10-315 Spring 2023 Practice Problems

INSTRUCTIONS

- Exam length: 80 minutes
- You are permitted to have one handwritten page of notes, double-sided
- No calculators or other electronic devices allowed

Name	
Andrew ID	

Q1. [0 pts] ML Concepts

In this problem, we will see how you can debug a classifier by looking at its train and test errors. Consider a classifier trained till convergence on some training data $\mathcal{D}^{\text{train}}$, and tested on a separate test set $\mathcal{D}^{\text{test}}$. You look at the test error, and find that it is very high. You then compute the training error and find that it is close to 0.

(a) Which of the following is expected to help? Select all that apply.

- ☒ Increase the training data size.
- ☐ Decrease the training data size.
- ☐ Increase model complexity
- ☒ Decrease model complexity.
- ☐ Train on a combination of $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$ and test on $\mathcal{D}^{\text{test}}$
- ☐ Conclude that Machine Learning does not work.

(b) Explain your choices.

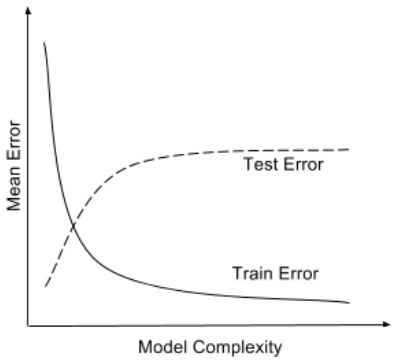
Answer:
The model is overfitting. In order to address the problem, we can either increase training data size or decrease model complexity. We should never train on the test set, the model shouldn't see any testing data in the training or validation process.

(c) What is this scenario called?

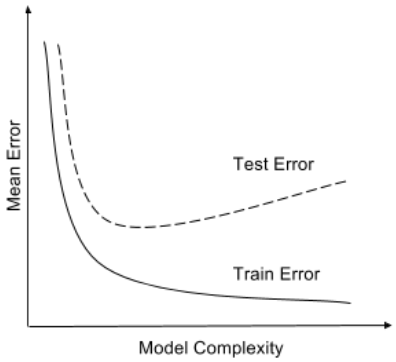
Answer:
Overfitting

(d) Say you plot the train and test errors as a function of the model complexity. Which of the following two plots below is your plot expected to look like?

A.



B.



Answer:
B. When model complexity increases, model can overfit better, so training error will decrease. But when it overfits too much, testing error will increase.

(e) In this question we will consider the effect of increasing the model complexity, while keeping the size of the training set fixed. To be concrete, consider a classification task on the real line $\mathbb{R}$ with distribution D and target function $c^*(x) : \mathbb{R} \to \{\pm 1\}$ and suppose we have a random sample S of size N drawn i.i.d. from D. For each degree d, let ϕ_d be the feature map given by $\phi_d(x) = (1, x, x^2, \ldots, x^d)$ that maps points on the real line to $(d+1)$-dimensional space.

Now consider the learning algorithm that first applies the feature map ϕ_d to all the training examples and then runs logistic regression as in the previous question. A new example is classified by first applying the feature map ϕ_d and then using the learned classifier.

(i) [0 pts] For a given dataset S, is it possible for the training error to increase when we increase the degree d of the feature map? **Please explain your answer in 1 to 2 sentences.**

Answer:
No. Every linear separator using the feature map ϕ_d can also be expressed using the feature map ϕ_{d+1}, since we are only adding new features. It follows that the training error must decrease for any given sample S.

(ii) [0 pts] Briefly **explain in 1 to 2 sentences** why the true error first drops and then increases as we increase the degree d.

Answer:
When the dimension d is small, the true error is high because it is not possible to the target function is not well approximated by any linear separator in the ϕ_d feature space. As we increase d, our ability to approximate c^* improves, so the true error drops. But, as we continue to increase d, we begin to overfit the data and the true error increases again.

(f) What are the effects of the following on overfitting? Choose the best answer.

(i) [0 pts] Increasing decision tree max depth.

- ○ Less likely to overfit
- ● More likely to overfit

Answer:
More likely to overfit

(ii) [0 pts] Increasing decision tree mutual information split threshold.

- ● Less likely to overfit
- ○ More likely to overfit

Answer:
Less likely to overfit

(iii) [0 pts] Increasing decision tree max number of nodes.

- ○ Less likely to overfit
- ● More likely to overfit

Answer:
More likely to overfit

(iv) [0 pts] Increasing k in k-nearest neighbor.

- ● Less likely to overfit
- ○ More likely to overfit

Answer:
Less likely to overfit

(v) [0 pts] Increasing the training data size for decision trees. Assume that training data points are drawn randomly from the true data distribution.

- ● Less likely to overfit
- ○ More likely to overfit

Answer:
Less likely to overfit

(vi) [0 pts] Increasing the training data size for 1-nearest neighbor. Assume that training data points are drawn randomly from the true data distribution.

- ● Less likely to overfit
- ○ More likely to overfit

Answer:
Less likely to overfit

(g) Consider a learning algorithm that uses two hyperparameters, γ and ω, and it takes 1 hour to train *regardless* of the size of the training set.

We choose to do random subsampling cross-validation, where we do K runs of cross-validation and for each run, we randomly subsample a fixed fraction αN of the dataset for validation and use the remaining for training, where $\alpha \in (0, 1)$ and N is the number of data points.

(i) [0 pts] In combination with the cross-validation method above, we choose to do grid search on discrete values for the two hyperparameters.

Given $N = 1000$ data points, $K = 4$ runs, and $\alpha = 0.25$, if we have 100 hours to complete the entire cross-validation process, what is the most number of discrete values of γ that we can include in our search if we also want to include 8 values of ω? Assume that any computations other than training are negligible.

Answer:
$3 + -0.001$. Round $100/4/8 = 3.33$ down to 3.

(ii) [0 pts] In one sentence, give one advantage of increasing the value of α.

Answer:
More data used for validation, giving a better estimate of performance on held-out data.

Q2. [0 pts] Decision Trees

Perceptron Trees: To exploit the desirable properties of decision tree classifiers and perceptrons, Adam came up with a new algorithm called "perceptron trees", which combines features from both. Perceptron trees are similar to decision trees, however each leaf node is a perceptron, instead of a majority vote.

To create a perceptron tree, the first step is to follow a regular decision tree learning algorithm (such as ID3) and perform splitting on attributes until the specified maximum depth is reached. Once maximum depth has been reached, at each leaf node, a perceptron is trained on the remaining attributes which have not been used up in that branch. Classification of a new example is done via a similar procedure. The example is first passed through the decision tree based on its attribute values. When it reaches a leaf node, the final prediction is made by running the corresponding perceptron at that node.

