Solutions

10-601 Machine Learning
Summer 2025
Practice Problems
Updated: May 22, 2025
Time Limit: N/A

Name:
Andrew ID:
Room:
Seat:
Exam Number:

Instructions:

- Verify your name and Andrew ID above.
- This exam contains 52 pages (including this cover page).
  The total number of points is 0.
- Clearly mark your answers in the allocated space. If you have made a mistake, cross out the invalid parts of your solution, and circle the ones which should be graded.
- Look over the exam first to make sure that none of the 52 pages are missing.
- No electronic devices may be used during the exam.
- Please write all answers in pen or *darkly* in pencil.
- You have N/A to complete the exam. Good luck!

Question	Points
1. Decision Trees	0
2. K Nearest Neighbors	0
3. Model Selection and Errors	0
4. Perceptron	0
5. MLE/MAP	0
6. Logistic Regression and Regularization	0
7. Linear Regression	0
8. Optimization	0
9. Feature Engineering and Regularization	0
10. Neural Networks	0
Total:	0

Instructions for Specific Problem Types

For "Select One" questions, please fill in the appropriate bubble completely:

Select One: Who taught this course?

● Matt Gormley

○ Marie Curie

○ Noam Chomsky

If you need to change your answer, you may cross out the previous answer and bubble in the new answer:

Select One: Who taught this course?

● Henry Chai

○ Marie Curie

~~●~~ Noam Chomsky

For "Select all that apply" questions, please fill in all appropriate squares completely:

Select all that apply: Which are instructors for this course?

■ Matt Gormley

■ Henry Chai

□ I don't know

Again, if you need to change your answer, you may cross out the previous answer(s) and bubble in the new answer(s):

Select all that apply: Which are the instructors for this course?

■ Matt Gormley

■ Henry Chai

~~■~~ I don't know

For questions where you must fill in a blank, please make sure your final answer is fully included in the given space. You may cross out answers or parts of answers, but the final answer must still be within the given space.

Fill in the blank: What is the course number?

$\boxed{\text{10-601}}$ $\boxed{\text{10-}\cancel{6}\text{301}}$

1 Decision Trees (0 points)

1.1. To exploit the desirable properties of decision tree classifiers and perceptrons, Adam came up with a new algorithm called the "perceptron tree" that combines features from both. Perceptron trees are similar to decision trees, but each leaf node contains a perceptron rather than 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 yet been used 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, 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 $b = 1$ for each perceptron.

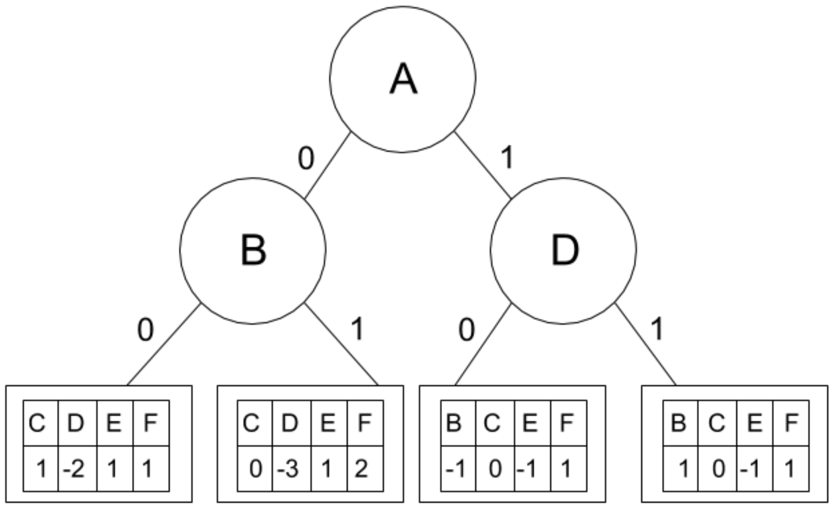


Figure 1: Perceptron Tree of depth 2

(a) **Numerical answer:** What would the given perceptron tree predict as the output label for the sample $\mathbf{x} = [1, 1, 0, 1, 0, 1]$?

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

False, since decision tree boundaries need not be linear.

(c) **True or False:** For small values of max depth (e.g., 2 or 3), decision trees are *more* likely to underfit the data than perceptron trees.

◯ True

◯ False

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.

1.2. **Select all that apply:** Given a training dataset, $\mathcal{D}$, suppose you train two decision trees, one using mutual information as the splitting criterion and the other using training error rate. If both trees are trained until they achieve zero training error, which of the following statements must be true?

☐ Both trees split on the same feature at the root node.

☐ Both trees have the same depth

☐ Both trees make the same predictions for each data point in $\mathcal{D}$

☐ Both trees use all of the features in $\mathcal{D}$

☐ None of the above.

C; the two splitting criterion could make completely different splits and use completely different features but because we know both achieve zero training error, they must make the same predictions (i.e., the correct labels) for each training data point.

1.3. **True or False:** One advantage of the ID3 algorithm for training decision trees is that it is not susceptible to overfitting because it uses recursion.

◯ True

◯ False

False

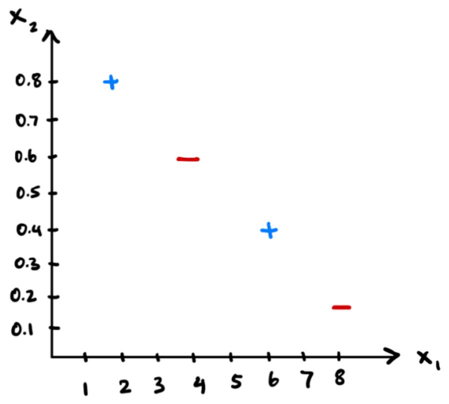
1.4. **Fill in the blank:** Complete the following paragraph about pruning decision trees by circling the best of the provided options for each of blanks:

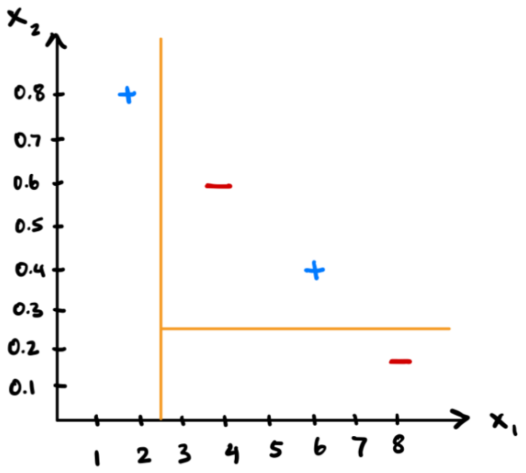
When pruning a decision tree, each split is replaced with the most common label among the __training / validation__ data points that end up at each node. Splits are compared based on their reduction in __training / validation__ error rate and if two splits are tied, we prune the node __closer to / farther from__ the root.

When pruning a decision tree, each split is replaced with the most common label among the training data points that end up at each node. Splits are compared based on their reduction in validation error rate and if two splits are tied, we prune the node closer to the root.

1.5. The ID3 algorithm for learning decision trees greedily picks the split with the best mutual information at every node. In this question, you'll see an example of when this strategy may not result in a globally optimal tree.

(a) **Drawing:** On the training dataset below, draw the decision boundaries learned by a decision tree with a max depth of 2. The decision tree uses mutual information as its splitting criterion and breaks ties by choosing +. You may **not** use a feature twice in a single path from the root node to a leaf node in the tree.





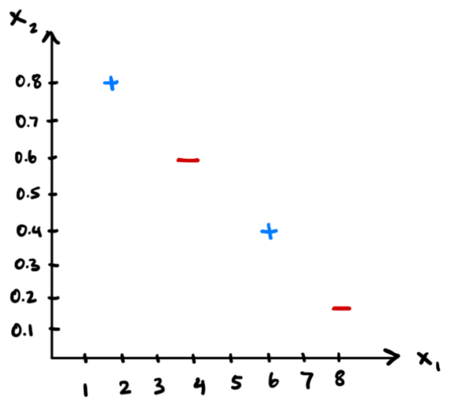
Solution:

(b) **Numerical answer:** What is the training error of the tree drawn in part (a)?

