Solutions

10-301/601 Machine Learning **Name:**
Summer 2023 **AndrewID:**
Final Practice Problems
June 7, 2024
Time Limit: N/A

Instructions:

- Fill in your name and Andrew ID above. Be sure to write neatly, or you may not receive credit for your exam.
- Clearly mark your answers in the allocated space **on the front of each page.** If needed, use the back of a page for scratch space, but you will not get credit for anything written on the back of a page. If you have made a mistake, cross out the invalid parts of your solution, and circle the ones which should be graded.
- No electronic devices may be used during the exam.
- Please write all answers in pen.
- You have N/A to complete the exam. Good luck!

Instructions for Specific Problem Types

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

Select One: Who taught this course?

● Henry Chai

○ 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 scientists?

■ Stephen Hawking

■ Albert Einstein

■ Isaac Newton

☐ 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 scientists?

■ Stephen Hawking

■ Albert Einstein

■ Isaac Newton

~~■~~ 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?

10-601 10-~~7~~601

1 Decision Trees

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.

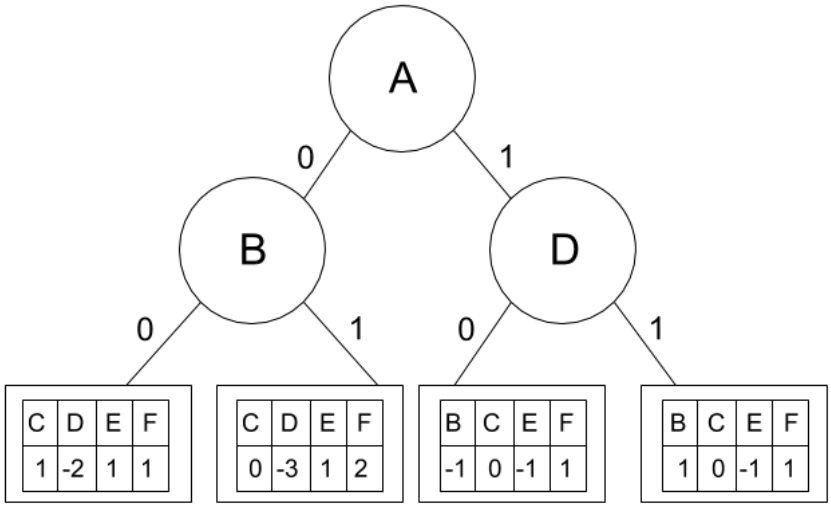


Figure 1: Perceptron Tree of depth 2

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

2. (2 points) **Select all that apply:** Given an input feature vector $\mathbf{x}$, where $\mathbf{x} \in \mathbb{R}^n$, you are tasked with predicting a label for y, where $y = 1$ or $y = -1$. You have no knowledge about the distributions of $\mathbf{x}$ 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

k-Nearest Neighbours and Decision Tree with unlimited depth, since these two methods do not making the assumption of linear separation.

3. **True or False:** The ID3 algorithm is guaranteed to find an optimal decision tree.

◯ True

◯ False

False.

4. **True or False:** One advantage of decision trees is that they are not easy to overfit.

◯ True

◯ False

False.

True or False: It is possible to have a decision tree with zero training error for this dataset. Assume only binary splits and attributes selected with replacement.

- ◯ True
- ◯ False

True

2 K Nearest Neighbors

1. **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

   A short Decision Tree with a low branching factor

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 is likely to help and with what side effect?

   ◯ 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

3. **Select all that apply:** Identify the correct relationship(s) 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

   ☐ None of the above

   A and D

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

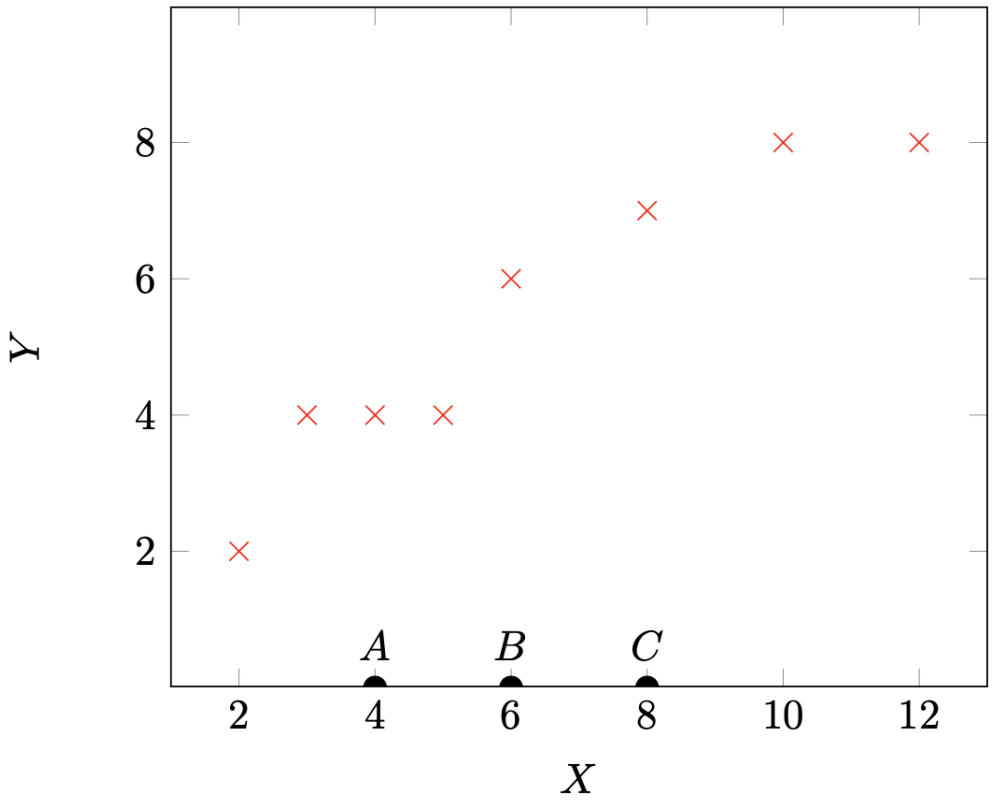
$$\mathcal{D} = \left\{ \left(x^{(1)}, y^{(1)}\right), \left(x^{(2)}, y^{(2)}\right), \cdots, \left(x^{(N)}, y^{(N)}\right) \right\}$$

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



In the above dataset, the red ×'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.