Assume that you have a dataset with 6 binary attributes **(A, B, C, D, E, F)** and two output labels **(-1 and 1)**. A perceptron tree of depth 2 on this dataset is given below. Weights of the perceptron are given in the leaf nodes. Assume bias=1 for each perceptron:

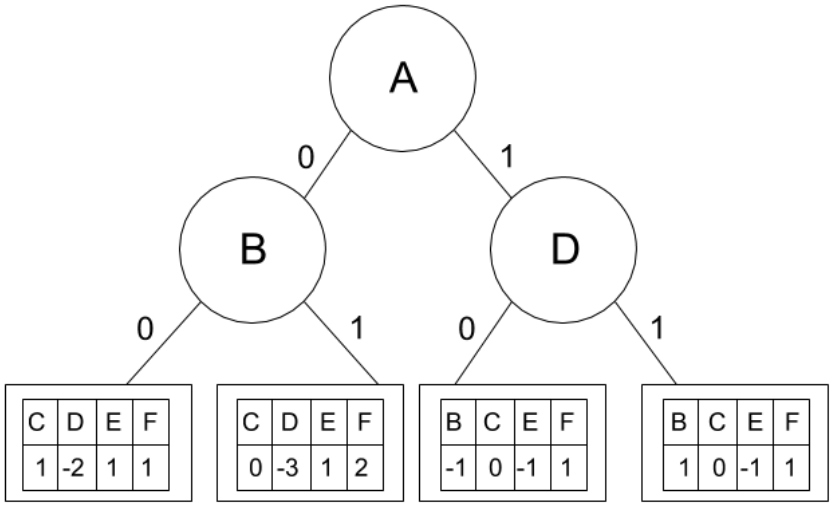


Figure 1: Perceptron Tree of max depth=2

(a) Numerical answer: Given a sample = $[1, 1, 0, 1, 0, 1]$, predict the output label for this sample.

Explanation:
1, Explanation: A=1 and D=1 so the point is sent to the right-most leaf node, where the perceptron output is (1*1)+(0*0)+((-1)*0)+(1*1)+1 = 3. Prediction = sign(3) = 1.

(b) True or False: The decision boundary of a perceptron tree will *always* be linear.

- ● True
- ○ False

Explanation:
False, since decision tree boundaries need not be linear.

(c) [0 pts] **True or False:** For small values of max depth, decision trees are *more* likely to underfit the data than perceptron trees

- ● True
- ○ False

Explanation:
True. For smaller values of max depth, decision trees essentially degenerate into majority-vote classifiers at the leaves. On the other hand, perceptron trees have the capacity to make use of "unused" attributes at the leaves to predict the correct class.

Explanation:
Decision trees: Non-linear decision boundaries
Perceptron: Ability to gracefully handle unseen attribute values in training data/ Better generalization at leaf nodes

(d) Select all that apply: Given a set of input features x, where $x \in \mathbb{R}^n$, you are tasked with predicting a label for y, where $y = 1$ or $y = -1$. You have no knowledge of about the distribution of x and of y. Which of the following methods are appropriate?

- ☐ Perceptron
- ☒ k-Nearest Neighbors
- ☐ Linear Regression
- ☒ Decision Tree with unlimited depth
- ☐ None of the Above

Explanation:
Kth Nearest Neighbours and Decision Tree with unlimited depth since these two methods do not making the assumption of linear separation.

(e) ID3 algorithm is a greedy algorithm for growing Decision Tree and it suffers the same problem as any other greedy algorithm that finds only locally optimal trees. Which of the following method(s) can make ID3 "less greedy"? **Select all that apply:**

- ☐ Use a subset of attributes to grow the decision tree
- ☒ Use different subsets of attributes to grow many decision trees
- ☐ Change the criterion for selecting attributes from information gain (mutual information) to information gain ratio (mutual information divided by entropy of splitting attributes) to avoid selecting attributes with high degree of randomness
- ☒ Keep using mutual information, but select 2 attributes instead of one at each step, and grow two separate subtrees. If there are more than 2 subtrees in total, keep only the top 2 with the best performance (e.g., top 2 with lowest training errors at the current step)

Explanation:
2nd and 4th choices; 1st choice should be wrong as the best performance will be determined by the deepest tree. Any shallower tree will make more mistakes, so ensemble learning can only make performance worse and it won't change the local optimality of the forest.

(f) **(i)** [0 pts] **[0 pts]** ID3 algorithm is guaranteed to find the optimal solution for decision tree.

- ○ True
- ● False

(ii) [0 pts] **[0 pts]** One advantages of decision trees algorithm is that they are not easy to overfit comparing to naive Bayes.

- ○ True
- ● False

Q3. [0 pts] K Nearest Neighbors

(a) True or False: Consider a binary (two classes) classification problem using k-nearest neighbors. We have n 1-dimensional training points $\{x_1, x_2, ..., x_n\}$ with $x_i \in$, and their corresponding labels $\{y_1, y_2, ..., y_n\}$ with $y_i \in \{0, 1\}$.

Assume the data points $x_1, x_2, ..., x_n$ are sorted in the ascending order, we use Euclidean distance as the distance metric, and a point can be it's own neighbor. True or False: We **CAN** build a decision tree (with decisions at each node has the form "$x \geq t$" and "$x < t$", for $t \in$) that behave exactly the same as the 1-nearest neighbor classifier, on this dataset.

- ● True
- ○ False

Explanation:
True, we can build a decision tree by setting the internal nodes at the mid-points between each pair of adjacent training points.

(b) Select all that apply: Please select all that apply about kNN in the following options:

Assume a point can be its own neighbor.

- ☒ k-NN works great with a small amount of data, but struggles when the amount of data becomes large.
- ☐ k-NN is sensitive to outliers; therefore, in general we decrease k to avoid overfitting.
- ☐ k-NN can only be applied to classification problems, but it cannot be used to solve regression problems.
- ☒ We can always achieve zero training error (perfect classification) with k-NN, but it may not generalize well in testing.

Explanation:
True: A, Curse of dimensionality; D, by setting k = 1
False: B, we increase k to avoid overfitting; C, KNN regression

(c) Select one: A k-Nearest Neighbor model with a large value of K is analogous to...

- ● A *short* Decision Tree with a *low* branching factor
- ○ A *short* Decision Tree with a *high* branching factor
- ○ A *long* Decision Tree with a *low* branching factor
- ○ A *long* Decision Tree with a *high* branching factor

Explanation:
A short Decision Tree with a low branching factor

(d) Select one. Imagine you are using a *k*-Nearest Neighbor classifier on a data set with lots of noise. You want your classifier to be *less* sensitive to the noise. Which is more likely to help and with what side-effect?