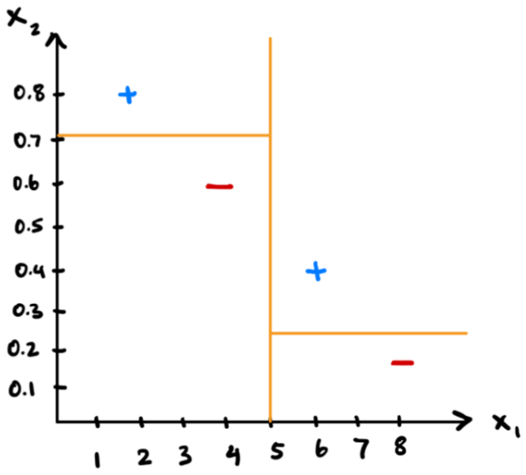
$\boxed{}$

1/4

(c) **Drawing:** On the training dataset below, draw the decision boundaries of the globally optimal tree–that is, the tree with the lowest training error. The same criteria apply from above: max depth of 2, ties are broken by in favor of +, and a feature may only be used once in a path from root to leaf.



Solution:



(d) **Numerical answer:** What is the training error of the tree drawn in part (c)?

0

2 K Nearest Neighbors (0 points)

2.1. Consider the following training dataset for a regression task:

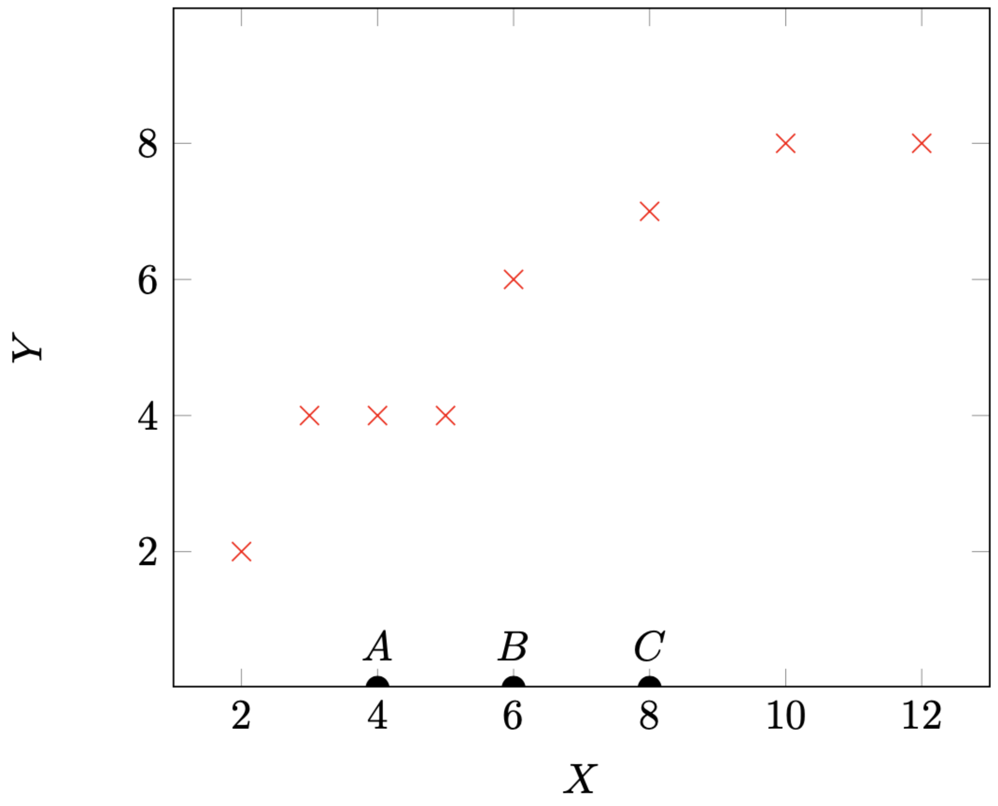
$$\mathcal{D} = \left\{ \left(x^{(1)}, y^{(1)}\right), \left(x^{(2)}, y^{(2)}\right), \cdots, \left(x^{(N)}, y^{(N)}\right) \right\}$$

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

For regression with k-nearest neighbors, 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 $\mathcal{N}$ is the neighborhood function i.e., $\mathcal{N}(x, \mathcal{D})$ is the set of indices of the k closest training points to x.



In the dataset shown above, the red $\times$'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. Assume ties are broken by selecting the point with the lower x value.

(a) **Numerical answer:** When $k = 1$, what is the mean squared error on the training set?

0

(b) **Numerical answer:** When $k = 2$, what is the predicted value at A?

4

(c) **Numerical answer:** When $k = 2$, what is the predicted value at B?

5

(d) **Numerical answer:** When $k = 3$, what is the predicted value at C?

7

(e) **Numerical answer:** When $k = 8$, what is the predicted value at C?

5.375

(f) **Math:** 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 $\mathcal{N}()$.

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

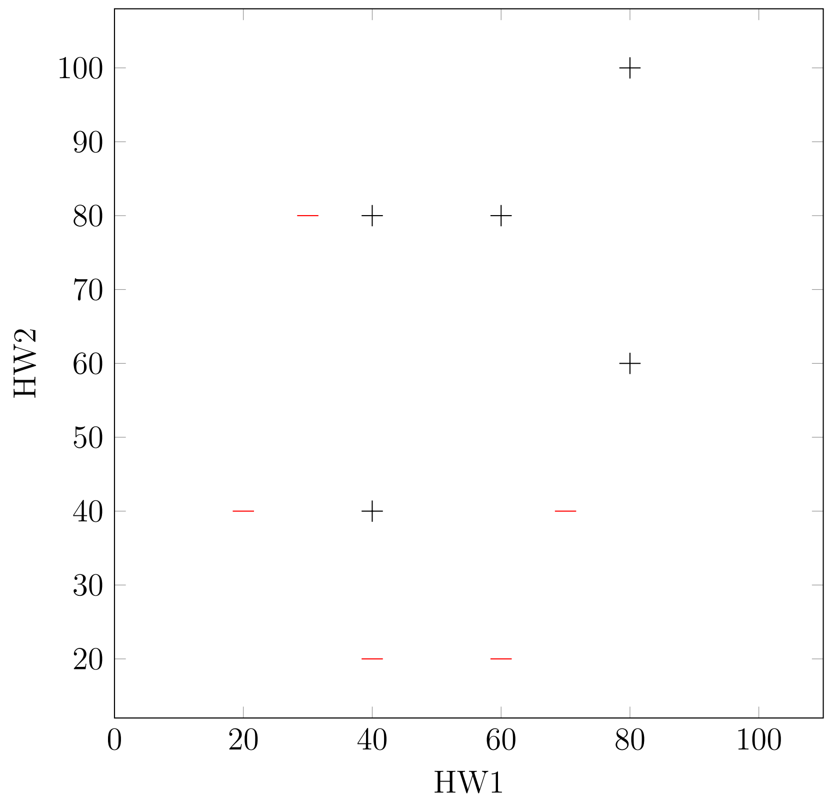
2.2. **Select one:** Imagine you are using a k-Nearest Neighbor classifier on a dataset with lots of noise. You want your classifier to be *less* sensitive to the noise. Which of the following changes is likely to help achieve this goal and what is the corresponding effect on the prediction time?

- ○ Increase the value of $k \rightarrow$ Increase in prediction time
- ○ Decrease the value of $k \rightarrow$ Increase in prediction time
- ○ Increase the value of $k \rightarrow$ Decrease in prediction time
- ○ Decrease the value of $k \rightarrow$ Decrease in prediction time

Increase the value of $k \rightarrow$ Increase in prediction time

2.3. You have just enrolled into your favourite course at CMU - Introduction to Machine Learning 10-301/601 - but you have not yet decided if you want to take it for a grade or as pass/fail. You want to use your performance in HW1 and HW2 to make this decision. You follow a general rule that if you can get at least a B in the course, you will take it for a grade, and if not, you will take it as pass/fail.

