

Solutions

10-601 Machine Learning
Spring 2020
Exam 1 Practice Problems
February 12, 2020
Time Limit: N/A

Name:
Andrew Email:
Room:
Seat:
Exam Number:

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!

Topic	Pages	# Questions
Decision Trees	3	6
KNN	6	5
Model Selection	8	4
Perceptron	11	8
Linear Models	14	7
Optimization	19	4

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?

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

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

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

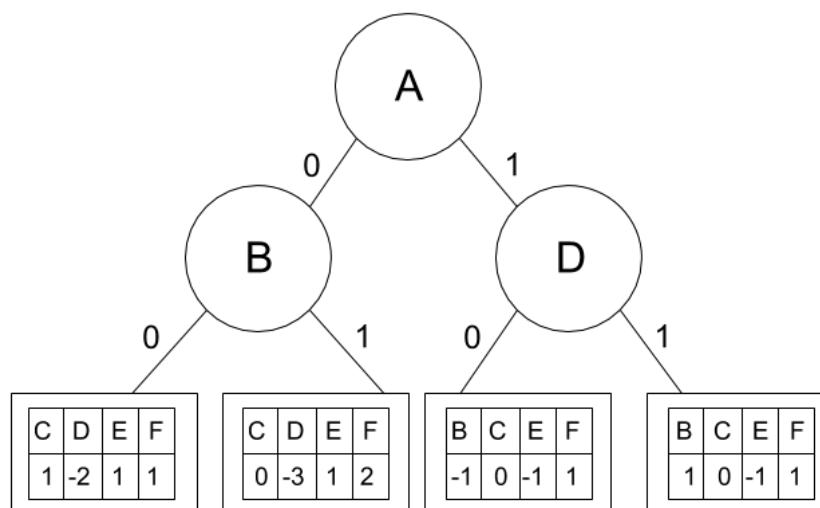


Figure 1: Perceptron Tree of max depth=2

- (a) **Numerical answer:** Given a sample $\mathbf{x} = [1, 1, 0, 1, 0, 1]$, predict the output label for this sample

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 = $\text{sign}(3) = 1$.

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

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

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

3. (2 points) ID3 algorithm is a greedy algorithm for growing Decision Tree and it suffers the same problem as any other greedy algorithm that finds only locally optimal trees. Which of the following method(s) can make ID3 “less greedy”? **Select all that apply:**

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

only the top 2 with the best performance (e.g., top 2 with lowest training errors at the current step)

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

4. [2 pts] ID3 algorithm is guaranteed to find the optimal solution for decision tree.

Circle one: True False

False.

5. [2 pts] One advantages of decision trees algorithm is that they are not easy to overfit comparing to naive Bayes.

Circle one: True False

False.

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

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

☐ True

☐ False

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

2 K Nearest Neighbors

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

Assume the data points x_1, x_2, \dots, x_n are sorted in the ascending order, we use Euclidean distance as the distance metric, and a point can be its own neighbor. True or False: We **CAN** build a decision tree (with decisions at each node has the form " $x \geq t$ " and " $x < t$ ", for $t \in \mathbb{R}$) that behave exactly the same as the 1-nearest neighbor classifier, on this dataset.

- ☐ True
☐ False

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

2. **Select all that apply:** Please select all that apply about kNN in the following options:

Assume a point can be its own neighbor.

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

True: A, Curse of dimensionality; D, by setting $k = 1$

False: B, we increase k to avoid overfitting; C, KNN regression

3. (1 point) **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

4. (1 point) **Select one.** Imagine you are using a k -Nearest Neighbor classifier on a data set with lots of noise. You want your classifier to be *less* sensitive to the noise. Which is more likely to help and with what side-effect?
- ☐ Increase the value of $k \Rightarrow$ 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

5. (1 point) **Select all that apply:** Identify the correct relationship between bias, variance, and the hyperparameter k in the k -Nearest Neighbors algorithm:
- ☐ Increasing k leads to increase in bias
 - ☐ Decreasing k leads to increase in bias
 - ☐ Increasing k leads to increase in variance
 - ☐ Decreasing k leads to increase in variance

A and D

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 till convergence on some training data $\mathcal{D}^{\text{train}}$, and tested on a separate test set $\mathcal{D}^{\text{test}}$. You look at the test error, and find that it is very high. You then compute the training error and find that it is close to 0.