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

0 ± 0.00001

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

4 ± 0.00001

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

5 ± 0.00001

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

7 ± 0.00001

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

5.375 ± 0.1

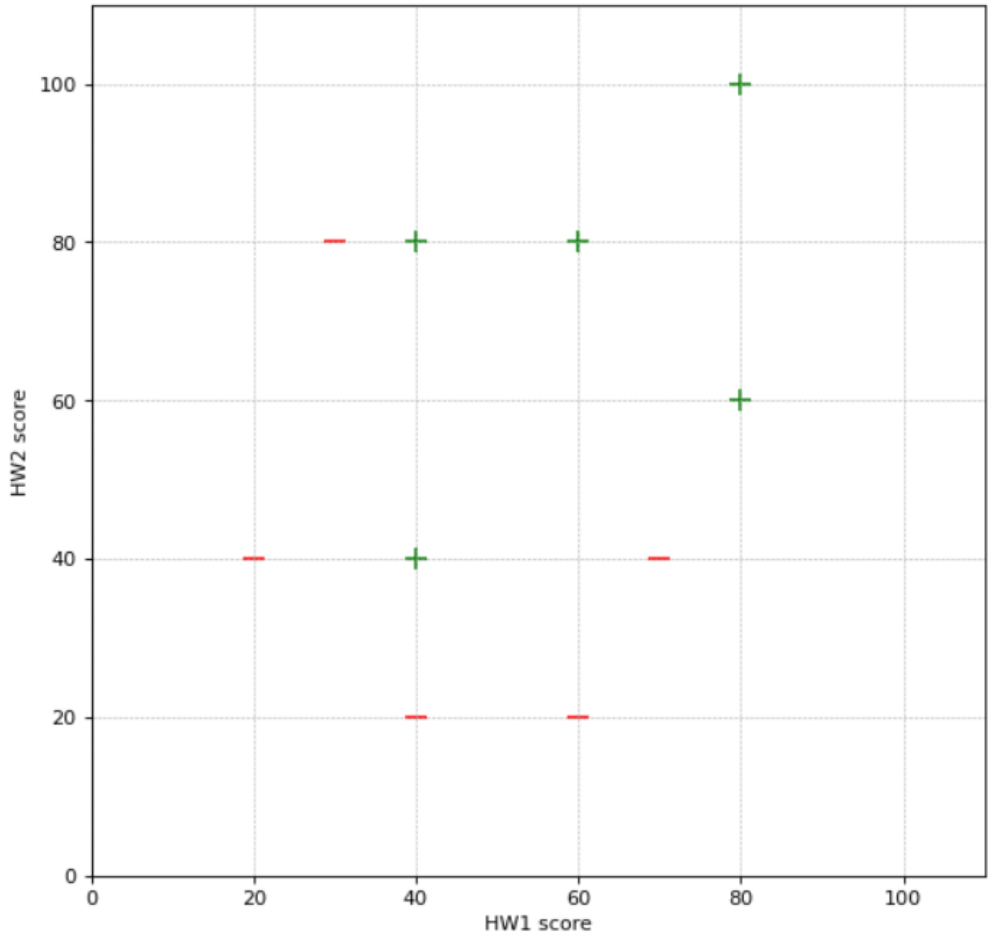
(f) **Math:** With $k = N$, for any dataset $\mathcal{D}$ with the form specified in the beginning of this question, write down a mathematical expression for the predicted value $\hat{y} = h(x)$. Your response shouldn't include a reference to the neighborhood function $\mathcal{N}()$.

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

5. 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 learnt the new classification technique, k-NN, and wish to employ it to make your decision. You start by collecting data on prior student performance in HW1 and HW2, along with their final letter grades. You then create a binary label (1/0) based on the final grades such that you assign a label of 1 if the final grade is at least a B and 0 otherwise. Next, you train the model on this data set and calculate the training error. The distance measure you use is **Euclidean distance**.

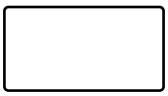
Note: For ease of computation, we will use only 10 randomly selected training data points as plotted below. Label 1 is represented by '+' and green color; label 0 is represented by '-' and red color.



(a) **Numerical answer:** What will be the training error if you choose $k = 1$?

0

(b) **Numerical answer:** What will be the training error if you choose $k = 3$?



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

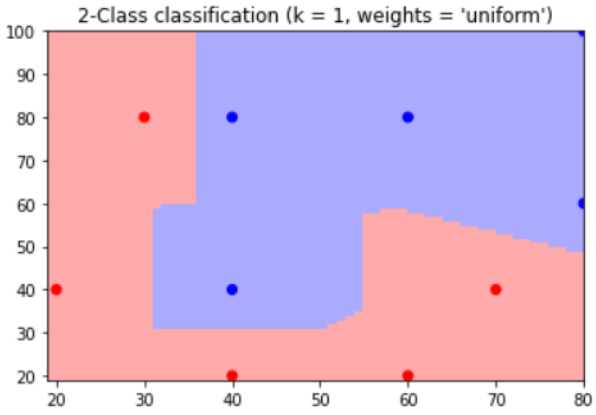
(c) **True or False:** Using Euclidean distance as the distance measure, the decision boundary of k-NN for $k = 1$ is a piece-wise straight line, that is, it contains only straight line segments. **Justify your answer.**

◯ 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 (1/0) for $k = 1$?

0

ii. **Numerical answer:** What would be the predicted class (1/0) for $k = 3$?

0

(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.

6. **Select all that apply:** Please select all that apply about k-NN 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 be applied to classification problems but not regression problems.
   - ☐ We can always achieve zero training error (perfect classification) with k-NN, but it may not generalize well in testing.

   True: A, runtime becomes large; D, by setting k = 1
   False: B, we increase k to avoid overfitting; C, KNN regression

3 Model Selection and Errors

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

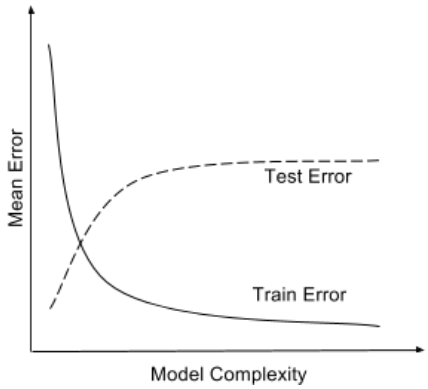
   (b) **Select all that apply:** Which of the following are expected to help?