You have just learned the new classification technique, k-NN, and wish to employ it to make your decision. You start by collecting data from 10 prior students: you measure their performance on HW1 and HW2, along with their final letter grades. You then create a binary label based on the final grades such that you assign a label of $+$ if the final grade is at least a B and $-$ otherwise. Next, you train a k-NN model on this data set using the **Euclidean distance metric**.



(a) **Numerical answer:** What is the training error rate on this dataset for a k-NN model where $k = 1$?

0

(b) **Numerical answer:** What is the training error rate on this dataset for a k-NN model where $k = 3$?

2/10, (30,80) and (40, 40) will be miss-classified

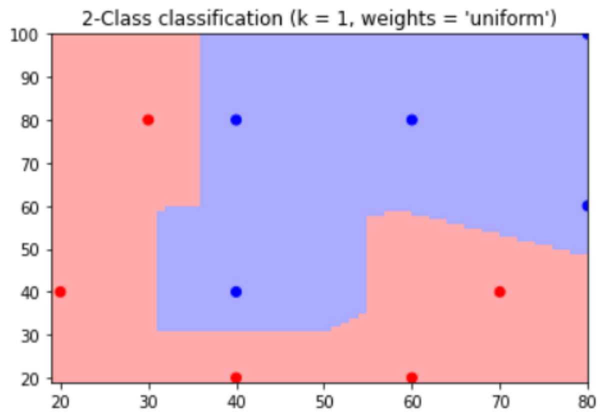
(c) **True or False:** Using Euclidean distance as the distance measure, the decision boundary of a k-NN model with $k = 1$ is a piece-wise straight line, that is, it contains only straight line segments. **Justify your answer.**

○ True

○ False

True. Because 1-NN decision boundary always follows a boundary equidistant from the two closest points in that region.

(d) **Drawing:** In the image above, draw a rough decision boundary for k = 1. Clearly label the + and - sides of the decision boundary.



(e) You have scored 60 in HW1 and 40 in HW2, and you now want to predict if your final grade would be at least a B.

i. **Numerical answer:** What would be the predicted class when $k = 1$?

$\boxed{-}$

ii. **Numerical answer:** What would be the predicted class when $k = 3$?

$\boxed{-}$

(f) **Short answer:** Looking at the training errors, you choose the model with k = 1 as it has the lowest training error. Do you think this is the right approach to select a model? Why or why not?

No, we would use validation dataset (validation error) to choose k as k is a hyper parameter.

2.4. **Select all that apply:** Which of the following statements about k-NN models is/are *always* true?

- ☐ Decreasing k makes a k-NN model have simpler or less complex decision boundaries.
- ☐ Increasing k makes a k-NN model less sensitive to outliers.
- ☐ k-NNs can be applied to classification problems but not regression problems.
- ☐ k-NNs can be applied to datasets with real-valued features but not categorical features.
- ☐ None of the above

B

3 Model Selection and Errors (0 points)

3.1. **Train and test errors:** In this problem, we will see how you can debug a classifier by looking at its train and test errors. Consider a classifier trained until 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) **Short Answer:** What is this scenario called?

overfitting

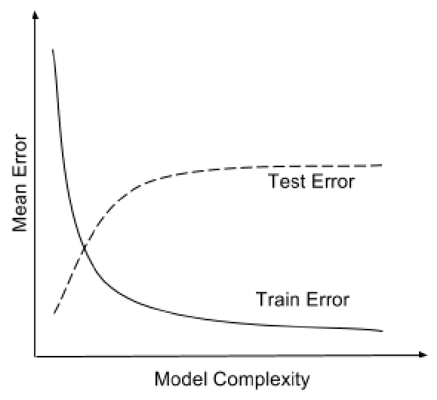
(b) **Select all that apply:** Which of the following actions are likely to address the issue you identified in the previous question?

- ☐ Increasing the training data size.
- ☐ Decreasing the training data size.
- ☐ Increasing model complexity (For example, if your classifier is a decision tree, increase the depth).
- ☐ Decreasing model complexity.
- ☐ Training on a combination of $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$ and test on $\mathcal{D}^{\text{test}}$
- ☐ None of the above

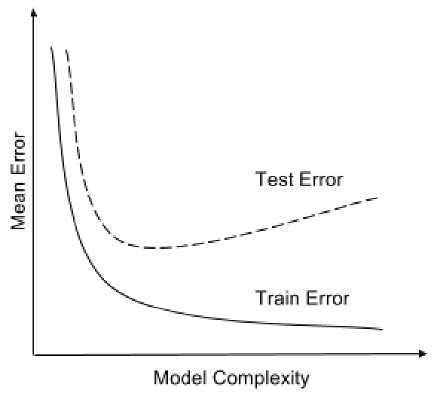
a and d

The model is overfitting. In order to address the problem, we can either increase training data size or decrease model complexity. We should never do (e), the model shouldn't see any testing data in the training process.

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



(a)



(b)

◯ Plot A

◯ Plot B

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

3.2. What are the effects of the following on overfitting? Choose the best answer.

(a) Increasing the max depth a decision tree is allowed to grow to.

- ◯ Less likely to overfit
- ◯ More likely to overfit

More likely to overfit

(b) Increasing the minimum mutual information threshold required to add a split to a decision tree.

- ◯ Less likely to overfit
- ◯ More likely to overfit

Less likely to overfit

(c) Increasing the minimum number of data points required to add a split to a decision tree.

- ◯ Less likely to overfit
- ◯ More likely to overfit

Less likely to overfit

(d) Increasing k in k-nearest neighbor.

- ◯ Less likely to overfit
- ◯ More likely to overfit

Less likely to overfit

(e) Increasing the training data size for decision trees. Assume that training data points are drawn independently from the true data distribution.

- ◯ Less likely to overfit
- ◯ More likely to overfit

Less likely to overfit

(f) Increasing the training data size for 1-nearest neighbor. Assume that training data points are drawn independently from the true data distribution.

- ◯ Less likely to overfit
- ◯ More likely to overfit

Less likely to overfit

3.3. 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.

(a) **Numerical answer:** 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 maximum 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.

$\boxed{3}$

Round $100/4/8 = 3.125$ down to 3.

(b) **Short answer:** In 1-2 concise sentences, describe one advantage of increasing the value of α.

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

3.4. **Select one:** When implementing k-Fold Cross-Validation for a Linear Regression model with a 1-dimensional input feature, which of the following best describes the correct procedure?

- ◯ The dataset is divided into k contiguous blocks, where each block consists of adjacent data points based on the sorting of the 1-dimensional feature values. The model is trained k times, each time using a different block as the test set and the remaining blocks for training.
- ◯ The dataset is randomly partitioned into k equal-sized subsets. For each fold, the Linear Regression model is trained on k-1 subsets and validated on the remaining subset to assess the model's performance.
- ◯ The model is trained once using the entire dataset as the training set and then tested k times, each time with a different single data point as the test set.

◯ We fit a Linear Regression model to every possible subset of size k of the training data (resulting in $\binom{N}{k}$ models) and then average their errors on a held-out test dataset.

B

1. Randomly split the dataset into k equal-sized folds.
2. For each fold:
   (a) Treat the fold as the validation set, and the remaining k-1 folds as the training set.
   (b) Train the model on the k-1 training sets and evaluate it on the validation set.
   (c) Retain the evaluation score and discard the model.
3. The performance measure reported by k-Fold Cross-Validation is then the average of the values computed in the loop. This method ensures that each data point is used exactly once as part of the validation set, providing a comprehensive assessment of the model's performance.

4 Perceptron (0 points)

4.1. Suppose you are given the following dataset:

Example Number	X_1	X_2	Y
1	-1	2	-1
2	-2	-2	+1
3	1	-1	+1
4	-3	1	-1

You wish to perform the Batch Perceptron algorithm on this data. Assume you start with initial weights $\theta^T = [0, 0]$ and bias $b = 0$, and that you pass through all of the examples in order of their example number.

(a) **Numerical answer:** What would be the updated weight vector θ after we pass example 1 through the Perceptron algorithm?

