1. The red circles and blue squares correspond to the two labels of the data points, and each point is one-dimensional.

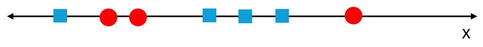


Figure 1: Data points for Problem 24.

(a) *[2 points]* After training the first classifier, you find the decision boundary shown in Figure 2. Points to the left of the boundary (the dashed line) are classified as "red," and points to the right are classified as "blue." Which of the data points will have their weights increased in the second round of AdaBoost?

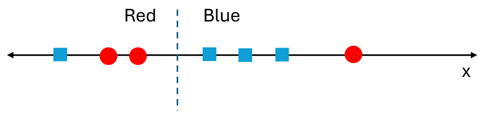


Figure 2: The dashed line shows the decision boundary of the round 1 classifier for Problem 24. Points to the left are classified as "red" and points to the right as "blue."

You can indicate these points by circling them in Figure 2. Briefly explain your answer.

Solution: Mis-classified data points (in this case, the two points on the ends) will have their weights increased.

(b) *[2 points]* You decide to use a decision stump (a single-level decision tree) to train the second classifier. Find the decision stump and explain your answer. You can either describe the decision stump in words or draw its decision boundary in Figure 1.

Hint: You should not need to explicitly calculate the weight of each data point in this round of AdaBoost.

Solution: We consider the eight candidate places for the round 2 classifier. Let "little" denote the smaller weight (i.e., the weight for data points whose weights decreased after round 1) and "big" denote the larger weight (the weight for data points whose weights increased after round 1). Each candidate classifier will have the following weighted errors. Each row corresponds to a possible classifier, moving from left to right, and the left entry denotes the error when labeling the left side blue and right side red, with the right entry denoting the converse.

- 1 big and 3 little or 1 big and 2 little
- 3 little or 2 big and 2 little
- 4 little or 2 big and 1 little
- 5 little or 2 big
- 4 little or 2 big and 1 little
- 3 little or 2 big and 2 little
- 2 little or 2 big and 3 little
- 1 big and 2 little or 1 big and 3 little

We thus see that we should draw a line just to the left of the rightmost red point and label the right side "red" and left side "blue."

(c) *[2 points]* In the third round of AdaBoost, you decide to learn the classifier by training a SVM model to minimize the weighted hinge loss, i.e., you find the parameters $\mathbf{w}, b$ that minimize

$$\sum_{n=1}^{7} r_3(n) \max\left\{0, 1 - y_n \left[\mathbf{w}^T \mathbf{x}_n + b\right]\right\} + \lambda \left\|\mathbf{w}\right\|_2^2,$$

where $r_3(n)$ denotes the weight of the nth data point in round 3 of AdaBoost and $\lambda > 0$ is the weight of the regularization term. Will this training method preserve AdaBoost's iterative minimization of the exponential loss function? Explain why or why not.

Solution: Adaboost requires that we learn a classifier that minimizes the weighted classification error. Minimizing the weighted hinge loss is a proxy for minimizing the weighted classification error, but does not exactly do so. Thus, we do not exactly minimize the exponential loss function.

Answering that the regularization term prevents us from minimizing the classification error was also given full credit.

WRITE YOUR ANSWER BELOW

WRITE YOUR ANSWER BELOW

WRITE YOUR ANSWER BELOW

Problem 25: *[8 points]* A user runs computing jobs using a cloud service that allocates each job to a fast server (whose speed is 10 jobs per unit time) or a slow server (whose speed 1 job per unit time). The service does not reveal its decision (fast versus slow) to the user. The probability of assigning a job to a fast server is α, and the assignment is independent across jobs. Thus, if z denotes the speed at which a job is served,

$$z = \begin{cases} 10 & \text{with prob. } \alpha \\ 1 & \text{with prob. } 1-\alpha \end{cases} \tag{4}$$

The time x taken to complete the job at a given server with speed z is exponentially distributed with rate z. Thus, the probability density function of x is given by:

$$p(x) = z\exp(-xz) \quad \text{for } x \geq 0 \tag{5}$$

Our goal is to estimate α using observed runtimes $x_1, x_2, \ldots x_n$ of n jobs that are executed using the cloud service. Since z is not observed, we can use the EM algorithm to estimate α and in this problem, we will perform one step of the EM algorithm. Let us start with an initial estimate $\alpha^{(0)}$.