   - ☐ Increasing the training data size.
   - ☐ Decreasing the training data size.
   - ☐ Increasing model complexity (For example, if your classifier is an SVM, use a more complex kernel. Or if it 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

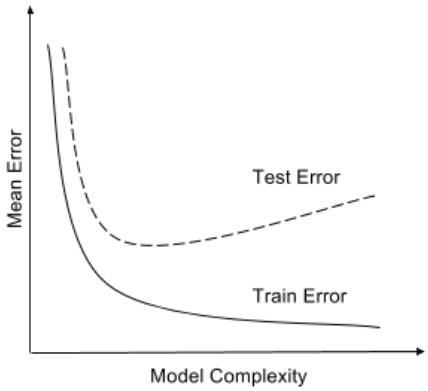
   a and d

   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)

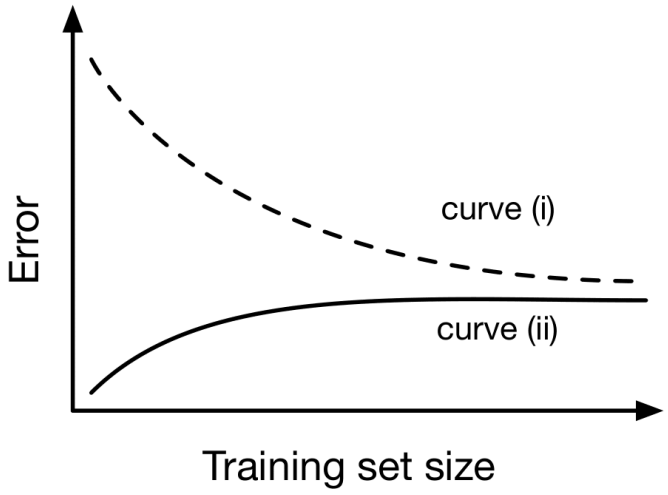
◯ Plot A

◯ 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.

2. **Training Sample Size:** In this problem, we will consider the effect of training dataset size n on a logistic regression classifier with d 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 trends in training and testing error as we increase the sample size $n = |S|$.



   (a) **Short Answer:** Which curve represents the training error? Provide 1-2 sentences of justification.

   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.

   (b) **Short Answer:** In one word, what does the gap between the two curves represent?

   overfitting

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

   (a) Increasing decision tree max depth.

   ◯ Less likely to overfit

◯ More likely to overfit

More likely to overfit

(b) Increasing decision tree mutual information split threshold.

◯ Less likely to overfit

◯ More likely to overfit

Less likely to overfit

(c) Increasing decision tree max number of nodes.

◯ Less likely to overfit

◯ More likely to overfit

More 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 randomly 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 randomly from the true data distribution.

◯ Less likely to overfit

◯ More likely to overfit

Less likely to overfit

4. 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.

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

(b) **Short answer:** In one sentence, give one advantage of increasing the value of α.

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

4 Perceptron

1. **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, 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.

2. **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. Refer mistake bound concept.

3. **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. **True or False:** If the training data is linearly separable and representative of the true distribution, the perceptron algorithm always finds the optimal decision boundary for the true distribution.

   - ◯ True

◯ False

False.

5. (1 point) **True or False:** Consider two datasets $\mathcal{D}^{(1)}$ and $\mathcal{D}^{(2)}$, where $\mathcal{D}^{(1)} = \{(x_1^{(1)}, y_1^{(1)}), ..., (x_n^{(1)}, y_n^{(1)})\}$ and $\mathcal{D}^{(2)} = \{(x_1^{(2)}, y_1^{(2)}), ..., (x_m^{(2)}, y_m^{(2)})\}$ such that $x_i^{(1)} \in \mathbb{R}^{d_1}$, $x_i^{(2)} \in \mathbb{R}^{d_2}$, $d_1 > d_2$ and $n > m$. The maximum number of mistakes the perceptron algorithm will make is always higher for dataset $D^{(1)}$ than it is for dataset $D^{(2)}$.

   ◯ True

   ◯ False

   False.

6. 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.

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

   [1, −2]

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

   −1

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

   [1, −2]

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

-1

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

$[1, -2]$

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

-1

vii. **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.

7. **True or False:** Data (X, Y) has a non-linear decision boundary. Fortunately, there is a function $\mathcal{F}$ that maps (X, Y) to $(\mathcal{F}(X), Y)$ such that $(\mathcal{F}(X), Y)$ is linearly separable. We have tried to build a modified perceptron to classify (X, Y). Is the given (modified) perceptron update rule correct?

if sign$(w\mathcal{F}(x^{(i)}) + b)$!= $y^{(i)}$:
$w' = w + y^{(i)}\mathcal{F}(x^{(i)})$
$b' = b + y^{(i)}$

◯ True

◯ False

True

8. (a) **True or False:** All *examples* $(\mathbf{x}, y)$ that the perceptron algorithm has seen are weighted equally.

◯ True

◯ False

False. Only mistakes affect perceptron weights

(b) **True or False:** All *mistakes* the perceptron algorithm has made are weighted equally.

- ◯ True
- ◯ False

False. The most recent mistakes are more heavily weighted

5 Linear Regression

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

   ◯ X is 35×6, y is 35×1, θ is 6×1

   ◯ X is 35×6, y is 35×6, θ is 6×6

   ◯ X is 35×5, y is 35×1, θ is 5×1

   ◯ X is 35×5, y is 35×5, θ is 5×5

   A.

2. Answer the following True/False questions, providing brief explanations to support your answers.

   (a) **True or False:** Consider a linear regression model with only one parameter, the bias, i.e., $y = \beta_0$. 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

   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.

   (b) **True or False:** Given data $\mathcal{D} = \{(x_1, y_1), ..., (x_n, y_n)\}$, 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 $\mathcal{D}$.

   Consider a new dataset $\mathcal{D}_{\text{new}}$ 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 $\mathcal{D}_{\text{new}}$ is equal to $\hat{w}$.

   ◯ True

   ◯ False

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.

3. **Select all that apply:** 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.

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

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

4. **Numerical answer:** 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 objective function, 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 $y^{(2)}$.

   $x_1^{(3)}$

   [ ]

   $x_2^{(3)}$

   [ ]

   $y^{(3)}$

   [ ]

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

   Select one: 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

   No

5. 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:

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

Consider the dataset S plotted in Fig. 3 along with its associated regression line. For each of the altered data sets S^{new} plotted in Fig. 5, indicate which regression line (relative to the original one) in Fig. 4 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					