$[1, -2]$

(b) **Numerical answer:** What would be the updated bias b after we pass example 1 through the Perceptron algorithm?

-1

(c) **Numerical answer:** What would be the updated weight vector θ after we pass example 2 through the Perceptron algorithm?

$[1, -2]$

(d) **Numerical answer:** What would be the updated bias b after we pass example 2 through the Perceptron algorithm?

-1

(e) **Numerical answer:** What would be the updated weight vector θ after we pass example 3 through the Perceptron algorithm?

$[1, -2]$

(f) **Numerical answer:** What would be the updated bias b be after we pass example 3 through the Perceptron algorithm?

$\boxed{}$

-1

(g) **True or False:** Your friend stops you here and tells you that you do not need to update the Perceptron weights or bias anymore; is this true or false?

◯ True

◯ False

True, all points are classified correctly.

4.2. **Select all that apply:** Let $S = \{(\mathbf{x}^{(1)}, y^{(1)}), \cdots, (\mathbf{x}^{(n)}, y^{(n)})\}$ be n linearly separable points by a separator through the origin in $\mathbb{R}^d$. Let S' be generated from S as: $S' = \{(c\mathbf{x}^{(1)}, y^{(1)}), \cdots, (c\mathbf{x}^{(n)}, y^{(n)})\}$, where $c > 1$ is a constant. Suppose that we would like to run the perceptron algorithm on both data sets separately, with the same initial $\boldsymbol{\theta}$ from class, and that the perceptron algorithm converges on S. Which of the following statements are true?

□ The mistake bound of perceptron on S' is larger than the mistake bound on S

□ The perceptron algorithm when run on S and S' returns the same classifier, modulo constant factors (i.e., if $\mathbf{w}_S$ and $\mathbf{w}'_S$ are outputs of the perceptron for S and S', then $\mathbf{w}'_S = c_1\mathbf{w}_S$ for some constant c_1).

□ The perceptron algorithm converges on S'.

□ None of the above.

B and C are true. Simply follow the perceptron update rule and we see that the update on $\mathbf{w}_S$ and $\mathbf{w}_{S'}$ is identical up to the constant c. A is false as the maximum margin between any point to the decision hyperplane is also scaled up by c, and the mistake bound is unchanged.

4.3. **True or False:** Given a linearly separable dataset, the convergence time of the perceptron algorithm depends on the sample size n.

◯ True

◯ False

False. For a linearly separable dataset, the runtime of the perceptron algorithm does not depend on the size of the training data. The perceptron converges after making mistakes. The upper bound on number of mistakes is decided by the distance of the furthest point from the origin and the margin only.

4.4. **Select all that apply:** Which of the following are inductive biases of the perceptron algorithm?

□ Most of the cases in a small neighborhood in feature space belong to the same class.

□ The true decision boundary is linear.

□ We prefer to correct the most recent mistakes.

- ☐ We prefer the simplest hypothesis that explains the data.
- ☐ None of the above.

BC

4.5. **True or False:** All *data points* $(\mathbf{x}, y)$ that the Perceptron learning algorithm sees during training affect the weights equally.

- ◯ True
- ◯ False

False: only mistakes affect the Perceptron weights

5 MLE/MAP (0 points)

5.1. Magnetic Resonance Imaging (MRI) scans are commonly used to generate detailed images of patients' internal anatomy at hospitals. The scanner returns an image with N pixels. For each pixel, we extract the noise from that pixel to obtain a vector of noise terms $\mathbf{x} \in \mathbb{R}^N$ s.t. $\forall\ i \in \{1...N\}$, $x_i \geq 0$ and x_i is independent and identically distributed and follows a Rayleigh distribution. The probability density function of a Rayleigh distribution is given by:

$$f(x \mid \sigma) = \frac{x}{\sigma^2} \exp\left(\frac{-x^2}{2\sigma^2}\right)$$

for scale parameter $\sigma \geq 0$ and $x \geq 0$.

(a) **Math:** Write the log-likelihood $\ell(\sigma)$ of a noise vector $\mathbf{x}$ obtained from one image. Report your answer in terms of the variables x_i, i, N, σ, the function $\exp(\cdot)$, and any constants you may need. For full credit you must push the log through to remove as many multiplications/divisions as possible.

$$\ell(\sigma) = \sum_{i=1}^{N} \left[\log x_i - 2\log\sigma - \frac{x_i^2}{2\sigma^2}\right]$$

(b) **Math:** Report the maximum likelihood estimator of the scale parameter, σ, for a single image's noise vector $\mathbf{x}$.

$$0 = \frac{\partial}{\partial \sigma} \sum_{i=1}^{N} \log p(x_i \mid \sigma) = \sum_{i=1}^{N} \frac{-2}{\sigma} + \frac{x_i^2}{\sigma^3}$$

$$\implies \hat{\sigma} = \left[\frac{1}{2N} \sum_{i=1}^{N} x_i^2 \right]^{\frac{1}{2}}$$

5.2. Suppose a random variable k follows a **Poisson distribution** with unknown rate parameter λ:

$$p(k \mid \lambda) = \frac{\lambda^k e^{-\lambda}}{k!} \qquad k = 1, 2, \ldots$$

The Poisson distribution is used for modeling the number of times an event occurs within a fixed time interval given a mean occurrence rate assuming that the occurrences are independent.

A conjugate prior for the rate parameter in a Poisson likelihood is a **gamma distribution** with shape parameter α and rate parameter β:

$$f(\lambda \mid \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} \lambda^{\alpha-1} e^{-\beta\lambda} \qquad \lambda > 0$$

where Γ is some normalizing constant that does not depend on λ.

(a) **Math:** Suppose we receive a set of N observations of k: $\mathcal{D} = \{k_1, k_2, \ldots, k_N\}$; assume the observations are independent and identically distributed. Using the Poisson distribution and gamma prior with parameters α and β, derive an expression for the unnormalized log posterior of λ i.e. the sum of the log prior and the log likelihood of $\mathcal{D}$.

Express your answer in terms of α, β, λ, $k_1, k_2, \ldots, k_N$, N and Γ; simplify your answer as much as possible.

$$\log p(\lambda \mid \alpha, \beta) + \log \prod_{i=1}^{N} P(k_i \mid \lambda) = \log\left(\frac{\beta^\alpha}{\Gamma(\alpha)} \lambda^{\alpha-1} e^{-\beta\lambda}\right) + \sum_{i=1}^{N} \log\left(\frac{\lambda^{k_i} e^{-\lambda}}{k_i!}\right)$$

$$= \alpha \log \beta + (\alpha - 1) \log \lambda - \beta\lambda - \log \Gamma(\alpha) - N\lambda + \sum_{i=1}^{N} k_i \log \lambda - \log(k_i!)$$

(b) **Math:** Take the partial derivative of the expression you derived in the previous part with respect to λ.

Express your answer in terms of α, β, λ, $k_1, k_2, \ldots, k_N$, and N; simplify your answer as much as possible.

$$\frac{\alpha - 1}{\lambda} - \beta - N + \sum_{i=1}^{N} \frac{k_i}{\lambda}$$

(c) **Math:** Finally, compute the MAP estimate of λ given $\mathcal{D}$ by setting the partial derivative you computed in the previous part equal to 0 and solving for λ.

Express your answer in terms of α, β, $k_1, k_2, \ldots, k_N$, and N; simplify your answer as much as possible.

$$\frac{\alpha - 1}{\hat{\lambda}} - \beta - N + \sum_{i=1}^{N} \frac{k_i}{\hat{\lambda}} = 0$$
$$\rightarrow \alpha - 1 + \sum_{i=1}^{N} k_i = (\beta + N)\hat{\lambda}$$
$$\rightarrow \hat{\lambda} = \frac{\alpha - 1 + \sum_{i=1}^{N} k_i}{\beta + N}$$

Intuitively, the MLE estimate for λ is the empirical mean of $\mathcal{D}$. The β and α parameters of the gamma distribution can be interpreted as the number of "pseudo-samples" previously observed and the total number of "hits" observed in those pseudo-samples respectively.