- ● Increase the value of k => Increase in prediction time
- ○ Decrease the value of k => Increase in prediction time
- ○ Increase the value of k => Decrease in prediction time
- ○ Decrease the value of k => Decrease in prediction time

Explanation:
Increase the value of k => Increase in prediction time

(e) Select all that apply: Identify the correct relationship between bias, variance, and the hyperparameter k in the k-Nearest Neighbors algorithm:

- ☒ Increasing k leads to increase in bias
- ☐ Decreasing k leads to increase in bias
- ☐ Increasing k leads to increase in variance
- ☒ Decreasing k leads to increase in variance

Explanation:
A and D

(f) Consider a training dataset for a regression task as follows.

$$\mathcal{D} = \left\{ \left(x^{(1)}, y^{(1)} \right), \left(x^{(2)}, y^{(2)} \right), \cdots, \left(x^{(N)}, y^{(N)} \right) \right\}$$

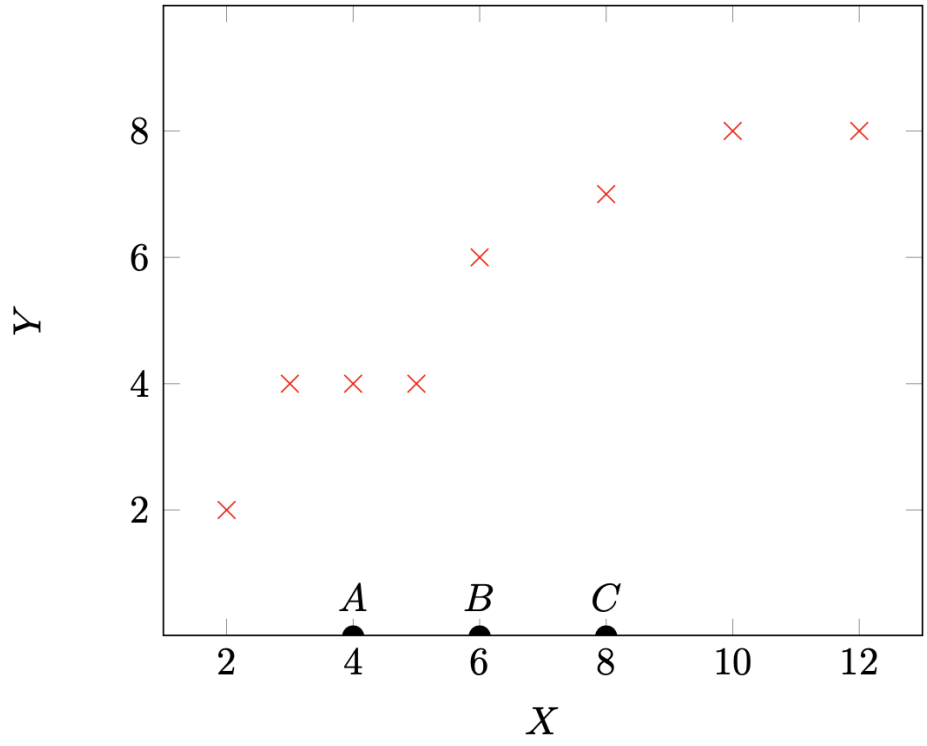
with $x^{(i)} \in \mathbb{R}$ and $y^{(i)} \in \mathbb{R}$.

For regression with k-nearest neighbor, we make predictions on unseen data points similar to the classification algorithm, but instead of a majority vote, we take the mean of the output values of the k nearest points to some new data point x. That is,

$$h(x) = \frac{1}{k} \sum_{i \in \mathcal{N}(x, \mathcal{D})} y^{(i)}$$

where $\mathcal{N}(x, \mathcal{D})$ is the set of indices of the k closest training points to x.

Please answer the following questions using this example dataset:



The red X's denote training points and the black semi-circles A, B, C denote test points of unknown output values. For convenience, all training data points have integer input and output values.

Any ties are broken by selecting the point with the lower x value.

(i) [0 pts] With $k = 1$, what is the mean squared error on the training set?

Explanation:
0 ± 0.00001

(ii) [0 pts] With $k = 2$, what is the predicted value at A?

Explanation:
4 ± 0.00001

(iii) [0 pts] With $k = 2$, what is the predicted value at B?

Explanation:
5 ± 0.00001

(iv) [0 pts] With $k = 3$, what is the predicted value at C?

Explanation:
7 ± 0.00001

(v) [0 pts] With $k = 8$, what is the predicted value at C?

Explanation:
5.375 ± 0.1

(vi) [0 pts] With $k = N$, for any dataset $\mathcal{D}$ with the form specified in the beginning of this question, write down a mathematical expression for the predicted value $\hat{y} = h(x)$. Your response shouldn't include a reference to the neighborhood function $\mathcal{N}()$.

Explanation:
$\bar{y} = \frac{1}{N} \sum_{i=1}^{N} y^{(i)}$

Q4. [0 pts] Linear Regression

(a) Select one: The closed form solution for linear regression is $\theta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$. Suppose you have $N = 35$ training examples and $M = 5$ features (excluding the bias term). Once the bias term is now included, what are the dimensions of $\mathbf{X}$, $\mathbf{y}$, θ in the closed form equation?

- ● $\mathbf{X}$ is 35×6, $\mathbf{y}$ is 35×1, θ is 6×1
- ○ $\mathbf{X}$ is 35×6, $\mathbf{y}$ is 35×6, θ is 6×1
- ○ $\mathbf{X}$ is 35×5, $\mathbf{y}$ is 35×1, θ is 5×1
- ○ $\mathbf{X}$ is 35×5, $\mathbf{y}$ is 35×5, θ is 5×5

(b) Please fill in **True** or **False** for the following questions, providing brief explanations to support your answer.

(i) [0 pts] Consider the linear regression model $y = \mathbf{w}^T\mathbf{x} + \epsilon$. Assuming $\epsilon \sim \mathcal{N}(0, \sigma^2)$ and maximizing the conditional log-likelihood is equivalent to minimizing the sum of squared errors $\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2$. Recall:

$$\mathcal{N}(\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

- ● True
- ○ False

Answer:
True. The squared error term comes from the squared term in the Gaussian distribution.