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

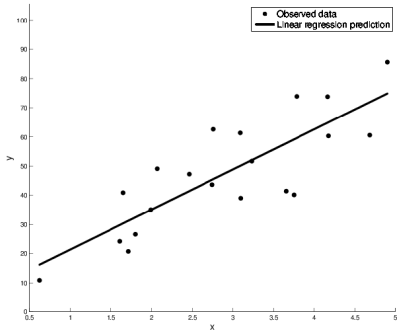
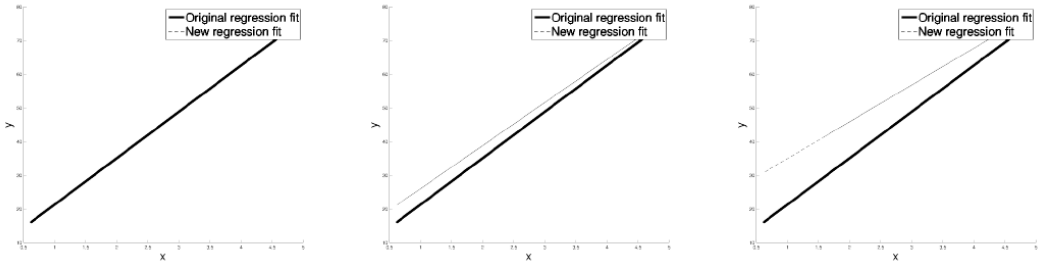
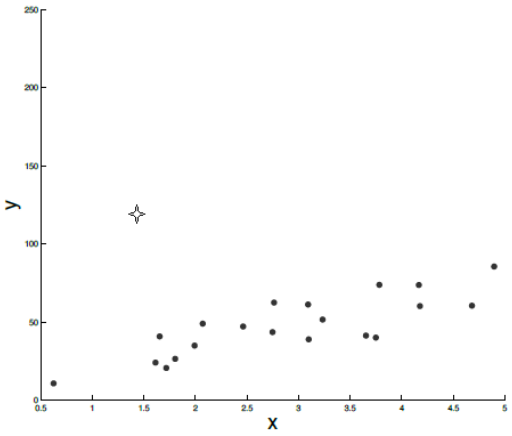


Figure 3: An observed data set and its associated regression line.

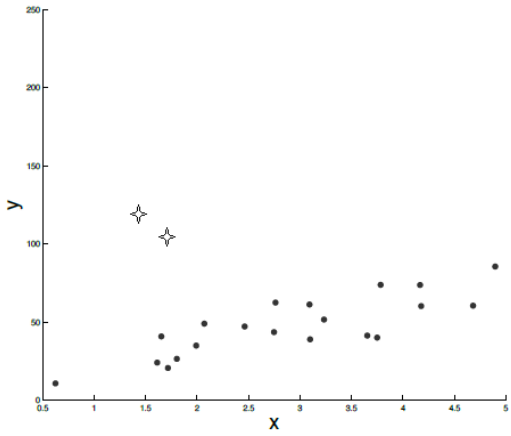


(a) Old and new regression lines. (b) Old and new regression lines. (c) Old and new regression lines.

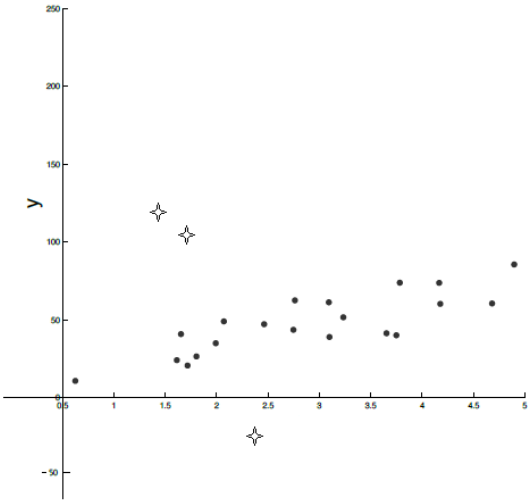
Figure 4: New regression lines for altered data sets S^{new}.



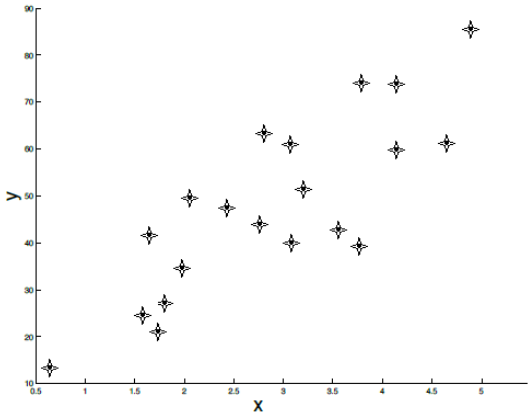
(a) Adding one outlier to the original data set.



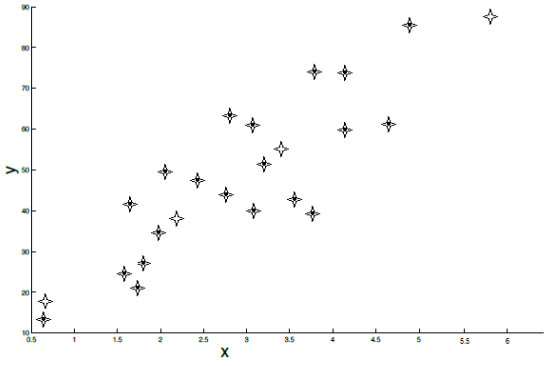
(b) Adding two outliers to the original data set.



(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 5: New data set S^{new}.

6 Optimization

1. **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.
   - ☐ 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.

2. (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:

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

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

   (b) **Select all that apply:** 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?

   - ☐ $\alpha = 0$
   - ☐ $\alpha = \frac{1}{2}$
   - ☐ $\alpha = 1$
   - ☐ $\alpha = 2$

☐ None of the above

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

(c) **Numerical answer:** 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 $z^{(0)} = 1$.

$(0, 1)$.

7 MLE/MAP

1. For the following questions, answer True or False and provide a brief justification of your answer.

   1. **True or False:** Consider the linear regression model $y = w^T x + \epsilon$. Assuming $\epsilon \sim \mathcal{N}(0, \sigma^2)$ and maximizing the conditional log-likelihood is equivalent to minimizing the sum of squared errors $\|y - w^T x\|_2^2$.

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

   2. **True or False:** 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:

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

      .

      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

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

2. **Math:** Let $X_1, X_2, ..., X_N$ be data drawn independently from a uniform distribution over a diamond-shaped area with edge length $\sqrt{2}\theta$ in $\mathbb{R}^2$, where $\theta \in \mathbb{R}^+$ (see Figure 6). Thus, $X_i \in \mathbb{R}^2$ and the distribution is

   $$p(x|\theta) = \begin{cases} \frac{1}{2\theta^2} & \text{if } \|x\| \leq \theta \\ 0 & \text{otherwise} \end{cases}$$

where $\|x\| = |x_1| + |x_2|$ is the L_1 norm. Find the maximum likelihood estimate of θ.