5.3. **True or False:** A random variable x follows a probability distribution with a single, real-valued parameter, θ. In the limit of infinitely many samples of x, the MAP estimate of θ will *always* approach the MLE estimate, regardless of your choice of prior on θ. Briefly justify your answer in 1-2 sentences.

False: if the prior does not have support over the entire domain (e.g., a uniform prior over just the range [0,1]) and the MLE estimate falls outside the prior, then the MAP estimate will approach the boundary nearest to the MLE, not the MLE estimate itself.

6 Logistic Regression and Regularization (0 points)

6.1. **Math:** A generalization of logistic regression to a multiclass settings involves expressing the per-class probabilities $P(y = c \mid x)$ as the softmax function

$$\frac{\exp(w_c^T x)}{\sum_{d \in C} \exp(w_d^T x)},$$

where c is some class from the set of all classes C.

Consider a 2-class problem with labels 0 or 1. Rewrite the above expression for this situation to derive the expressions for $P(Y = 1|x)$ and $P(Y = 0|x)$ that we have already seen in class for binary logistic regression.

$P(y = 1|x) = \frac{\exp(w_1^T x)}{\exp(w_0^T x) + \exp(w_1^T x)} = \frac{\exp((w_1 - w_0)^T x)}{1 + \exp((w_1 - w_0)^T x)} = \frac{\exp(w^T x)}{1 + \exp(w^T x)} = p$

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

6.2. **Short answer:** Given 3 data points $(1, 1), (1, 0), (0, 0)$ with labels $0, 1, 0$ respectively, consider 2 models that define $p(y = 1 \mid \mathbf{x})$:

$$\text{Model 1: } \sigma(w_1 x_1 + w_2 x_2)$$

$$\text{Model 2: } \sigma(w_0 + w_1 x_1 + w_2 x_2)$$

As usual, $\sigma(z)$ is the sigmoid function $\frac{1}{1+e^{-z}}$. Using the given dataset, suppose we learn parameters for both models, $\hat{\mathbf{w}}_1$ and $\hat{\mathbf{w}}_2$, by maximizing the conditional log-likelihood.

If we switched $(0, 0)$ to label 1 instead of label 0, would the parameters we learn for Model 1 change? What about Model 2?

The parameters for Model 1 wouldn't change because $w_1x_1 + w_2x_2 = 0$ for $(0, 0)$. Hence $p = 0.5$ irrespective of the labels or the values of $\mathbf{w}$.

Model 2 has a bias term which remains non-zero for $(0, 0)$, and can thus change the model depending on the label assigned.

6.3. Given a training dataset $\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(N)}, y^{(N)})\}$ 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 of the training dataset, assuming a parametric 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 $\mathcal{D}$ is

$$\ell(\mathbf{w}) = \sum_{i=1}^{n} y_i \log p(y_i \mid \mathbf{x}_i; \mathbf{w}) + (1 - y_i) \log(1 - p(y_i \mid \mathbf{x}_i; \mathbf{w})),$$

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) **Short answer:** Is it possible to solve for the optimal parameters $\hat{\mathbf{w}}$ in closed form? If yes, explain how and if no, describe how you would compute $\hat{\mathbf{w}}$ in practice?

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 $\hat{w}$.

(b) **Math:** For a binary logistic regression model, we predict $y = 1$ when $p(y = 1 \mid \mathbf{x}) \geq 0.5$. Show that this is a linear classifier in $\mathbf{x}$.

Using the parametric form for $p(y = 1 \mid \mathbf{x})$:

$$\begin{aligned} p(y = 1 \mid \mathbf{x}) \geq \frac{1}{2} &\implies \frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x})} \geq \frac{1}{2} \\ &\implies 1 + \exp(-\mathbf{w}^T\mathbf{x}) \leq 2 \\ &\implies \exp(-\mathbf{w}^T\mathbf{x}) \leq 1 \\ &\implies -\mathbf{w}^T\mathbf{x} \leq 0 \\ &\implies \mathbf{w}^T\mathbf{x} \geq 0, \end{aligned}$$

so we predict $\hat{y} = 1$ if $\mathbf{w}^T\mathbf{x} \geq 0$, which precisely defines a linear function of $\mathbf{x}$.

(c) Consider the case where all features are binary, i.e, $\mathbf{x} \in \{0,1\}^d$. Furthermore, suppose feature x_1 is rare and just happens to take on value 1 in the training set only when the label is also 1.

What is $\hat{w}_1$ i.e. the optimal weight on the first feature? Is the gradient ever zero for any finite value of w_1?

If a binary feature fired for only label 1 in the training set then, by maximizing the conditional log likelihood, we will make the weight associated to that feature be infinite; this implies that the gradient will never be zero: we can always improve the solution by increasing the weight. This is because, when this feature is observed in the training set, we will want to predict 1 irrespective of everything else. This is an undesired behavior from the point of view of generalization performance, as most likely we do not believe this rare feature to have that much information about class 1. Most likely, it is a spurious co-occurrence. Controlling the norm of the weight vector will prevent these pathological cases.

6.4. **Math:** Given the following dataset, $\mathcal{D}$, and a fixed parameter vector, $\mathbf{w}$, write an expression for the conditional likelihood of a binary logistic regression model on this dataset.

$$\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)} = 0), (\mathbf{x}^{(2)}, y^{(2)} = 0), (\mathbf{x}^{(3)}, y^{(3)} = 1), (\mathbf{x}^{(4)}, y^{(4)} = 1)\}$$

- Write your answer in terms of $\mathbf{w}$, $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$, $\mathbf{x}^{(3)}$, and $\mathbf{x}^{(4)}$.
- Do not include $y^{(1)}$, $y^{(2)}$, $y^{(3)}$, or $y^{(4)}$ in your answer.
- Do not try to simplify your expression.

$$\left(1-\frac{1}{1+e^{-\mathbf{w}^T x^1}}\right)\left(1-\frac{1}{1+e^{-\mathbf{w}^T x^2}}\right)\frac{1}{1+e^{-\mathbf{w}^T x^3}}\frac{1}{1+e^{-\mathbf{w}^T x^4}}$$

6.5. Suppose we apply feature engineering to a two-dimensional input, $x_1 \in \mathbb{R}$ and $x_2 \in \mathbb{R}$, mapping it to a new input vector: $\mathbf{x} = \begin{bmatrix} 1 \\ {x_1}^2 \\ {x_2}^2 \end{bmatrix}$

(a) Write an expression for the decision boundary of binary logistic regression with this feature vector $\mathbf{x}$ and the corresponding parameter vector $\boldsymbol{\theta} = [b, w_1, w_2]^T$. Assume that the decision boundary occurs when $P(Y = 1 \mid x, \boldsymbol{\theta}) = P(Y = 0 \mid x, \boldsymbol{\theta})$. Write your answer in terms of x_1, x_2, b, w_1, and w_2.

Decision boundary expression:

$0 = b + w_1 x_1^2 + w_2 x_2^2.$

(b) Assume that $w_1 > 0$, $w_2 > 0$, and $b < 0$. What is the geometric shape defined by this equation?

An ellipse

(c) If we add an L2 regularization term when learning $[w_1, w_2]^T$, what happens to the **parameters** as we increase the λ that scales this regularization term?

The magnitude of the parameters will decrease.

(d) If we add an L2 regularization term when learning $[w_1, w_2]^T$, what happens to the **decision boundary shape** as we increase the λ that scales this regularization term?

The parameters shrink, so the ellipse will get bigger.

7 Linear Regression (0 points)

7.1. **Select all that apply:** For some function $f\colon \mathbb{R} \to \mathbb{R}$, suppose the unique argmax of f is x^*. Which of the following functions are guaranteed to have the same argmax as f?

- ☐ $g(x) = f(x) + 2$
- ☐ $g(x) = 2f(x)$
- ☐ $g(x) = f(x)^2$
- ☐ $g(x) = 2^{f(x)}$
- ☐ None of the above.

A, B, and D

C is not guaranteed to have the same argmax because the square operation inverts negation e.g., if f is always negative, then its argmax is the location where f is closest to zero but the argmax of g would be the location where f is furthest from zero.