(ii) [0 pts] Consider N data points, each with one feature x_i and an output y_i. In linear regression, we assume $y_i \sim \mathcal{N}(wx_i, \sigma^2)$ and compute $\hat{w}$ through MLE.
Suppose $y_i \sim \mathcal{N}(\log(wx_i), 1)$ instead. Then the maximum likelihood estimate $\hat{w}$ is the solution to the following equality:

$$\sum_{i=1}^{N} x_i y_i = \sum_{i=1}^{N} x_i \log(wx_i)$$

- ○ True
- ● False

Answer:
False. The likelihood function can be written as

$$\prod_{i=1}^{N} \frac{\exp(-(y_i - \log(wx_i))^2/2)}{\sqrt{2\pi}} = \frac{\exp(-\sum_{i=1}^{N}(y_i - \log(wx_i))^2/2)}{\sqrt{2\pi}}$$

Differentiating wrt w and setting to zero gives us

$$\sum_{i=1}^{N} 2(y_i - \log(wx_i))\frac{x_i}{wx_i} = 0 \implies \sum_{i=1}^{N} y_i = \sum_{i=1}^{N} \log(wx_i)$$

(iii) [0 pts] Consider a linear regression model with only one parameter, the bias, ie., $y = \beta_0$. Then given N data points (x_i, y_i) (where x_i is the feature and y_i is the output), minimizing the sum of squared errors results in β_0 being the median of the y_i values.

- ◯ True
- ● False

Answer:
False. $\sum_{i=1}^{N}(y_i - \beta_0)^2$ is the training cost, which when differentiated and set to zero gives $\beta_0 = \frac{\sum_{i=1}^{N} y_i}{n}$, the mean of the y_i values.

(iv) [0 pts] Given data $\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}, ..., (\mathbf{x}^{(N)}, y^{(N)})\}$, we obtain $\hat{\mathbf{w}}$, the parameters that minimize the training error cost for the linear regression model $y = \mathbf{w}^T\mathbf{x}$ we learn from $\mathcal{D}$.

Consider a new dataset $\mathcal{D}_{\text{new}}$ generated by duplicating the points in $\mathcal{D}$ and adding 10 points that lie along $y = \hat{\mathbf{w}}^T\mathbf{x}$, for a total of $2N + 10$ points. Then the $\hat{\mathbf{w}}_{\text{new}}$ that we learn for $y = \mathbf{w}^T\mathbf{x}$ from $\mathcal{D}_{\text{new}}$ is equal to $\hat{\mathbf{w}}$.

- ● True
- ◯ False

Answer:
True. The new squared error can be written as $2k + m$, where k is the old squared error. $m = 0$ for the 10 points that lie along the line, the lowest possible value for m. And $2k$ is least when k is least, which is when the parameters don't change.

(c) Short answer: Assume we have data $\mathbf{X} \in \mathbb{R}^{n \times d}$ with label $\mathbf{y} \in \mathbb{R}^n$. If the underlying distribution of the data is $\mathbf{y} = \mathbf{X}\beta^* + \epsilon$, where $\epsilon \sim N(0, \mathbf{I})$. Assume the closed form solution $\hat{\beta}$ for mean squared error linear regression exists for this data, write out $\hat{\beta}$'s distribution:

Answer:
$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{X}\beta^* + \epsilon) = \beta^* + (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\epsilon \sim N(\beta^*, (\mathbf{X}^T\mathbf{X})^{-1})$

(d) Consider linear regression on 1-dimensional data $\mathbf{x} \in \mathbb{R}^n$ with label $\mathbf{y} \in \mathbb{R}^n$. We apply linear regression in both directions on this data, i.e., we first fit y with x and get $y = \beta_1 x$ as the fitted line, then we fit x with y and get $x = \beta_2 y$ as the fitted line. Discuss the relations between β_1 and β_2:

(i) [0 pts] **True or False:** The two fitted lines are always the same, i.e. we always have $\beta_2 = \frac{1}{\beta_1}$.

- ◯ True
- ● False

Answer:
False.

(ii) [0 pts] **Numerical answer:** We further assume that $\mathbf{x}^T\mathbf{y} > 0$. What is the minimum value of $\frac{1}{\beta_1} + \frac{1}{\beta_2}$?

Answer:
2.

(e) Please circle **True** or **False** for the following questions, providing brief explanations to support your answer.

(i) [0 pts] **[0 pts]** Consider a linear regression model with only one parameter, the bias, ie., $y = \beta_0$. Then given n data points (x_i, y_i) (where x_i is the feature and y_i is the output), minimizing the sum of squared errors results in β_0 being the median of the y_i values.

Circle one: **True** **False**
Brief explanation:

Answer:
False. $\sum_{i=1}^{n}(y_i - \beta_0)^2$ is the training cost, which when differentiated and set to zero gives $\beta_0 = \frac{\sum_{i=1}^{n} y_i}{n}$, the mean of the y_i values.

(ii) [0 pts] **[3 pts]** Given data $D = \{(x_1, y_1), ..., (x_n, y_n)\}$, we obtain $\hat{w}$, the parameters that minimize the training error cost for the linear regression model $y = w^T\mathbf{x}$ we learn from D.

Consider a new dataset D_{new} generated by duplicating the points in D and adding 10 points that lie along $y = \hat{w}^T\mathbf{x}$. Then the $\hat{w}_{\text{new}}$ that we learn for $y = w^T\mathbf{x}$ from D_{new} is equal to $\hat{w}$.

Circle one: **True** **False**
Brief explanation:

Answer:
True. The new squared error can be written as $2k + m$, where k is the old squared error. $m = 0$ for the 10 points that lie along the line, the lowest possible value for m. And $2k$ is least when k is least, which is when the parameters don't change.

(f) Given a set of training pairs $\{(x_i, y_i), i = 1, \ldots, n\}$ where $x_i \in \mathbb{R}^d$ is the input and $y_i \in \mathbb{R}$ is the output, we want to find a linear function $f(x) = w^T x$ that minimizes a tradeoff between training error and model complexity. The ridge regression formulation captures this tradeoff:

$$\begin{aligned}\hat{w} &= \operatorname*{argmin}_{w \in \mathbb{R}^d} J(w) \\ &= \operatorname*{argmin}_{w \in \mathbb{R}^d} \frac{1}{2}\sum_{i=1}^{n}(w^T x_i - y_i)^2 + \frac{\lambda}{2}\sum_{i=1}^{d} w_i^2,\end{aligned}$$

where $\lambda \geq 0$. It is possible to derive a closed form expression for the parameter vector $\hat{w}$ that minimizes this

cost function. The gradient, using matrix notation, is

$$\nabla J(w) = \begin{bmatrix} \frac{\partial \ell(w)}{\partial w_1} \\ \vdots \\ \frac{\partial \ell(w)}{\partial w_d} \end{bmatrix} = X^T X w - X^T y + \lambda w,$$

where $X \in \mathbb{R}^{n \times d}$ is the matrix with the the training input instances on the rows (x_i on row i), and $y \in \mathbb{R}^n$ is the vector of training output instances.

(i) [0 pts] [0 pts.] What is the closed form expression for $\hat{w}$ that we obtain by solving $\nabla \ell(w) = 0$? Hint: use $\lambda w = I w \lambda$, where I is the identity matrix.