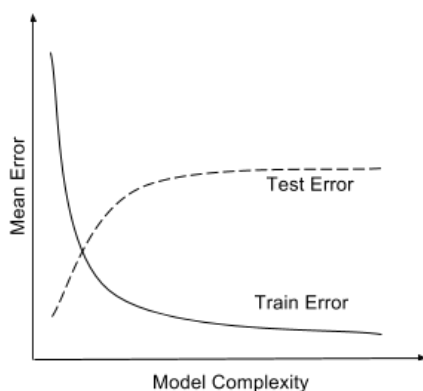
1. [4 pts] Which of the following is expected to help? Select all that apply.
 - (a) Increase the training data size.
 - (b) Decrease the training data size.
 - (c) Increase model complexity (For example, if your classifier is an SVM, use a more complex kernel. Or if it is a decision tree, increase the depth).
 - (d) Decrease model complexity.
 - (e) Train on a combination of $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$ and test on $\mathcal{D}^{\text{test}}$
 - (f) Conclude that Machine Learning does not work.

a and d

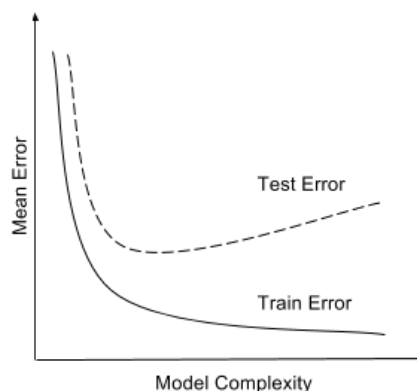
2. [5 pts] Explain your choices. The model is overfitting. In order to address the problem, we can either increase training data size or decrease model complexity. We should never do (e), the model shouldn't see any testing data in the training process.

3. [2 pts] What is this scenario called? Overfitting

4. [1 pts] 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)

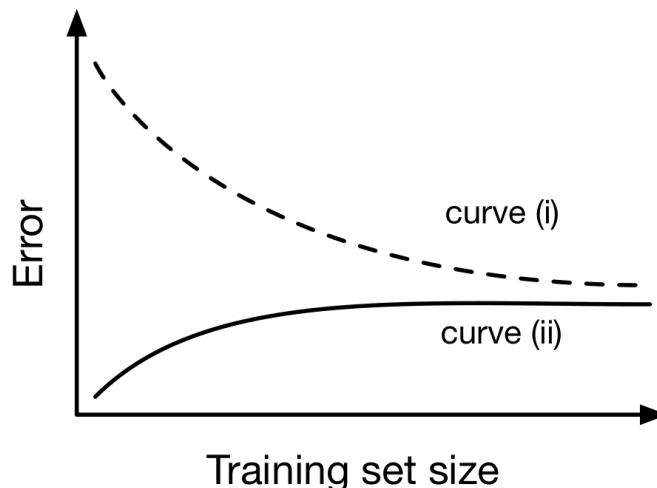
B. When model complexity increases, model can overfit better, so training error will decrease. But when it overfits too much, testing error will increase.

2. **True and Sample Errors:** Consider a classification problem with distribution D and target function $c^* : \mathcal{R}^d \mapsto \pm 1$. For any sample S drawn from D , answer whether the following statements are true or false, along with a brief explanation.

1. [4 pts] For a given hypothesis space H , it is possible to define a sufficient size of S such that the true error is bounded by the sample error by a margin ϵ , for all hypotheses $h \in H$ with a given probability. **True. According to uniform convergence theorem, we can get such bound.**
2. [4 pts] The true error of any hypothesis h is an upper bound on its training error on the sample S . **False. We said true error is close to training error, but it might be smaller than training error, so it is not an upper bound.**

3. **Training Sample Size:** In this problem, we will consider the effect of training sample 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 trend for how the training and testing error change as we increase the sample size $n = |S|$. Your task in this question is to analyze this plot and identify which curve corresponds to the training and test error. Specifically:



1. Which curve represents the training error? **Please 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.

2. In one word, what does the gap between the two curves represent? **Overfitting**

4. **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 $c^* : \mathbb{R} \rightarrow \{\pm 1\}$ 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 $\phi_d(x) = (1, x, x^2, \dots, x^d)$ that maps points on the real line to $(d + 1)$ -dimensional space.

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

1. [4 pts.] For a given dataset S , is it possible for the training error to increase when we increase the degree d of the feature map? **Please explain your answer in 1 to 2 sentences.** No. Every linear separator using the feature map ϕ_d can also be expressed using the feature map ϕ_{d+1} , since we are only adding new features. It follows that the training error must decrease for any given sample S .
2. [4 pts.] 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.