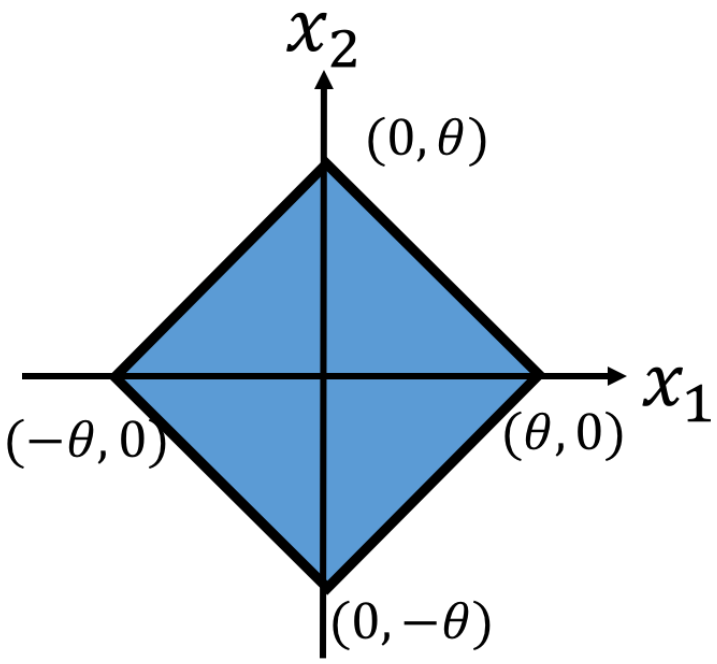


Figure 6: Area of $\|x\| \leq \theta$

Analysis:

The likelihood function is

$$L(X_1, X_2, ..., X_N; \theta) = \frac{1}{(2\theta^2)^N} 1\left\{ \max_{1 \leq i \leq N} \|X_i\| \leq \theta \right\}$$

To maximize likelihood, we want θ to be as small as possible with the constraint of

$\max_{1 \le i \le N} \|X_i\| \le \theta$, otherwise the likelihood drops to 0. So the MLE of θ is

$$\widehat{\theta} = \max_{1 \le i \le N} \|X_i\|$$

3. **Math:** Suppose we want to model a 1-dimensional dataset of N real valued features $\left(x^{(i)}\right)$ and targets $\left(y^{(i)}\right)$ by:

$$y^{(i)} \sim \mathcal{N}\left(\exp(wx^{(i)}), 1\right),$$

where w is our unknown (scalar) parameter and $\mathcal{N}$ is the normal distribution with probability density function:

$$f(a)_{\mathcal{N}(\mu,\sigma^2)} = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(a-\mu)^2}{2\sigma^2}\right)$$

Can the maximum conditional negative log likelihood estimator of w be solved analytically? If so, find the expression for w_{MLE}. If not, say so and write down the update rule for w using gradient descent.

Cannot be found analytically.

Taking the derivative of the negative log likelihood with respect to w yields:

$$\frac{\partial \text{NLL}}{\partial w} = \sum_i^N -x^{(i)}y^{(i)} \exp(wx^{(i)}) + x^{(i)} \exp(2wx^{(i)}) = 0$$

Taking log on both sides yields:

$$\sum_i^N -log(x^{(i)}) - log(y^{(i)}) - (wx^{(i)}) + log(x^{(i)}) + (2wx^{(i)}) = 0$$

$$w = \sum_i^N log(y^{(i)})/x^{(i)}$$

Update rule is thus

$$w \leftarrow w - \eta \frac{\partial \text{NLL}}{\partial w}$$

4. Assume we have n random variables $x_i, i \in [1, n]$, each drawn independently from a Normal distribution with mean μ and variance σ^2.

$$p(x_1, x_2, ..., x_n | \mu, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-(x_i - \mu)^2}{2\sigma^2}\right)$$

a) Write the log-likelihood function $\ell(x_1, x_2 ... x_n | \mu, \sigma^2)$.

$$\log(\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\{\frac{-(x_i - \mu)^2}{2\sigma^2}\}) = \sum_{i=1}^{n} [\log(\frac{1}{\sqrt{2\pi}\sigma}) - \frac{(x_i - \mu)^2}{2\sigma^2}] \quad (1)$$

$$= -n \log(\sqrt{2\pi}\sigma) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (x_i - \mu)^2 \quad (2)$$

b) Derive an expression for the Maximum Likelihood Estimate for the variance (σ^2).

We can find the estimator by solving $\nabla_\sigma l(x_1, x_2 ... x_n | \mu, \sigma^2) = 0$.

$$-n\frac{1}{\sqrt{2\pi}\sigma}\sqrt{2\pi} + \frac{1}{\sigma^3}\sum_{i=1}^{n}(x_i - \mu)^2 = 0 \tag{3}$$

$$\frac{1}{\sigma^3}\sum_{i=1}^{n}(x_i - \mu)^2 = \frac{n}{\sigma} \tag{4}$$

$$\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2 = \sigma^2 \tag{5}$$

5. Assume we have n random variables $x_i, i \in [1, n]$, each drawn independently from a Bernoulli distribution with mean θ. Recall that in a Bernoulli distribution $X \in \{0, 1\}$ and the pdf is:

$$p(X|\theta) = \theta^x(1-\theta)^{1-x}$$

a) Derive the likelihood $L(\theta | X_1, \ldots, X_n)$.

$L(\theta; X_1, \ldots, X_n) = \prod_{i=1}^{n} p(X_i; \theta)$
$L(\theta; X_1, \ldots, X_n) = \prod_{i=1}^{n} \theta^{x_i}(1-\theta)^{1-x_i}$
$L(\theta; X_1, \ldots, X_n) = \theta^{\sum_{i=1}^{n} x_i}(1-\theta)^{n-\sum_{i=1}^{n} x_i}$
Either of the final two steps are acceptable.

b) Show that the log-likelihood is:

$$l(\theta|X_1,\ldots,X_n) = \left(\sum_{i=1}^{n} X_i\right)\log(\theta) + \left(n - \sum_{i=1}^{n} X_i\right)\log(1-\theta)$$

$l(\theta; X_1,\ldots,X_n) = \log L(\theta; X_1,\ldots,X_n)$
$l(\theta; X_1,\ldots,X_n) = \log\left[\theta^{\sum_{i=1}^n x_i}(1-\theta)^{n-\sum_{i=1}^n x_i}\right]$
$l(\theta; X_1,\ldots,X_n) = (\sum_{i=1}^n x_i)\log(\theta) + (n - \sum_{i=1}^n x_i)\log(1-\theta)$

c) Show that the MLE is $\hat{\theta} = \frac{1}{n}\sum_{i=1}^{n} X_i$.