(a) *[2 points]* For the i-th job, write down the expression of the complete log-likelihood $\log p(x_i, z_i; \alpha)$, assuming that we are taking natural logarithms (with base e). Observe that this is a function of observed runtime x_i, the hidden variable z_i, and α.

Solution: The complete log-likelihood is:

$$\begin{aligned}\log p(x_i, z_i; \alpha) &= \mathbb{1}(z_i = 10)\log(10\exp(-10x_i) \times \alpha) + \mathbb{1}(z_i = 1)\log(\exp(-x_i) \times (1-\alpha)) \\ &= \mathbb{1}(z_i = 10)\left[\log(10\alpha) - 10x_i\right] + \mathbb{1}(z_i = 1)\left[\log(1-\alpha) - x_i\right]\end{aligned}$$

(b) *[2 points]* Given an observation x_i and current estimate $\alpha^{(0)}$, derive an expression for the posterior probability $\beta_i^{(0)} = p(z = 10|x_i; \alpha^{(0)})$ that this job was executed on a fast server.

Solution:

$$\begin{aligned}\beta_i^{(0)} = p(z = 10|x_i; \alpha^{(0)}) &= \frac{p(x_i, z = 10; \alpha^{(0)})}{p(x_i; \alpha^{(0)})} \\ &= \frac{10\alpha^{(0)}\exp(-10x_i)}{10\alpha^{(0)}\exp(-10x_i) + (1-\alpha^{(0)})\exp(-x_i)}\end{aligned}$$

Similarly, we have

$$\begin{aligned}1 - \beta_i^{(0)} = p(z = 1|x_i; \alpha^{(0)}) &= \frac{p(x_i, z = 1; \alpha^{(0)})}{p(x_i; \alpha^{(0)})} \\ &= \frac{(1-\alpha^{(0)})\exp(-x)}{10\alpha^{(0)}\exp(-10x_i) + (1-\alpha^{(0)})\exp(-x_i)}\end{aligned}$$

(c) *[2 points]* In the E-step of the EM-algorithm, we compute the expected complete log-likelihood function:

$$Q(\alpha|\alpha^{(0)}) = \sum_{i=1}^{n} \mathbb{E}_{z_i|x_i;\alpha^{(0)}} \log p(x_i, z_i; \alpha) \tag{6}$$

where the expectation is taken with respect to the posterior distribution that you derived in part (b). Simplify the expression of $Q(\alpha|\alpha^{(0)})$ and express it in terms of $\beta_i^{(0)}$, x_i and α. Identify which terms depend on α.

Solution:

$$Q(\alpha|\alpha^{(0)}) = \sum_{i=1}^{n} \beta_i^{(0)} \left[\log(10\alpha) - 10x_i\right] + (1-\beta_i^{(0)}) \left[\log(1-\alpha) - x_i\right] \quad (7)$$

$$= \left(\sum_{i=1}^{n} \beta_i^{(0)}\right) \log(10\alpha) + \sum_{i=1}^{n}(1-\beta_i^{(0)}) \log(1-\alpha) + (\text{terms that don't depend on } \alpha) \quad (8)$$

(d) *[2 points]* In the M-step of the EM-algorithm, we maximize $Q(\alpha|\alpha^{(0)})$ with respect to α to get the maximum likelihood estimate $\alpha^{(1)} = \arg\max Q(\alpha|\alpha^{(0)})$. Using your answer to part (c), find an expression for $\alpha^{(1)}$ in terms of n and $\beta_i^{(0)}$ for $i = 1, 2, \ldots, n$.