4 Perceptron

1. **Select all that apply:** Let $S = \{(\mathbf{x}^{(1)}, y^{(1)}), \dots, (\mathbf{x}^{(n)}, y^{(n)})\}$ be n linearly separable points by a separator through the origin in \mathbb{R}^d . Let S' be generated from S as: $S' = \{(c\mathbf{x}^{(1)}, y^{(1)}), \dots, (c\mathbf{x}^{(n)}, y^{(n)})\}$, 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' .

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:** We know that if the samples are linearly separable, the perceptron algorithm finds a separating hyperplane in a finite number of steps. Given such a dataset with linearly separable samples, select whether the following statement is True or False: The running 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 proof can be found on slide 34 of http://www.cs.cmu.edu/~10701/slides/8_Perceptron.pdf

3. (1 point) **Select all that apply:** Which of the following are considered as inductive bias of perceptron.

- ☐ Assume that most of the cases in a small neighborhood in feature space belong to the same class
- ☐ Decision boundary should be linear
- ☐ Prefer to correct the most recent mistakes
- ☐ Prefer the smallest hypothesis that explains the data

BC

4. (1 point) **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 $D^{(1)}$ and $D^{(2)}$ where $D^{(1)} = \{(x_1^{(1)}, y_1^{(1)}), \dots, (x_n^{(1)}, y_n^{(1)})\}$ and $D^{(2)} = \{(x_1^{(2)}, y_1^{(2)}), \dots, (x_m^{(2)}, y_m^{(2)})\}$ such that $x_i^{(1)} \in \mathbb{R}^{d_1}$, $x_i^{(2)} \in \mathbb{R}^{d_2}$. Suppose $d_1 > d_2$ and $n > m$. Then the maximum number of mistakes a perceptron algorithm will make is higher on dataset $D^{(1)}$ than on dataset $D^{(2)}$.

☐ True

☐ False

False.

4.1 Perceptron Calculation

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 $\theta^T = [0, 0]$, bias $b = 0$ and that we pass all of our examples through in order of their example number.

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

$[1, -2]$

2. (1 point) **Numerical answer:** What would be the updated bias b be after we pass example 1 through our the Perceptron algorithm?

-1

3. (1 point) **Numerical answer:** What would be the updated weight vector θ be after we pass example 2 through the Perceptron algorithm?

[1, -2]

4. (1 point) **Numerical answer:** What would be the updated bias b be after we pass example 2 through the Perceptron algorithm?

-1

5. (1 point) **Numerical answer:** What would be the updated weight vector θ be after we pass example 3 through the Perceptron algorithm?

[1, -2]

6. (1 point) **Numerical answer:** What would be the updated bias b be after we pass example 3 through the Perceptron algorithm?

-1

7. (1 point) **True or False:** Your friend stops you here and tells you that you do not need to update the Perceptron weights or the bias anymore, is this true or false?

- ☐ True
☐ False

True, all points are classified correctly.

8. (2 points) **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 $\text{sign}(w\mathcal{F}(x^{(i)}) + b) \neq y^{(i)}$:

$$w' = w + y^{(i)}\mathcal{F}(x^{(i)})$$

$$b' = b + y^{(i)}$$

- ☐ True
☐ False

True

5 Linear Regression

1. (1 point) **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 now included, what are the dimensions of X , y , θ in the closed form equation?

- ☐ 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. (1 point) (True or False) A multi-layer perceptron model with linear activation is equivalent to linear regression model.

- ☐ True
☐ False

True

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

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

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

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

- ☐ True
☐ False

False.

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

2.

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

- (i) [2 pts] 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 $\|y - w^T x\|_2^2$.

Circle one: True False

One line justification (only if False):

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

- (ii) [4 pts] Consider n data points, each with one feature x_i and an output y_i . In linear regression, we assume $y_i \sim \mathcal{N}(wx_i, \sigma^2)$ and compute \hat{w} through MLE.

Suppose $y_i \sim \mathcal{N}(\log(wx_i), 1)$ instead. Then the maximum likelihood estimate \hat{w} is the solution to the following equality:

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

Circle one: True False

Brief explanation:

False. The likelihood function can be written as

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

Differentiating wrt w and setting to zero gives us

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

- (iii) [3 pts] Consider a linear regression model with only one parameter, the bias, ie., $y = \beta_0$. Then given n data points (x_i, y_i) (where x_i is the feature and y_i is the output), minimizing the sum of squared errors results in β_0 being the median of the y_i values.