Take the derivative of the log likelihood and set it to zero
$\frac{dl}{d\theta} = \frac{d}{d\theta}\left[(\sum_{i=1}^n x_i)\log(\theta) + (n - \sum_{i=1}^n x_i)\log(1-\theta)\right] = 0$
$\frac{\sum_{i=1}^n x_i}{\theta} - \frac{n - \sum_{i=1}^n x_i}{1-\theta} = 0$
$\left(\sum_{i=1}^n x_i\right)(1-\theta) - \left(n - \sum_{i=1}^n x_i\right)\theta = 0$
$\sum_{i=1}^n x_i - n\theta = 0$
$\hat{\theta} = \frac{1}{n}(\sum_{i=1}^n X_i)$

6. 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 $x \geq 0$.

i. (2 points) 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.

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

ii. (2 points) Report the maximum likelihood estimator of the scale parameter, σ, for a single image's noise vector $\mathbf{x}$.

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

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

8 Logistic Regression and Regularization

1. A generalization of logistic regression to a multiclass settings involves expressing the per-class probabilities $P(y = c|x)$ 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 (labels 0 or 1). Rewrite the above expression for this situation to end up with expressions for $P(Y = 1|x)$ and $P(Y = 0|x)$ that we have already come across 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, $1 - p = \frac{1}{1+\exp(w^T x)}$

2. Considering a Gaussian prior, write out the MAP objective function $J(w)_{MAP}$ in terms of the MLE objective $J(w)_{MLE}$. Name the variant of logistic regression this results in.

   $J_{MAP}(\mathbf{w}) = J_{MLE}(\mathbf{w}) - \lambda\|\mathbf{w}\|_2^2$. This is L2 regularized logistic regression.

3. Given a training set $\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}, \ldots, (\mathbf{x}^{(N)}, y^{(N)})\}$ 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 $\hat{w}$ that maximize the likelihood for the training set, assuming a parametric model of the form

$$p(y = 1|x; w) = \frac{1}{1 + \exp(-w^T x)}.$$

The conditional log likelihood of the training set is

$$\ell(w) = \sum_{i=1}^{n} y_i \log p(y_i, |x_i; w) + (1 - y_i) \log(1 - p(y_i, |x_i; w)),$$

and the gradient is

$$\nabla \ell(w) = \sum_{i=1}^{n} (y_i - p(y_i|x_i; w))x_i.$$

a) Is it possible to get a closed form for the parameters $\hat{w}$ that maximize the conditional log likelihood? How would you compute $\hat{w}$ 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) For a binary logistic regression model, we predict $y = 1$ when $p(y = 1|x) \geq 0.5$. Show that this is a linear classifier.

Using the parametric form for $p(y = 1|x)$:

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

so we predict $\hat{y} = 1$ if $w^T x \geq 0$.

c) Consider the case with binary features, i.e, $x \in \{0, 1\}^d$, where feature x_1 is rare and happens to appear in the training set with only label 1. What is $\hat{w}_1$? Is the gradient ever zero for any finite w? Why is it important to include a regularization term to control the norm of $\hat{w}$?

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 is because, when this feature is observed in the training set, we will want to predict predict 1 irrespective of everything else. This is an undesired behaviour 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 spurious co-occurrence. Controlling the norm of the weight vector will prevent these pathological cases.

4. Given the following dataset, $\mathcal{D}$, and a fixed parameter vector, $\boldsymbol{\theta}$, write an expression for the binary logistic regression conditional likelihood.

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

- Write your answer in terms of $\boldsymbol{\theta}$, $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$, $\mathbf{x}^{(3)}$, and $\mathbf{x}^{(4)}$.

- Do not include $y^{(1)}$, $y^{(2)}$, $y^{(3)}$, or $y^{(4)}$ in your answer.
- Don't try to simplify your expression.

Conditional likelihood:

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

5. Write an expression for the decision boundary of binary logistic regression with a bias term for two-dimensional input features $x_1 \in \mathbb{R}$ and $x_2 \in \mathbb{R}$ and parameters b (the intercept parameter), w_1, and w_2. Assume that the decision boundary occurs when $P(Y = 1 \mid \mathbf{x}, b, w_1, w_2) = P(Y = 0 \mid \mathbf{x}, b, w_1, w_2)$.

   (a) Write your answer in terms of x_1, x_2, b, w_1, and w_2.

   Decision boundary equation:

   $0 = b + w_1 x_1 + w_2 x_2$

   (b) What is the geometric shape defined by this equation?

   A line.

6. We have now feature engineered the 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. **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 1-2 concise sentences.

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

9 Feature Engineering and Regularization

1. **Model Complexity:** 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 $c^* : \mathbb{R} \to \{\pm 1\}$, and suppose we have a random sample S of size n drawn iid 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. 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? **Please explain 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 cvv for any given sample S.

   b) Briefly **explain in 1 to 2 sentences** why the true error first drops and then increases as we increase the degree d. 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.

10 Neural Networks

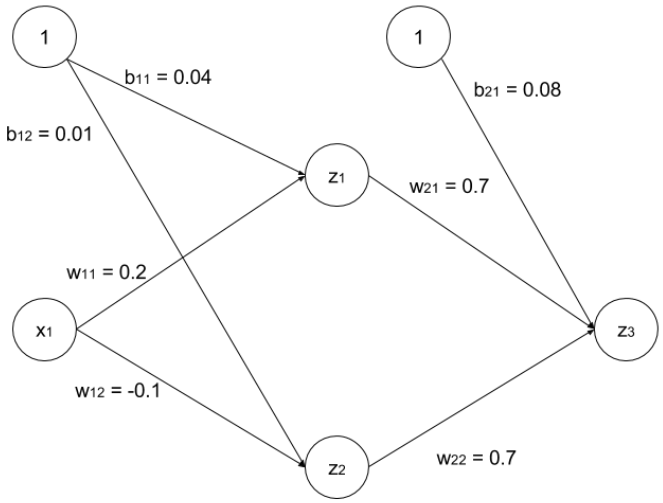


Figure 7: neural network

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

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

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

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

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

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

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

   (i) For $x_1 = 0.3$, compute z_3 in terms of e.

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

   (ii) Which class does the network predict for the data point ($x_1 = 0.3$)? Note that $\hat{y} = 1$ if $z_3 > \frac{1}{2}$, else $\hat{y} = 0$.

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

(iii) 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}, $\frac{\partial L}{\partial b_{21}}$, 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 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.