7.2. Consider linear regression on 1-dimensional data i.e., $x, y \in \mathbb{R}$. We apply linear regression in both directions on this data: we first fit a model to predict y using x and get $y = m_1 x + b_1$ as the fitted line that minimizes the mean squared error, then we fit a model to predict x using y and get $x = m_2 y + b_2$.

True or False: It is always the case that $m_2 = \frac{1}{m_1}$.

- ◯ True
- ◯ False

False: this is only true when the data lie on a perfect line; can verify this empirically with non-linear data.

7.3. **Select all that apply:** Which of the following statements about the gradient of a function is/are true?

- ☐ The gradient is the same dimensionality as the function's output.
- ☐ The gradient points in the direction of steepest descent.
- ☐ The gradient of a function always exists everywhere.
- ☐ The gradient is the limit of the slope between two points on the function as the distance between the points approaches ∞.
- ☐ None of the above.

None of the above

7.4. Instead of mean squared error, suppose instead that you fit a linear regression model by minimizing the mean cubed error:

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left(\theta^T \mathbf{x}^{(i)} - y^{(i)}\right)^3$$

(a) **Math:** What is the gradient of the mean cubed error with respect to the parameter vector θ?

$\nabla_\theta J(\theta) = \frac{3}{N} \sum_{i=1}^{N} \left(\theta^T \mathbf{x}^{(i)} - y^{(i)}\right)^2 \mathbf{x}^{(i)}$

(b) **True or False:** This loss function is more sensitive to outliers than the mean squared error.

◯ True

◯ False

True

(c) **Short answer:** In 1-2 concise sentences, briefly describe the primary issue with this objective function for regression?

This loss function is not strictly non-negative. Therefore, we could have underestimations canceling out overestimations, resulting in a poorly fit model.

7.5. **True or False:** Consider a linear regression model with only one parameter, the intercept, i.e., $y = b$. Then, given N data points $(x^{(i)}, y^{(i)})$ (where $x^{(i)}$ is the feature and $y^{(i)}$ is the target), minimizing the sum of squared errors results in b being the *median* of the $y^{(i)}$ values. Briefly justify your answer

◯ True

◯ False

False. $\sum_{i=1}^{N}(y^{(i)} - b)^2$ is the training loss, which when differentiated and set to zero gives $b = \frac{\sum_{i=1}^{N} y^{(i)}}{N}$, the *mean* of the $y^{(i)}$ values.

7.6. You decide to optimize the function $f(x) = x^2$ with respect to x using gradient descent. You initialize $x^{(0)}$ to -1 and use a step size of $\eta^{(0)} = 1.5$.

(a) **Fill in the blanks:** Fill in the table below with the results you get from running gradient descent in this setting for two iterations; most of the 0th iteration has been filled in on your behalf; you should fill in the missing element in this line along with all the other missing elements.

t	$x^{(t)}$	$f(x^{(t)})$	$\nabla_x f(x^{(t)})$
0	-1	1	________
1	________	________	________
2	________	________	________

t	$x^{(t)}$	$f(x^{(t)})$	$\nabla_x f(x^{(t)})$
0	-1	1	-2
1	2	4	4
2	-4	16	-8
3	8	64	16

(b) **Select all that apply:** Based on your findings from the previous part, which of the following changes could help you achieve better results?

☐ Decreasing the step size.

☐ Running gradient descent for more iterations.

☐ Moving the initial value $x^{(0)}$ further from the origin.

☐ Negating the function and running gradient *ascent*.

☐ None of the above.

A; This step size is too large so decreasing it to anything < 1 would improve the results.

7.7. **Math:** 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 compute the minimizer of the mean squared error. 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.

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

8 Optimization (0 points)

8.1. (a) **Select all that apply:** 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:

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

- ☐ $f(x) = x + b$ 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

A, B, D

C is nonconvex for $a < 0$

(b) **Select all that apply:** Suppose we are trying to minimize the convex function $f(z) = z^2$ using gradient descent. Let α be the learning rate and assume that we use an inital value of $z^{(0)} = 1$.

For which values of α will graident descent converge to the optimal value, $x^* = 0$?

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

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

$\alpha = 1$ causes gradient descent to bounce between 1 and -1 while $\alpha = 2$ causes gradient descent to diverge.

8.2. **Fill in the Blanks:** Complete the following sentence by circling the best option in each square (options are separated by "/"s):

The mean squared error objective function for linear regression is $\boxed{\text{non-convex / convex / strictly convex}}$ which means that gradient descent will converge to a $\boxed{\text{global minimum / local minimum which might not be a global minimum}}$ if it converges; however, a global minimum $\boxed{\text{does not / does}}$ always exist.

The mean squared error objective function for linear regression is convex which means that gradient descent will converge to a global minimum; however, a global minimum does always exist.

8.3. **Select all that apply:** Which of the following statements about gradient descent (GD) and stochastic gradient descent (SGD) are correct?

- ☐ 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.
- ☐ Both SGD and GD are guaranteed to converge if and only if the objective function is strictly convex.
- ☐ None of the above.

B.

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. D is incorrect because GD can converge if the function is convex but not strictly convex.

8.4. **True or False:** For a given dataset and objective function, one *epoch* of stochastic gradient descent (i.e., one pass through the entire dataset) will always have the same big-O computational cost as one iteration of gradient descent.

- ◯ True
- ◯ False

True

9 Feature Engineering and Regularization (0 points)

9.1. Suppose you have D-dimensional data points $\mathbf{x} = [x_1, x_2, \dots, x_D]^T$ and you want to apply a 2-dimensional polynomial feature expansion to them, ϕ_2. This feature expansion contains all first-order terms as well as *all possible second-order terms*, including combinations of features i.e.

$$\{x_4, x_1^2, x_2 x_3\} \subset \phi_2(\mathbf{x})$$

(a) **Math:** What is the dimensionality of $\phi_2(\mathbf{x})$? Express you answer in terms of D, the dimensionality of $\mathbf{x}$.

D first order terms + D squared terms + $\binom{D}{2}$ combination terms = $\frac{D(D+3)}{2}$

(b) **Math:** Given a dataset consisting of N data points, suppose a unique closed-form solution exists for linear regression using the ϕ_2-transformation. What is the big-O computational cost of computing $(\Phi^T\Phi)^{-1}$, where Φ is the design matrix of the transformed data points i.e.,

$$\Phi = \begin{bmatrix} 1 & \phi_2(\mathbf{x}^{(1)}) \\ 1 & \phi_2(\mathbf{x}^{(2)}) \\ \vdots & \vdots \\ 1 & \phi_2(\mathbf{x}^{(N)}) \end{bmatrix}$$

Express you answer in terms N and D.

Hint: don't forget about the computational costs associated with the bias or intercept parameters.

$$O\left(N\left(\frac{D(D+3)}{2}+1\right)^2 + \left(\frac{D(D+3)}{2}+1\right)^3\right) = O(ND^4 + D^6)$$

9.2. **Model Complexity:** In this question we will consider the effect of increasing the model complexity, while keeping the size of the training dataset fixed. To be concrete, consider a classification task on the real line $\mathbb{R}$ with some unknown distribution over data points D and unknown target function $c^* : \mathbb{R} \to \{\pm 1\}$.

Suppose we have a randomly sampled dataset 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 data points and then runs logistic regression. A new example is classified by first applying the feature map ϕ_d and then using the learned classifier.

(a) For a given dataset S, is it possible for the training error to increase when we increase the degree d of the feature map? Briefly justify your answer in 1 to 2 sentences.

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 will not increase for any given sample S.

(b) Suppose we plot the true error of our algorithm as a function of d: we observe that it initially decreases and then increases as we increase the degree d. Briefly explain this trend in 2 to 3 sentences.

When the dimension d is small, the true error is high because 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.

9.3. **Short Answer:** Your friend is training a logistic regression model with ridge regularization, where λ is the regularization constant. They run cross-validation for $\lambda = [0.01, 0.1, 1, 10]$ and compare train, validation and test errors. They choose $\lambda = 0.01$ because that had the lowest *test* error.