Answer:
$\nabla J(w) = 0 \implies \hat{w} = (X^T X + \lambda I)^{-1} X^T y.$

(ii) [0 pts] [0 pts.] What is the meaning of the λ parameter? What kind of tradeoff between training error and model complexity we have when λ approaches zero? What about when λ goes to infinity?

Answer:
When $\lambda \to 0$, we prefer models that achieve the smallest possible training error, irrespective their complexity (squared ℓ_2 norm). When $\lambda \to +\infty$, we prefer models that are the simplest possible (i.e. that have small norm). We expect $\hat{w}$ to approach the zero vector in this case.

(iii) [0 pts] [0 pts.] Answer true or false to each of the following questions and provide a brief justification for your answer:

- [0 pt.] **T or F:** In ridge regression, when solving for the linear regressor that minimizes the training cost function, it is always preferable to use the closed form expression rather than using an iterative method like gradient descent, even when the number of parameters is very high.

  Answer:
  False. Using the closed form expression to solve for $\hat{w}$ requires $O(d^3)$ operations (the cost of solving the linear system or computing the inverse), while using a gradient descent approach requires $O(d^2)$ (the cost of matrix multiplication) per step. If the number of parameters is very high, it may not be computationally feasible to use the closed form expression. Also, if the required precision for the solution $\hat{w}$ is moderate, as it is typically the case in machine learning applications, gradient descent may be preferable.

- [0 pt.] **T or F:** Using a non-linear feature map is never useful as all regression problems have linear input to output relations.

  Answer:
  False. We very rarely believe that the true regression function is linear in the initial input space. Many times, the choice of a linear model is done out of mathematical convenience.

- [0 pt.] **T or F:** In linear regression, minimizing a tradeoff between training error and model complexity, usually allows us to obtain a model with lower test error than just minimizing training error.

  Answer:
  True. Not using regularization leads us to choose more complex models, which have higher variance. In practice it is typically better to optimize a tradeoff between data fitting and model complexity. This is especially true when the ratio between the size of the training set and the number of parameters is not very big.

- [0 pt.] **T or F:** When $\lambda = 0$, we recover ordinary least squares (OLS). If $n < d$, then $X^T X$ does not have an inverse and the estimator $\hat{w}$ is not well-defined.

  Answer:
  True. This is a problem with the ordinary least squares estimator for the case where the number of parameters is bigger than the number of data points.

(g) **(i)** [0 pts] Which of the following are valid expressions for the mean squared error objective function for linear regression with dataset $\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N}$, with each $\mathbf{x}^{(i)} \in \mathbb{R}^M$ and the design matrix $\mathbf{X} \in \mathbb{R}^{N \times (M+1)}$. $\mathbf{y}$ and $\boldsymbol{\theta}$ are column vectors.

Select all that apply:

$J(\boldsymbol{\theta}) = \frac{1}{N}\|\mathbf{y} - \boldsymbol{\theta}\mathbf{X}\|_2^2$ $J(\boldsymbol{\theta}) = \frac{1}{N}\|\mathbf{y}^T - \boldsymbol{\theta}\mathbf{X}\|_2^2$ $J(\boldsymbol{\theta}) = \frac{1}{N}\|\mathbf{y}^T - \mathbf{X}\boldsymbol{\theta}\|_2^2$ $J(\boldsymbol{\theta}) = \frac{1}{N}\|\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\|_2^2$ None of the Above

Answer:
$J(\boldsymbol{\theta}) = \frac{1}{N}\|\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\|_2^2$

(ii) [0 pts] Your friend accidentally solved linear regression using the wrong objective function for mean squared error, specifically, they used the following objective function that contains two mistakes: 1) they forgot the $1/N$ and 2) they have one sign error.

$$J(\mathbf{w}, b) = \sum_{i=1}^{N} \left(y^{(i)} - \left(\sum_{j=1}^{M} w_j x_j^{(i)} - b \right) \right)^2$$

You realize that you can still use the parameters that they learned, $\mathbf{w}$ and b, to correctly predict y given $\mathbf{x}$.

Write the equation that implements this corrected prediction function $h(\mathbf{x}, \mathbf{w}, b)$ using your friend's learned parameters, $\mathbf{w}$ and b. The $\mathbf{w}$ and $\mathbf{x}$ vectors are column vectors.

Answer:
$h(\mathbf{x}, \mathbf{w}, b) = \mathbf{w}^T\mathbf{x} - b$

(iii) [0 pts] We have 2 data points:

$$\mathbf{x}^{(1)} = [2, 1]^T \quad y^{(1)} = 7$$

$$\mathbf{x}^{(2)} = [1, 2]^T \quad y^{(2)} = 5$$

We know that for linear regression with a bias/intercept term and mean squared error, there are infinite solutions with these two points.

Give a specific third point $\mathbf{x}^{(3)}$, $y^{(3)}$, such that, when included with the first two, will cause linear regression to still have infinite solutions. Your $\mathbf{x}^{(3)}$ should not equal $\mathbf{x}^{(1)}$ or $\mathbf{x}^{(2)}$ and your $y^{(3)}$ should not equal $y^{(1)}$ or $y^{(2)}$.

$x_1^{(3)}$ **Answer:**

$x_2^{(3)}$ **Answer:**

$y^{(3)}$ **Answer:**

Answer:
Any $\mathbf{x}^{(3)}$ that is colinear with the first two $\mathbf{x}$'s; y doesn't matter.

After adding your third point, if we then double the output of just the first point such that now $y^{(1)} = 14$, will this change the number of solutions for linear regression?

Yes No

Answer:
No

(h) Given that we have an input x and we want to estimate an output y, in linear regression we assume the relationship between them is of the form $y = wx + b + \epsilon$, where w and b are real-valued parameters we estimate and ϵ represents the noise in the data. When the noise is Gaussian, maximizing the likelihood of a dataset $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ to estimate the parameters w and b is equivalent to minimizing the squared error:

$$\underset{w}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - (wx_i + b))^2.$$

Consider the dataset S plotted in Fig. 2 along with its associated regression line. For each of the altered data sets S^{new} plotted in Fig. 4, indicate which regression line (relative to the original one) in Fig. 3 corresponds to the regression line for the new data set. Write your answers in the table below.