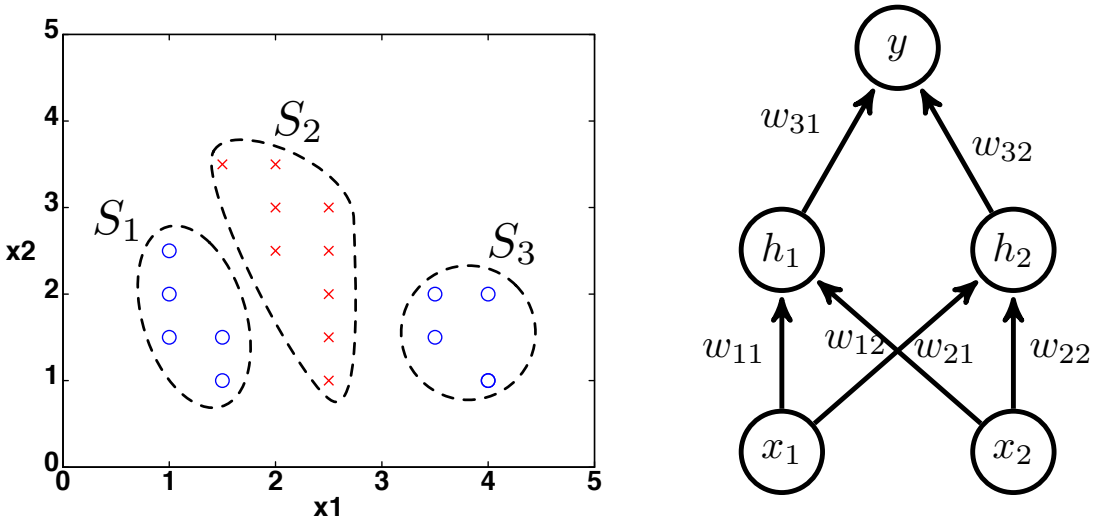
$\frac{\partial L}{\partial b_{21}} = \frac{\partial L}{\partial z_3} \frac{\partial z_3}{\partial a_3} \frac{\partial a_3}{\partial b_{21}}$

(iv) 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}, $\frac{\partial L}{\partial b_{12}}$, 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 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.

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

2. 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 8a. 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 8a) 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 8b in order to correctly classify this training set. For all nodes, we will use the threshold activation function

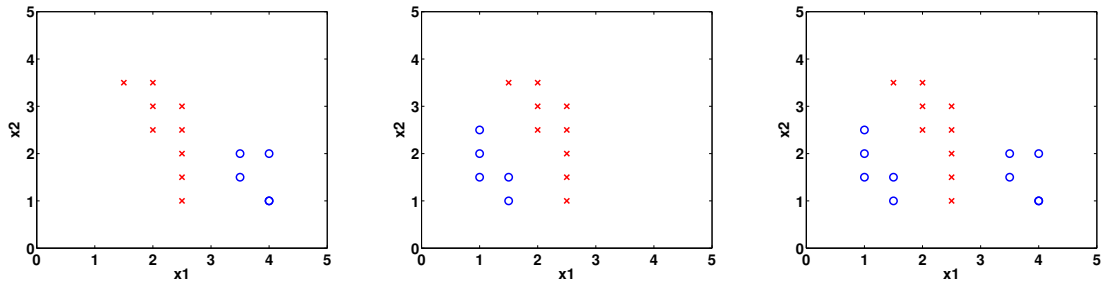
$$\phi(z) = \begin{cases} 1 & z > 0 \\ 0 & z \leq 0. \end{cases}$$



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

(b) The neural network architecture

Figure 8



(a) Set S2 and S3

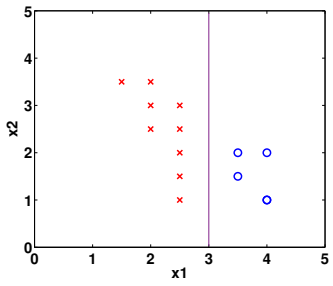
(b) Set S1 and S2

(c) Set S1, S2 and S3

Figure 9: NN classification.

(i) 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.

(a) On Fig 9a, draw a linear decision boundary that separates S_2 and S_3.

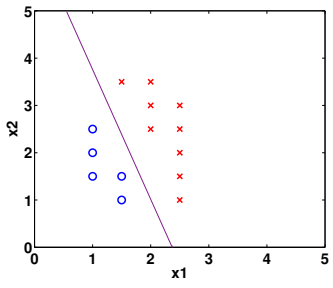


(b) Write down the corresponding weights w_{11}, w_{12}, and b_1 so that $h_1(x) = 0$ for all points in S_3 and $h_1(x) = 1$ for all points in S_2. One solution suffices and the same applies to (ii) and (iii).

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

(ii) Next we 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.

(a) On Fig 9b, draw a linear decision boundary that separates S_1 and S_2.



(b) Write down the corresponding weights w_{21}, w_{22}, and b_2 so that $h_2(x) = 0$ for all points in S_1 and $h_2(x) = 1$ for all points in S_2.

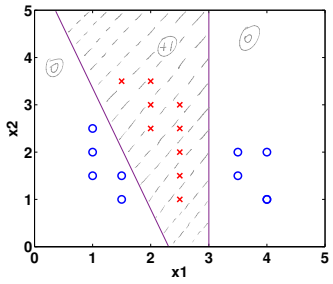
$w_{21} = 3, w_{22} = 1, b_2 = -7$

(iii) 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 $h_3(x) = \phi\big(w_{31}h_1(x) + w_{32}h_2(x) + b_3\big)$.

(a) Compute w_{31}, w_{32}, b_3 such that $h_3(x)$ correctly classifies the entire data set.

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

(b) Draw your decision boundary in Fig 9c.



3. One part of learning parameters in a neural network is getting the gradients of the parameters.

Suppose we have a dataset $\mathcal{D}$ with N data points x_i with label y_i, where $i \in [1, N]$. x_i is a $d \times 1$ vector and $y_i \in \{0, 1\}$. We use the data to train a neural network with one hidden layer:

$$\begin{aligned} h(x) =& \sigma(W_1 x + b_1) \\ p(x) =& \sigma(W_2 h(x) + b_2), \end{aligned}$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$ is the sigmoid function, W_1 is a n by d matrix, b_1 is a n by 1 vector, W_2 is a 1 by n matrix, and b_1 is a 1 by 1 vector.

We use cross entropy loss and minimize the negative log likelihood to train the neural network:

$$\ell_{\mathcal{D}}(W) = \frac{1}{N}\sum_{i=1}^{N} \ell_i(W) = \frac{1}{m}\sum_{i=1}^{N} -(y_i \log p_i + (1 - y_i)\log(1 - p_i)),$$

where $p_i = p(x_i), h_i = h(x_i)$.

(a) Describe how you would derive the gradients w.r.t the parameters W_1, W_2 and b_1, b_2. You do not need to write out the actual mathematical expression.

Use the chain rule.

(b) When N is large, we typically use a small subset of the dataset to estimate the gradient — stochastic gradient descent (SGD). Explain why we use SGD instead of gradient descent.