However, you observe that the test error linearly increases from $\lambda = 0.01$ to 10 and thus, there exists a value of $\lambda < 0.01$ that gives a lower test error. You tell your friend that they should run the cross-validation for $\lambda = [0.0001, 0.001, 0.01]$ to get the optimal model.

Do you think you did the right thing by giving your friend this suggestion? Briefly justify your answer in 1 to 2 concise sentences.

No. because we should not be using test error at all in making any model selection decisions.

10 Neural Networks (0 points)

10.1. **Matching:** Match the corresponding neural network component to its role in the neural network.

(a) Cross-Entropy

(b) Identity

(c) Mean Absolute Error

(d) Mean Squared Error

(e) ReLU

(f) Sigmoid

(g) Softmax

(h) Stochastic Gradient Descent

(i) Tanh

Activation Function

Loss Function

Optimizer

Activation function: Identity, ReLU, Sigmoid, Tanh, Softmax; Loss function: Cross-entropy, Mean Absolute Error, Mean Squared Error; Optimizer: Stochastic Gradient Descent

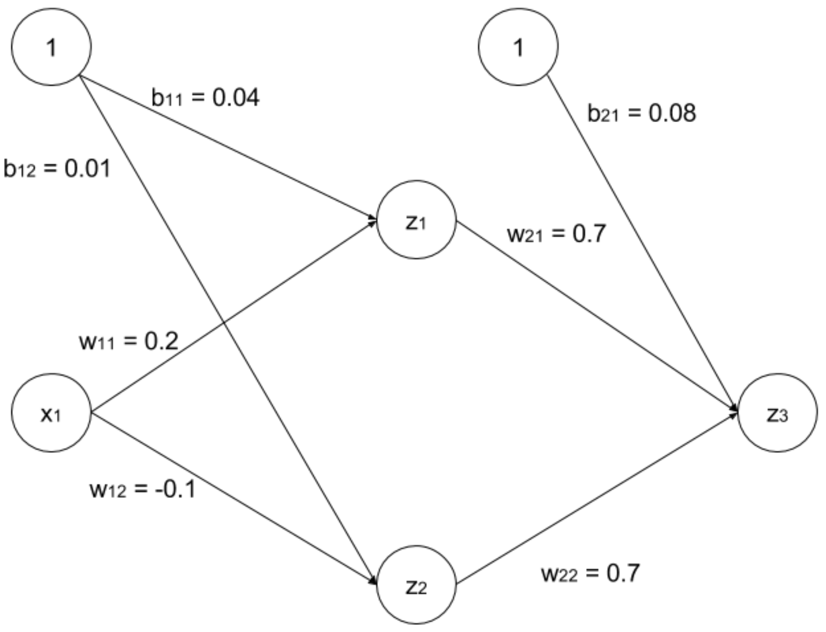


Figure 3: Neural Network

10.2. Consider the neural network architecture shown above for a binary classification problem. The values for the weights are shown in the figure. We define:

$a_1 = w_{11}x_1 + b_{11}$

$a_2 = w_{12}x_1 + b_{12}$

$a_3 = w_{21}z_1 + w_{22}z_2 + b_{21}$

$z_1 = \text{ReLU}(a_1)$

$z_2 = \text{ReLU}(a_2)$

$z_3 = \sigma(a_3)$, $\sigma(x) = \frac{1}{1+e^{-x}}$

where $\text{ReLU}(x) = \max(0, x)$.

(a) **Math:** For $x_1 = 0.3$, compute z_3 in terms of e.

$\boxed{}$

$z_3 = \frac{1}{1+e^{-0.15}}$

(b) **Select one:** Which class does the network predict for the data point $x_1 = 0.3$, assuming that $\hat{y} = 1$ if $z_3 > \frac{1}{2}$ and otherwise, $\hat{y} = 0$.

◯ 1

○ 0

$\hat{y}(x_1 = 0.3) = 1$

(c) **Math:** Perform backpropagation on the bias term b_{21} by deriving the expression for the gradient of the loss function $L(y, z_3)$ with respect to the bias term b_{21}, $\frac{\partial L}{\partial b_{21}}$.

Express your answer in terms of partial derivatives of the form $\frac{\partial \alpha}{\partial \beta}$, 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 backpropagation algorithm 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.

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

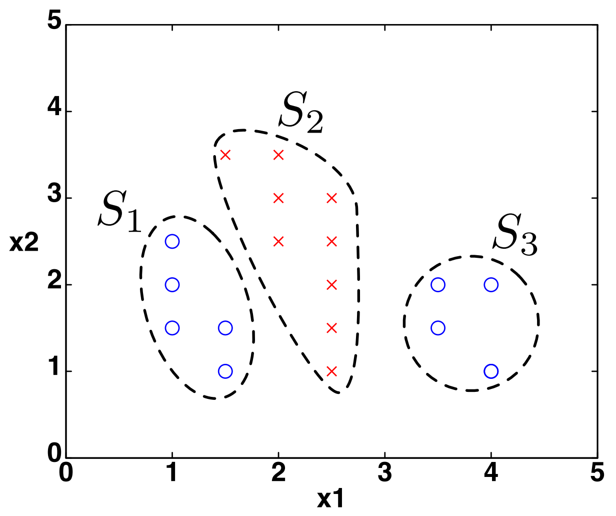
(d) **Math:** Perform backpropagation on the bias term b_{12} by deriving the expression for the gradient of the loss function $L(y, z_3)$ with respect to the bias term b_{12}, $\frac{\partial L}{\partial b_{12}}$.

Express your answer in terms of partial derivatives of the form $\frac{\partial \alpha}{\partial \beta}$, 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 backpropagation algorithm 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.

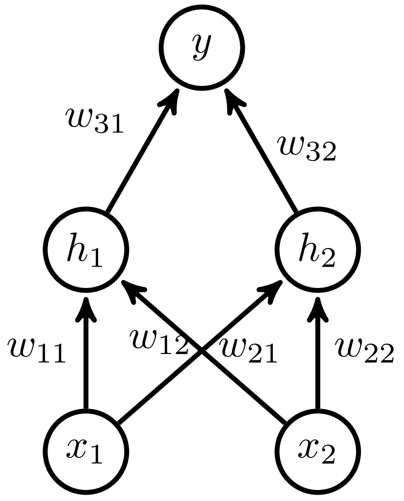
$$\frac{\partial L}{\partial b_{12}} = \frac{\partial L}{\partial z_3} \frac{\partial z_3}{\partial a_3} \frac{\partial a_3}{\partial z_2} \frac{\partial z_2}{\partial a_2} \frac{\partial a_2}{\partial b_{12}}$$

10.3. In this problem we will use a neural network to distinguish the crosses ($\times$) from the circles ($\circ$) in the simple data set shown in Figure 4a. Even though the crosses and circles are not linearly separable, we can break the examples into three groups, S_1, S_2, and S_3 (shown in Figure 4a) so that S_1 is linearly separable from S_2 and S_2 is linearly separable from S_3. We will exploit this fact to design weights for the neural network shown in Figure 4b in order to correctly classify this training dataset. For all nodes, we will use the threshold activation function

$$\phi(z) = \begin{cases} 1 & z > 0 \\ 0 & z \leq 0. \end{cases}$$

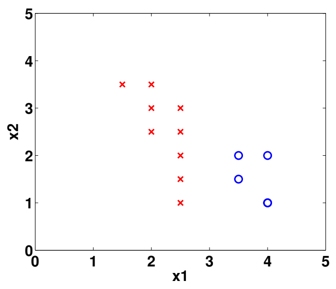


(a) The data set with groups S_1, S_2, and S_3.