Solution: Taking the derivative of $Q(\alpha|\alpha^{(0)})$ with respect to α and setting it to zero we have:

$$\frac{\left(\sum_{i=1}^{n} \beta_i^{(0)}\right)}{\alpha} - \frac{\sum_{i=1}^{n}(1-\beta_i^{(0)})}{1-\alpha} = 0 \quad (9)$$

$$\alpha = \frac{\sum_{i=1}^{n} \beta_i^{(0)}}{n} \quad (10)$$

Thus, the estimate of α that we use for the next iteration of EM will be:

$$\alpha^{(1)} = \frac{\sum_{i=1}^{n} \beta_i^{(0)}}{n} \quad (11)$$

WRITE YOUR ANSWER BELOW

WRITE YOUR ANSWER BELOW

WRITE YOUR ANSWER BELOW

Problem 26: *[6 points]* Consider a dataset $\mathcal{D}$ consisting of n data points, each with d-dimensional features and associated labels. Suppose that you run PCA (principal component analysis) on $\mathcal{D}$. You keep the top $k < d$ of the principal components, so that each data point's feature vector $\mathbf{x}$ is now represented by a k-dimensional vector. This choice retains only 90% of the variation in the dataset.
You then decide to run linear regression on the k-dimensional PCA representations. Let $Z \in \mathbb{R}^{n \times k}$ denote the resulting data matrix, each row of which is a k-dimensional PCA representation of one data point's features. You may assume throughout this question that the features of the original data are zero-centered.

(a) *[1 points]* Suppose that you decide to use regularized linear regression, i.e., you wish to find $\mathbf{w} \in \mathbb{R}^k$ that minimizes $\|Z\mathbf{w} - \mathbf{y}\|_2^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$, where $\mathbf{y} \in \mathbb{R}^n$ is a vector of the labels and $\lambda > 0$ is a hyperparameter. Find a closed-form expression for the optimal $\mathbf{w}$ in terms of Z, λ and $\mathbf{y}$.

Solution: The closed-form solution is $\mathbf{w} = \left(Z^T Z + \lambda I\right)^{-1} Z^T \mathbf{y}$.

(b) *[2 points]* Let $Y = Z^T\mathbf{y}$. Express your answer to part (a) in terms of Y, λ and the eigenvalues and eigenvectors of C_X, the covariance matrix of the original data matrix. Remember to show your work.

Solution: Recall that $Z = XP$, where P is the vector of the top k principal components. We can then write $Z^T Z = P^T X^T X P = P^T Q \Lambda Q^T P$, where Λ is a diagonal matrix and $Q\Lambda Q^T$ is the eigenvalue decomposition of $X^T X$. We now know that P simply equals the first k columns of Q, from which it follows that $Z^T Z = P^T X^T X P = P^T Q \Lambda Q^T P = \bar{\Lambda}$, where $\bar{\Lambda}$ is a diagonal matrix with the top k eigenvalues of Λ on the diagonal. We then find that

$$\mathbf{w} = \left(Z^T Z + \lambda I\right)^{-1} Z^T \mathbf{y} = \left(\bar{\Lambda} + \lambda I\right)^{-1} Y.$$

(c) *[2 points]* Does the expression you found in part (a) still give a valid closed-form solution when $\lambda = 0$? Explain why or why not.

Solution: The expression from part (a) is valid when $\lambda = 0$ if $Z^T Z$ is an invertible matrix. To see that it is, we see from part (b) that $Z^T Z = \bar{\Lambda}$. We know that the eigenvalues included in $\bar{\Lambda}$ are all strictly positive as they only capture 90% of the variation in X; thus, $Z^T Z$ is invertible.

(d) *[1 points]* Now suppose that you run regularized linear regression on the original d-dimensional data points, with the same value of λ as in part (a). Will the resulting training loss on $\mathcal{D}$ be the same as the training loss from the model you found in part (a)? Explain why or why not. If not, would we expect a lower or higher loss than that found in part (a)?

Solution: We expect the training loss to be lower with the original representation, since the PCA vectors are just linear combinations of the original data features. Thus, it is impossible for linear regression with the PCA representations to yield a lower training loss than that found with the original representations.

WRITE YOUR ANSWER BELOW

WRITE YOUR ANSWER BELOW

WRITE YOUR ANSWER BELOW