Dataset	(a)	(b)	(c)	(d)	(e)
Regression line					

Answer:

Dataset	(a)	(b)	(c)	(d)	(e)
Regression line	(b)	(c)	(b)	(a)	(a)

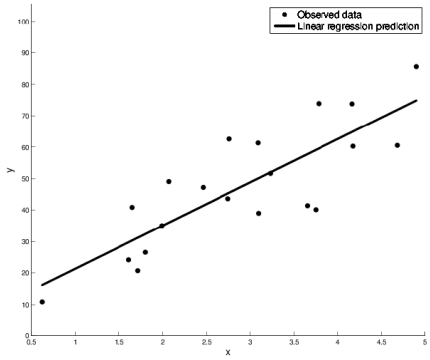
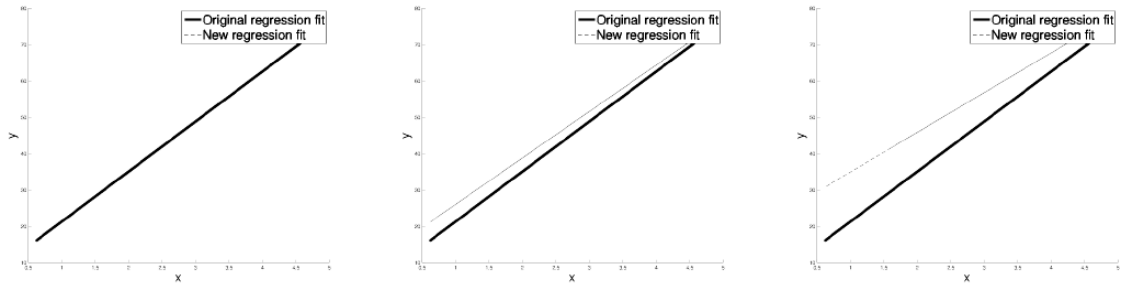


Figure 2: An observed data set and its associated regression line.



(a) Old and new regression lines. (b) Old and new regression lines. (c) Old and new regression lines.

Figure 3: New regression lines for altered data sets S^{new}.

Q5. [0 pts] Logistic Regression

Given a training set $\{(\mathbf{x}^{(i)}, y^{(i)}), i = 1, ..., N\}$ where $\mathbf{x}^{(i)} \in \mathbb{R}^D$ is a feature vector and $y^{(i)} \in \{0, 1\}$ is a binary label, we want to find the parameters $\hat{\mathbf{w}}$ that maximize the likelihood for the training set, assuming a model of the form:

$$p(Y = 1 \mid \mathbf{x}, \mathbf{w}) = \frac{1}{1 + exp(-\mathbf{w}^T \mathbf{x})}$$

The conditional log likelihood of the training set is:

$$\ell(w) = \sum_{i=1}^{N} y^{(i)} \log(p(y^{(i)} \mid \mathbf{x}^{(i)}, \mathbf{w})) + (1 - y^{(i)}) \log(1 - p(y^{(i)} \mid \mathbf{x}^{(i)}, \mathbf{w}))$$

and the gradient is:

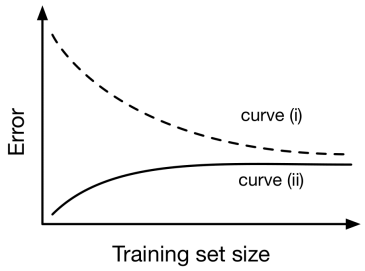
$$\nabla \ell(\mathbf{w}) = \sum_{i-1}^{n} (y^{(i)} - p(y^{(i)} \mid \mathbf{x}^{(i)}, \mathbf{w}))\mathbf{x}^{(i)}$$

(a) Is it possible to get a closed form for the parameters $\hat{\mathbf{w}}$ that maximize the conditional log likelihood? How would you compute $\hat{\mathbf{w}}$ in practice?

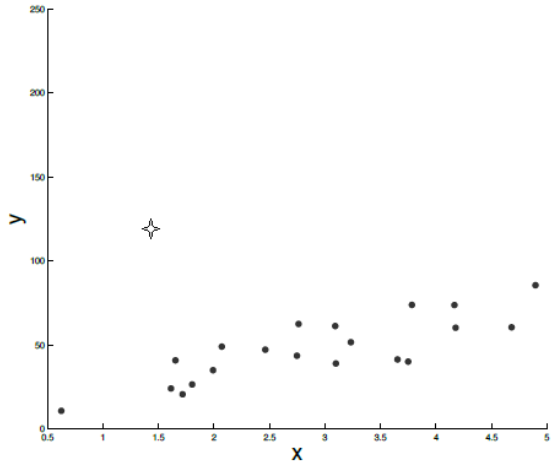
Answer:
There is no closed form expression for maximizing the conditional log likelihood. One has to consider iterative optimization methods, such as gradient descent, to compute $\hat{\mathbf{w}}$.

(b) In this problem, we will consider the effect of training sample size N on a logistic regression classifier with M features. The classifier is trained by optimizing the conditional log-likelihood. The optimization procedure stops if the estimated parameters perfectly classify the training data or they converge.

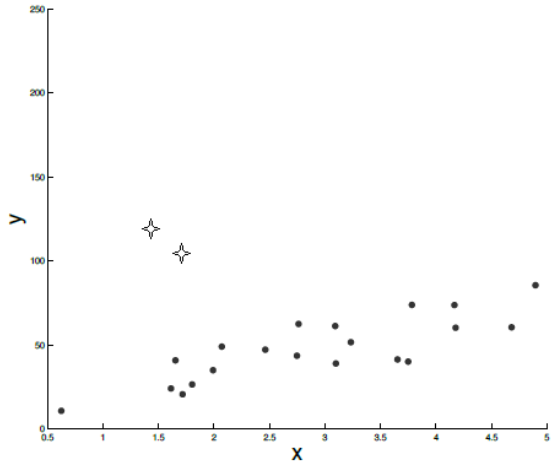
The following plot shows the general trend for how the training and testing error change as we increase the sample size $N = |S|$. Your task in this question is to analyze this plot and identify which curve corresponds to the training and test error. Specifically:



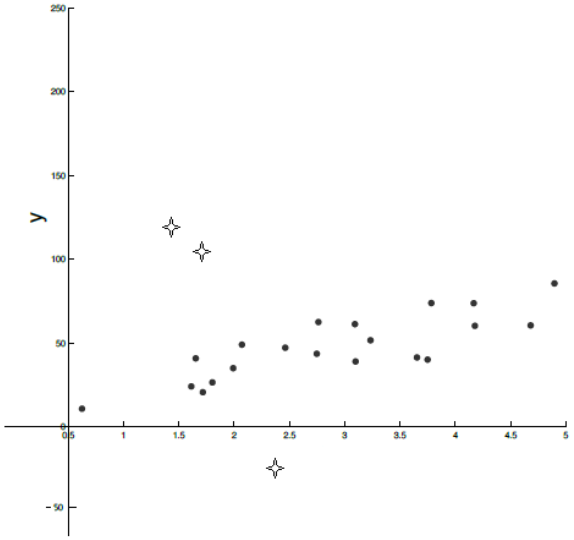
(i) [0 pts] Which curve represents the training error? **Please provide 1–2 sentences of justification**.