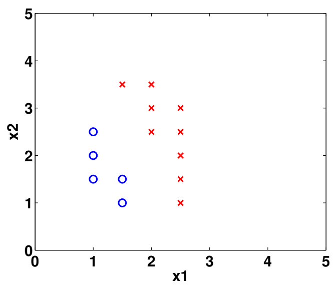


(b) The neural network architecture

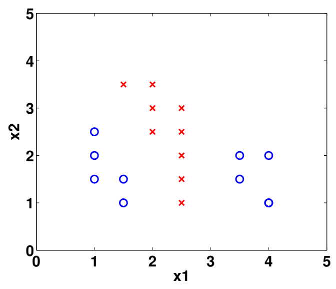
Figure 4



(a) Set S2 and S3



(b) Set S1 and S2

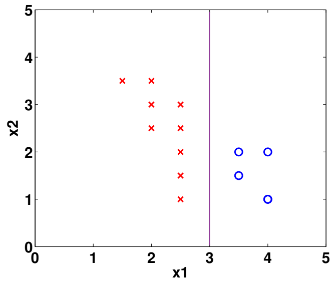


(c) Set S1, S2 and S3

Figure 5: NN classification.

(a) First we will set the parameters w_{11}, w_{12} and b_1 of the neuron labeled h_1 so that its output $h_1(x) = \phi(w_{11}x_1 + w_{12}x_2 + b_1)$ forms a linear separator between the sets S_2 and S_3.

i. On Fig 5a, draw a linear decision boundary that separates S_2 and S_3.

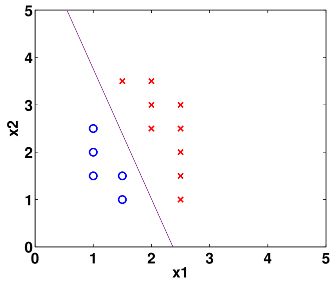


ii. Write down a possible setting of the weights w_{11}, w_{12}, and b_1 such that $h_1(x) = 0$ for all points in S_3 and $h_1(x) = 1$ for all points in S_2.

$\boxed{}$ $w_{11} = -1, w_{12} = 0, b_1 = 3$

(b) Next we will set the parameters w_{21}, w_{22} and b_2 of the neuron labeled h_2 so that its output $h_2(x) = \phi(w_{21}x_1 + w_{22}x_2 + b_2)$ forms a linear separator between the sets S_1 and S_2.

i. On Fig 5b, draw a linear decision boundary that separates S_1 and S_2.



ii. Write down a possible setting of the weights w_{21}, w_{22}, and b_2 such that $h_2(x) = 0$ for all points in S_1 and $h_2(x) = 1$ for all points in S_2.

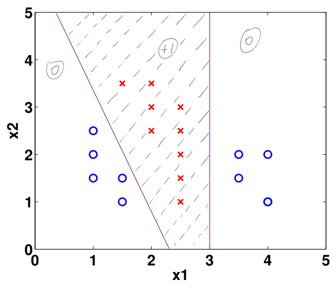
$\boxed{}$ $w_{21} = 3, w_{22} = 1, b_2 = -7$

(c) Now we have two classifiers h_1 (to classify S_2 from S_3) and h_2 (to classify S_1 from S_2). We will set the weights of the final neuron of the neural network based on the results from h_1 and h_2 to classify the crosses from the circles. Let $h_3(x) = \phi\big(w_{31}h_1(x) + w_{32}h_2(x) + b_3\big)$.

i. Write down a possible setting of the weights w_{31}, w_{32}, b_3 such that $h_3(x)$ correctly classifies the entire data set.

$w_{31} = 1, w_{32} = 1, b_3 = -1.5$

ii. Draw your decision boundary in Fig 5c.



10.4. Consider the following neural network for a 2-D input, $x_1 \in \mathbb{R}$ and $x_2 \in \mathbb{R}$ where:

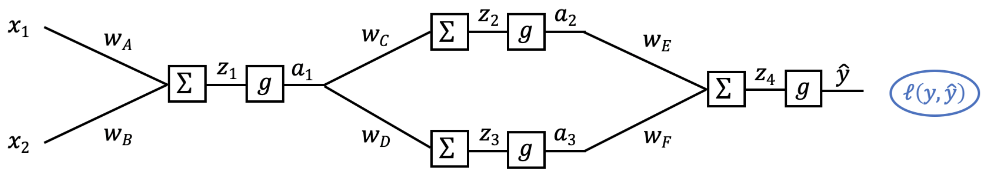


Figure 9: Neural Network

- All occurrences of the function g are the same arbitrary non-linear activation function with no parameters
- $\ell(y, \hat{y})$ is an arbitrary loss function with no parameters, and:

$$z_1 = w_A x_1 + w_B x_2,\ a_1 = g(z_1)$$
$$z_2 = w_C a_1,\ a_2 = g(z_2)$$
$$z_3 = w_D a_1,\ a_3 = g(z_3)$$
$$z_4 = w_E a_2 + w_F a_3,\ \hat{y} = g(z_4)$$

Note: There are no bias terms in this network.

(a) What is the chain of partial derivatives needed to calculate the derivative $\frac{\partial \ell}{\partial w_E}$?

Your answer should be in the form: $\frac{\partial \ell}{\partial w_E} = \frac{\partial ?}{\partial ?}\frac{\partial ?}{\partial ?}\dots$ Make sure each partial derivative $\frac{\partial ?}{\partial ?}$ in your answer cannot be decomposed further into simpler partial derivatives. **Do not evaluate the derivatives.** Be sure to specify the correct subscripts in your answer.

$\frac{\partial \ell}{\partial w_E} =$

$$\frac{\partial \ell}{\partial w_E} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_4} \frac{\partial z_4}{\partial w_E}$$

(b) The network diagram from above is repeated here for convenience: What is the

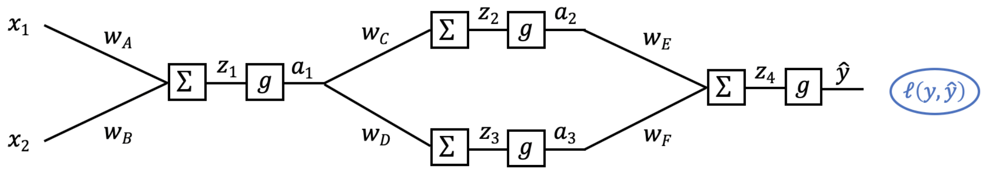


Figure 10: Neural Network

chain of partial deriviatives needed to calculate the derivative $\frac{\partial \ell}{\partial w_C}$?
Your answer should be in the form:

$$\frac{\partial \ell}{\partial w_C} = \frac{\partial ?}{\partial ?}\frac{\partial ?}{\partial ?}\dots$$

Make sure each partial derivative $\frac{\partial ?}{\partial ?}$ in your answer cannot be decomposed further into simpler partial derivatives. **Do not evaluate the derivatives**. Be sure to specify the correct superscripts in your answer.

$\frac{\partial \ell}{\partial w_C} =$

$\frac{\partial \ell}{\partial w_C} = \frac{\partial \ell}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial z_4}\frac{\partial z_4}{\partial a_2}\frac{\partial a_2}{\partial z_2}\frac{\partial z_2}{\partial w_C}$

(c) We want to modify our neural network objective function to add an L2 regularization term on the weights. The new objective is:

$$\ell(y,\hat{y}) + \lambda\frac{1}{2}\|w\|_2^2$$

where λ (lambda) is the regularization hyperparamter and $\mathbf{w}$ is all of the weights in the neural network stacked into a single vector, $\mathbf{w} = [w_A, w_B, w_C, w_D, w_E, w_F]^T$.

Write the right-hand side of the new gradient descent update step for weight w_C given this new objective function. You may use $\frac{\partial \ell}{\partial w_C}$ in your answer.

Update: $w_C \leftarrow \dots$

Update for w_C: $w_C \leftarrow w_C - \alpha\left(\frac{\partial \ell}{\partial w_C} + \lambda w_C\right)$

10.5. Define a function `floor` : $\mathbb{R}_n \rightarrow \mathbb{R}_n$ such that

$$\texttt{floor}(\mathbf{z}) = \left[\lfloor z_i \rfloor \text{ for } 0 \leq i \leq D\right]^T$$

or essentially, a function that produces an output vector by applying $\lfloor \cdot \rfloor$ element-wise to the input vector.

Neural wants to use this function as an activation function to train his neural network. Is this possible? Explain why or why not.

Yes, it is possible. Since the function is piecewise, we will not be able to use automatic differentiation to solve the gradients, but we can still use the finite difference method to approximate the gradient and train the model.