Circle one: True False

Brief explanation:

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

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

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

Circle one: True False

Brief explanation:

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.

6. 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 $S = \{(x_1, y_1), \dots, (x_n, y_n)\}$ 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)

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

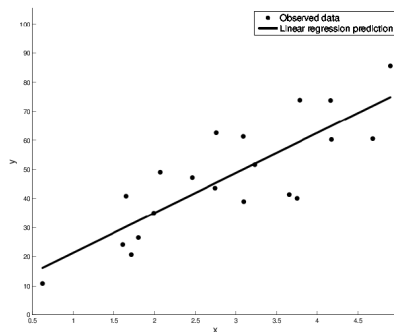
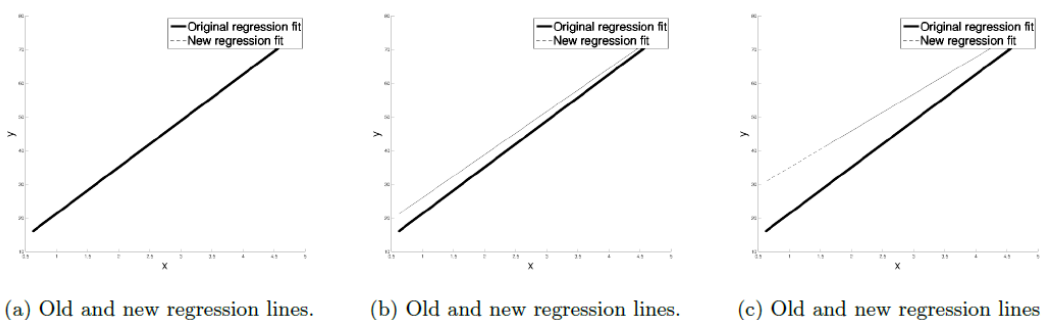


Figure 3: An observed data set and its associated regression line.

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

this tradeoff:

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

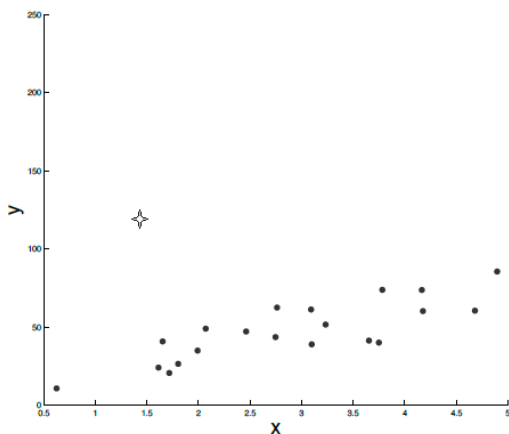
where $\lambda \geq 0$. It is possible to derive a closed form expression for the parameter vector \hat{w} that minimizes this cost function. The gradient, using matrix notation, is

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

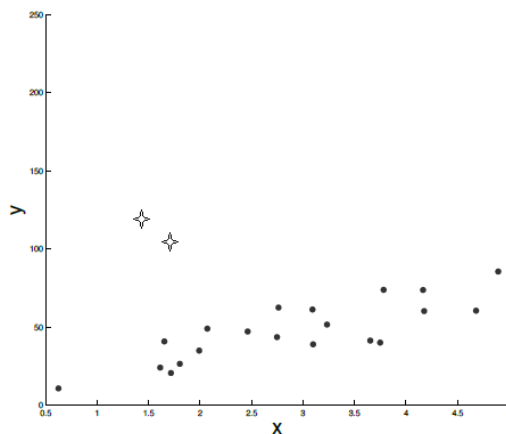
where $X \in \mathbb{R}^{n \times d}$ is the matrix with the the training input instances on the rows (x_i on row i), and $y \in \mathbb{R}^n$ is the vector of training output instances.