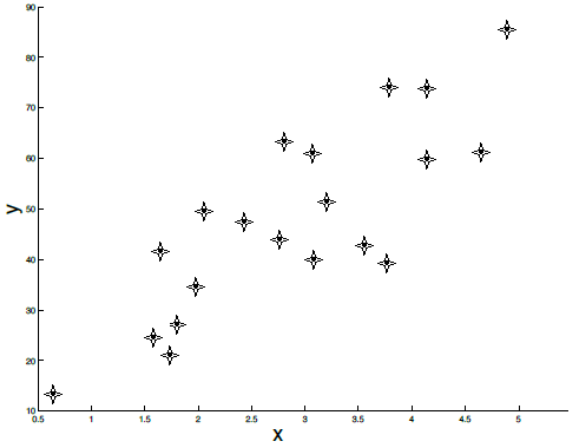
(a) Adding one outlier to the original data set.



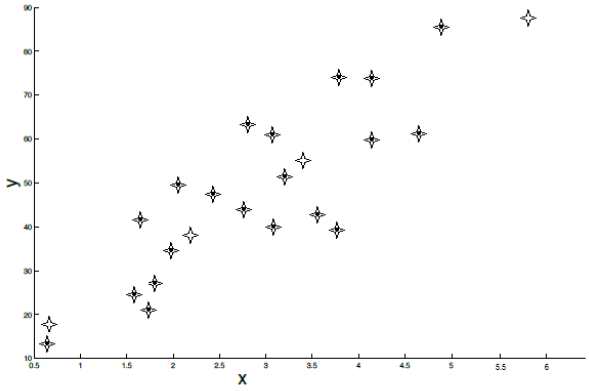
(b) Adding two outliers to the original data set.



(c) Adding three outliers to the original data set. Two on one side and one on the other side.



(d) Duplicating the original data set.



(e) Duplicating the original data set and adding four points that lie on the trajectory of the original regression line.

Figure 4: New data set S^{new}.

Answer:
Curve (ii) is the training set. Training error increases as the training set increases in size (more points to account for). However, the increase tapers out when the model generalizes well. Evidently, curve (i) is testing, since larger training sets better form generalized models, which reduces testing error.

(ii) [0 pts] In one word, what does the gap between the two curves represent?

Answer:
Overfitting

(c) We have a set of 2-dimensional data points $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(N)}\}$ with $\mathbf{x}^{(i)} \in \mathbb{R}^2$, and their corresponding labels $\{y^{(1)}, y^{(2)}, ..., y^{N)}\}$ with $y^{(i)} \in \{0, 1\}$ shown below in figure 5. Let "$\circ$" denote 1 and "$\times$" denote 0.

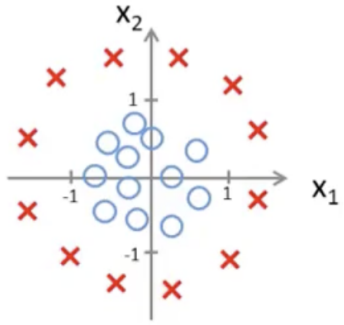


Figure 5: Non-linear classification boundary

(i) [0 pts] **True or False:** We can engineer a set of features such that a binary logistic regression model trained on the transformed features perfectly categorizes this dataset.

- ● True
- ○ False

(ii) [0 pts] If true, please specify the feature map $\phi(x_1, x_2)$ such that binary logistic regression can perfectly categorize the above dataset. If false, please explain why.

Answer:
For example, $\phi(x_1, x_2) = [1, x_1^2, x_2^2]^T$. A model such as $y = g(\theta_0 + \theta_1 x_1^2 + \theta_2 x_2^2), \theta_i \in \mathbb{R}$, can perfectly categorize such model.

Q6. [0 pts] Regularization

(a) Select all that apply: Recall in regularization, given an objective function $J(\Theta)$, we have

$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmin}}\, J(\Theta) + \lambda r(\Theta)$$

where $r(\Theta)$ is the regularization term. Which of the following are true.

- ☐ The recommended way to choose λ is to pick the value of λ that minimizes the **training set error**.
- ☒ The recommended way to choose λ is to pick the value of λ that minimizes the **validation error**.
- ☐ The recommended way to choose λ is to pick the value of λ that minimizes the **test set error**.

(b) Select all that apply: Which of the following is true about the regularization parameter λ.

- ☐ Larger values of λ can overfit the data.
- ☐ Larger λ does not affect the performance of your hypothesis
- ☒ Adding a regularization term to a classifier, ($\lambda > 0$), may cause more training examples to be classified incorrectly.

(c) In logistic regression, adding a regularization term typically results in a decrease in the training error.

- ○ True
- ● False

Explain:
False. Regularization prevents overfitting — overfitting leads to trivially low training error.

(d) In the unregularized binary logistic regression, the log likelihood may be written as:

$$\ell(\mathbf{w}) = \sum_{i=1}^{N} \left(y^{(i)} \mathbf{w}^T \mathbf{x}^{(i)} - \log\left(1 + \exp(\mathbf{w}^T \mathbf{x}^{(i)})\right)\right)$$

In this question, the goal is to find the following parameter estimates by optimizing the objective function with an extra regularization term:

$$\hat{\mathbf{w}} = \arg\max_{\mathbf{w}} f(\mathbf{w}),$$

$$f(\mathbf{w}) = \ell(\mathbf{w}) - \frac{1}{2} \sum_{m=1}^{M} q_m (w_m)^2$$

where, instead of a single hyperparameter, λ, we have a hyperparameter, q_m, on each weight.

Write the partial derivative of the objective function, f with respect to its m-th feature, $\frac{\partial f}{\partial w_m}$. You do not need to simplify the final form with linear algebra.

Answer:

$$\begin{aligned}
\frac{\partial f}{\partial w_m} &= \frac{\partial}{\partial w_m} \left(\sum_{i=1}^{N} \left(y^{(i)} \mathbf{w}^T \mathbf{x}^{(i)} - \log\left(1 + \exp(\mathbf{w}^T \mathbf{x}^{(i)})\right)\right) - \frac{1}{2} \sum_{m=1}^{M} q_m (w_m)^2 \right) \\
&= \sum_{i=1}^{n} \left(y^{(i)} x_m^{(i)} - \frac{\exp(\mathbf{w}^T \mathbf{x}^{(i)})}{1 + \exp(\mathbf{w}^T \mathbf{x}^{(i)})} x_m^{(i)} \right) - \frac{1}{2} q_m * 2 w_m \\
&= \sum_{i=1}^{n} \left(y^{(i)} x_m^{(i)} - \frac{\exp(\mathbf{w}^T \mathbf{x}^{(i)})}{1 + \exp(\mathbf{w}^T \mathbf{x}^{(i)})} x_m^{(i)} \right) - q_m w_m
\end{aligned}$$

Q7. [0 pts] Optimization

(a) Select all that apply: Which of the following are correct regarding Gradient Descent (GD) and stochastic gradient descent (SGD)

- ☐ Each update step in SGD pushes the parameter vector closer to the parameter vector that minimizes the objective function.
- ☒ The gradient computed in SGD is, in expectation, equal to the gradient computed in GD.
- ☐ The gradient computed in GD has a higher variance than that computed in SGD, which is why in practice SGD converges faster in time than GD.