SGD converges faster than gradient descent.

(c) Derive expressions for the following gradients: $\frac{\partial l}{\partial p_i}, \frac{\partial l}{\partial W_2}, \frac{\partial l}{\partial b_2}, \frac{\partial l}{\partial h_i}, \frac{\partial l}{\partial W_1}, \frac{\partial l}{\partial b_1}$. When deriving the gradient w.r.t. the parameters in lower layers, you may assume the gradient in upper layers are available to you (i.e., you can use them in your equation). For example, when calculating $\frac{\partial l}{\partial W_1}$, you can assume $\frac{\partial l}{\partial p_i}, \frac{\partial l}{\partial W_2}, \frac{\partial l}{\partial b_2}, \frac{\partial l}{\partial h_i}$ are known.

$$\frac{\partial l}{\partial p_i} = \frac{1}{m}(-\frac{y_i}{p_i} + \frac{1-y_i}{1-p_i})$$
$$\frac{\partial l}{\partial W_2} = \frac{1}{m}\sum_i \frac{\partial l_i}{\partial p_i}\frac{\partial p_i}{\partial W_2} = \frac{1}{m}\sum_i \frac{\partial l_i}{\partial p_i} p_i(1-p_i)h_i^T$$
$$\frac{\partial l}{\partial b_2} = \frac{1}{m}\sum_i \frac{\partial l_i}{\partial p_i} p_i(1-p_i)$$
$$\frac{\partial l}{\partial h_i} = \frac{\partial p_i}{\partial h_i}\frac{\partial l}{\partial p_i} = W_2^T p_i(1-p_i)\frac{\partial l}{\partial p_i}$$
$$\frac{\partial l}{\partial W_1} = \frac{1}{m}\sum_i \frac{\partial l_i}{\partial h_i}\frac{\partial h_i}{\partial W_1} = \frac{1}{m}\sum_i \left[\frac{\partial l_i}{\partial h_i} \circ h_i \circ (1-h_i)\right] x_i^T$$
$$\frac{\partial l}{\partial b_1} = \frac{1}{m}\sum_i \frac{\partial l_i}{\partial h_i}\frac{\partial h_i}{\partial b_1} = \frac{1}{m}\sum_i \frac{\partial l_i}{\partial h_i} \circ h_i \circ (1-h_i)$$

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

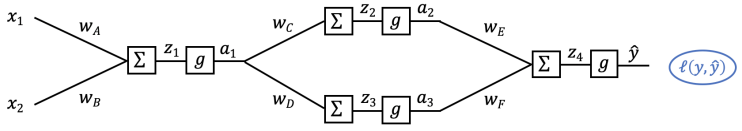


Figure 13: Neural Network

- All g functions 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: $\frac{\partial \ell}{\partial w_E} = \frac{\partial ?}{\partial ?}\frac{\partial ?}{\partial ?}\ldots$ 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

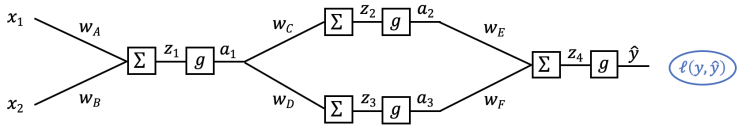


Figure 14: Neural Network

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

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

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: $w_C \leftarrow \ldots$

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

5. Backpropagation in neural networks can lead to slow or unstable learning because of the vanishing or exploding gradients problem. Understandably, Neural the Narwhal does not believe this. To convince Neural, Lamar Jackson uses the example of an N layer neural network that takes in a scalar input x, and where each layer consists of a single neuron. More formally, $x = o_0$, and for each layer $i \in \{1, 2, ..., N\}$, we have

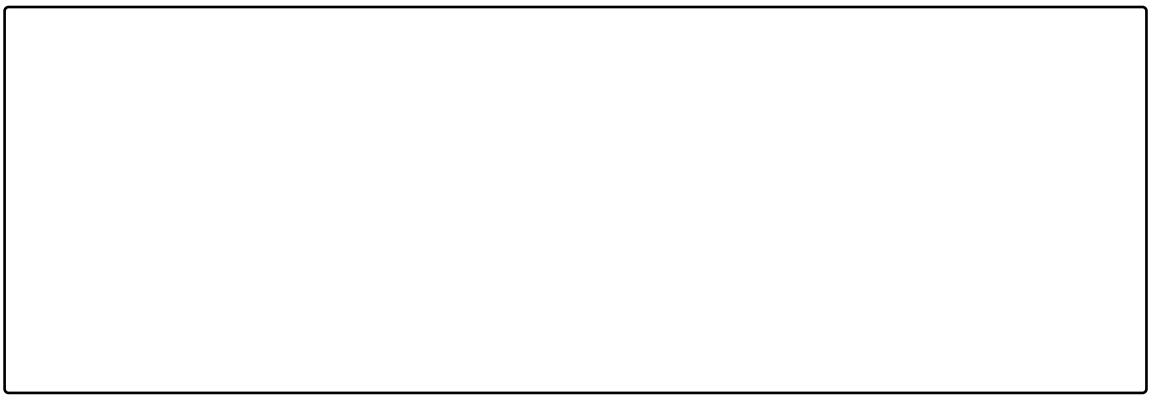
$$s_i = w_i o_{i-1} + b_i$$
$$o_i = \sigma(s_i)$$

where σ is the sigmoid activation function. Note that w_i, b_i, o_i, s_i are all scalars.

i. (1 point) Give an expression for $\frac{\partial o_N}{\partial w_1}$. Your expression should be in terms of the s_i's, the w_i's, N, x_i, and $\sigma'(\cdot)$, the derivative of the sigmoid function.

$$\begin{aligned}\frac{\partial o_N}{\partial w_1} &= \frac{\partial o_N}{\partial o_{N-1}} \frac{\partial o_{N-1}}{\partial o_{N-2}} \cdots \frac{\partial o_1}{\partial w_1} \\ &= \frac{\partial o_1}{\partial w_1} \prod_{i=2}^{N} \frac{\partial o_i}{\partial o_{i-1}} \\ &= \sigma'(s_1) x \prod_{i=2}^{N} \sigma'(s_i) w_i\end{aligned}$$

ii. (1 point) Knowing that $\sigma'(\cdot)$ is at most $\frac{1}{4}$ and supposing that all the weights are 1 (i.e. $w_i = 1$ for all i), give an upper bound for $\frac{\partial o_N}{\partial w_1}$. Your answer should be in terms of x and N.



$\frac{\partial o_N}{\partial w_1} \leq x \left(\frac{1}{4}\right)^N$