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

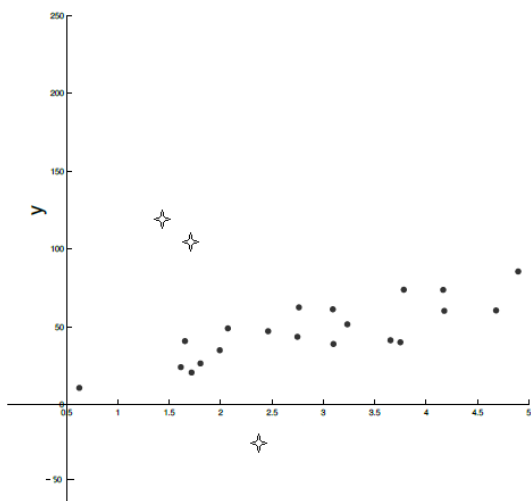
$$\nabla J(w) = 0 \implies \hat{w} = (X^T X + \lambda I)^{-1} X^T y.$$



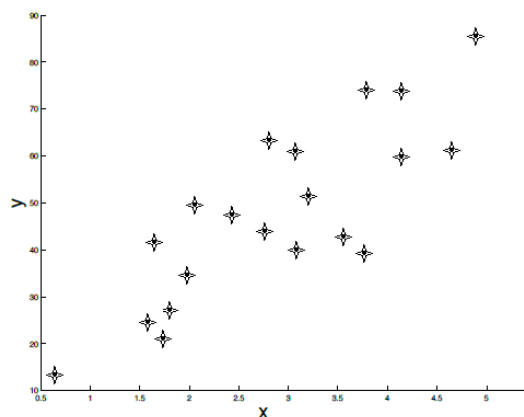
(a) Adding one outlier to the original data set.



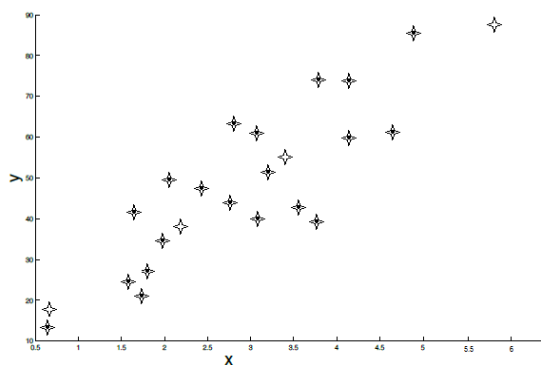
(b) Adding two outliers to the original data set.



(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} .

2. [3 pts.] What is the meaning of the λ parameter? What kind of tradeoff between training error and model complexity we have when λ approaches zero? What about when λ goes to infinity?

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

3. [5 pts.] Answer true or false to each of the following questions and provide a brief justification for your answer:

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

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

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

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

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

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

- [1 pt.] **T or F:** Minimizing the sum of squared errors arises as the result of maximizing the conditional log likelihood under a model where the output is generated as a linear function of the input with additive gaussian noise.

True. If the output is generated as a linear function of the input with additive gaussian noise, maximizing the conditional log likelihood is equivalent to minimizing the sum of squared errors.

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

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

6 Optimization

1. **Select all that apply:** Which of the following are correct regarding Gradient Descent (GD). Assume data log-likelihood is $L(\theta|X)$, which is a function of the parameter θ , and the objective function is negative log-likelihood .

- ☐ GD requires that $L(\theta|X)$ is concave with respect to parameter θ in order to converge
- ☐ GD requires that $L(\theta|X)$ is convex with respect to parameter θ in order to converge
- ☐ GD update rule is $\theta \leftarrow \theta - \alpha \nabla_{\theta} L(\theta|X)$
- ☐ Given a fixed small learning rate (say $\alpha = 10^{-10}$), GD will always reach the optimum after infinite iterations (assume that the objective function satisfies the convergence condition).

A

Analysis: C should replace minus with plus. D is wrong because it is possible that θ will jump around the minimum and never reach the optimum even though α is (finitely) small.

2. **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.

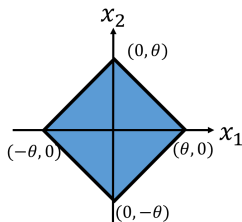
B.

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 since they can converge if the function is convex, not just strongly convex.

3. Let X_1, X_2, \dots, X_N be i.i.d. data 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 $L1$ norm. Please find the maximum likelihood estimator of θ .

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

Analysis:

The likelihood function is

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

To maximize likelihood, we want θ to be as small as possible with the constraint of $\max_{1 \leq i \leq N} \|X_i\| \leq \theta$, otherwise the likelihood drops to 0. So the MLE of θ is

$$\hat{\theta} = \max_{1 \leq i \leq N} \|X_i\|$$

4. **Short answer:** Suppose we want to model a 1-dimensional dataset of N real valued features $(x^{(i)})$ and targets $(y^{(i)})$ by:

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

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 in gradient descent.

Cannot be found analytically.

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

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

Update rule is thus

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