Explanation:
B is the correct answer. A is incorrect, SGD updates are high in variance and may not go in the direction of the true gradient. C is incorrect, for the same reason.

(b) **(i)** [0 pts] Determine if the following 1-D functions are convex. Assume that the domain of each function is $\mathbb{R}$. The definition of a convex function is as follows:

$$f(x) \text{ is convex} \iff f(\alpha x + (1-\alpha)z) \leq \alpha f(x) + (1-\alpha)f(z), \forall \alpha \in [0,1] \text{ and } \forall x, z.$$

Select all convex functions:

- ☒ $f(x) = x + b$ for any $b \in \mathbb{R}$
- ☒ $f(x) = c^2 x$ for any $c \in \mathbb{R}$
- ☐ $f(x) = ax^2 + b$ for any $a \in \mathbb{R}$ and any $b \in \mathbb{R}$
- ☒ $f(x) = 0$
- ☐ None of the Above

Explanation:
$f(x) = x + b$ for any $b \in \mathbb{R}$, $f(x) = c^2 x$ for any $c \in \mathbb{R}$, $f(x) = 0$.

(ii) [0 pts] Consider the convex function $f(z) = z^2$. Let α be our learning rate in gradient descent. For which values of α will $\lim_{t\to\infty} f(z^{(t)}) = 0$, assuming the initial value of z is $z^{(0)} = 1$ and $z^{(t)}$ is the value of z after the t-th iteration of gradient descent.

Select all that apply:

- ☐ $\alpha = 0$
- ☒ $\alpha = \frac{1}{2}$
- ☐ $\alpha = 1$
- ☐ $\alpha = 2$
- ☐ None of the Above

Explanation:
$\alpha = \frac{1}{2}$

(iii) [0 pts] Give the range of all values for $\alpha \geq 0$ such that $\lim_{t\to\infty} f(z^{(t)}) = 0$, assuming the initial value of z is $z^{(0)} = 1$. Be specific.

Explanation:
$(0, 1)$.

Q8. [0 pts] MLE

(a) Let $\mathcal{D} = X_1, X_2, \ldots, X_N$ be a set of i.i.d. random variables with a Poisson probability distribution.

$$p(X = x \mid \lambda) = \frac{\lambda^x e^{-\lambda}}{x!}$$

Derive the maximum likelihood estimate for $\mathcal{D}$.

Answer:

$$L(\lambda) = \prod_{i=1}^{N} p\left(x_i|\lambda\right) = \prod_{i=1}^{N} \lambda^{x_i} e^{-\lambda} \left(x_i!\right)^{-1}$$

Take log of both sides. RHS will be

$$\sum_{i=1}^{N} \log \left[\lambda^{x_i} e^{-\lambda} \left(x_i!\right)^{-1}\right]$$

$$\sum_{i=1}^{N} x_i \log \lambda - \lambda - \log\left(x_i!\right)$$

Differentiate wrt λ to get:

$$\sum_{i=1}^{N} \left(\frac{x_i}{\lambda} - 1\right)$$

Simplify further to get:

$$\lambda = \frac{\sum_{i=1}^{N} x_i}{N}$$

Q9. [0 pts] Neural Networks

(a) Select all that apply: The XOR function for two binary variables x and y is defined as follows:

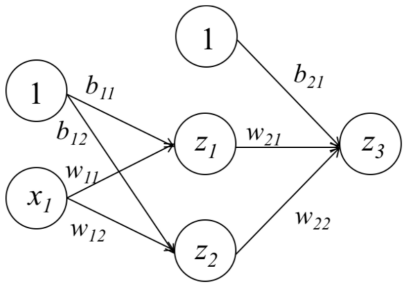
$$f(x,y) = 0, x = y$$
$$= 1, x \neq y$$

Based on this information, select all statements which are true.

- ☐ The XOR function is linearly separable.
- ☒ A single-layer perceptron is incapable of representing the XOR function.
- ☐ A multi-layer perceptron is incapable of representing the XOR function.
- ☐ None of the above.

Answer:
B is the only correct option. The XOR function is not linearly separable, hence a single-layer perceptron cannot represent it but a multi-layer perceptron can.

(b) Consider the neural network architecture shown below for a binary classification problem.



We define:

$$a_1 = w_{11}x_1 + b_{11}$$
$$a_2 = w_{12}x_1 + b_{12}$$
$$a_3 = w_{21}z_1 + w_{22}z_2 + b_{21}$$
$$z_1 = ReLU(a_1)$$
$$z_2 = ReLU(a_2)$$
$$z_3 = g(a_3)$$
$$g(x) = \frac{1}{1+e^{-x}}$$

(i) [0 pts] Write the partial derivative of loss function $L(y, z_3)$ with respect to the bias term b_{21}, $\frac{\partial L}{\partial b_{21}}$, in terms of the partial derivatives $\frac{\partial \alpha}{\partial \beta}$, where α and β can be any of $L, z_i, a_i, b_{ij}, w_{ij}, x_1$ for all valid values of i,j. Your answer should be as explicit as possible – that is, making sure each partial derivative $\frac{\partial \alpha}{\partial \beta}$ cannot be decomposed further into simpler partial derivatives. Do NOT evaluate the partial derivatives.

Answer:
$\frac{\partial L}{\partial b_{21}} = \frac{\partial L}{\partial z_3}\frac{\partial z_3}{\partial a_3}\frac{\partial a_3}{\partial b_{21}}$

(ii) [0 pts] Using the same neural network, write the partial derivative of the loss function $L(y, z_3)$ with respect to the bias term b_{12}, $\frac{\partial L}{\partial b_{12}}$, in terms of the partial derivatives $\frac{\partial \alpha}{\partial \beta}$, where α and β can be any of $L, z_i, a_i, b_{ij}, w_{ij}, x_1$ for all valid values of i,j. Your answer should be as explicit as possible—that is, make sure each partial derivative $\frac{\partial \alpha}{\partial \beta}$ cannot be decomposed further into simpler partial derivatives. Do NOT evaluate the partial derivatives.

Answer:

$$\frac{\partial L}{\partial b_{12}} = \frac{\partial L}{\partial z_3}\frac{\partial z_3}{\partial a_3}\frac{\partial a_3}{\partial z_2}\frac{\partial z_2}{\partial a_2}\frac{\partial a_2}{\partial b_{12}}$$

THIS PAGE INTENTIONALLY LEFT BLANK