

Homework 7

Loss Functions and Linear Algebra


EECS 398-003: Practical Data Science, Fall 2024

Due Thursday, October 24th at 11:59PM

Instructions

Welcome to Homework 7! In this homework, you'll gain a strong understanding of a key concept in machine learning: loss functions. Along the way, you'll practice working with summation notation, derivatives, limits, and linear algebra, all concepts that are crucial to machine learning. See the [Readings section of the Resources tab on the course website](#) for supplemental resources.

You are given six slip days throughout the semester to extend deadlines. See the [Syllabus](#) for more details. With the exception of using slip days, late work will not be accepted unless you have made special arrangements with your instructor.

To access this notebook, you'll need to clone our [public GitHub repository](#). The  [Environment Setup](#) page on the course website walks you through the necessary steps.

Unlike other homeworks, you **are not** going to submit this notebook! Instead, you will write **all** of your answers to the questions in this homework in a separate PDF. You can create this PDF either digitally, using your tablet or using [Overleaf + LaTeX](#) (or some other sort of digital document), or by writing your answers on a piece of paper and scanning them in.

Make sure to show your work for all questions! Answers without work shown may not receive full credit.

This homework is worth a total of **56 points**, all of which come from **manual grading**. The number of points each question is worth is listed at the start of each question. **All questions in the assignment are independent, so feel free to move around if you get stuck.** Tip: if you're using Jupyter Lab, you can see a Table of Contents for the notebook by going to View > Table of Contents.

To get started, run the cell below. There's no need to `import otter` at the top, since there are no autograder tests. But, you will still run some code.

```
In [2]: import pandas as pd
import numpy as np
import math as math
import plotly
import plotly.figure_factory as ff
import plotly.graph_objects as go
import plotly.express as px
from plotly.subplots import make_subplots
import plotly.io as pio

# Preferred styles
pio.templates["pds"] = go.layout.Template(
    layout=dict(
        margin=dict(l=30, r=30, t=30, b=30),
        autosize=True,
        width=600,
        height=400,
        xaxis=dict(showgrid=True),
        yaxis=dict(showgrid=True),
        title=dict(x=0.5, xanchor="center"),
    )
)
pio.templates.default = "simple_white+pds"
pd.options.plotting.backend = 'plotly'

import warnings
warnings.simplefilter('ignore')
```

Question 1: Imputation Returns

Earlier in the semester, we learned about mean imputation, a technique for handling missing values in a dataset. Specifically, with mean imputation, we fill in all of the missing values in a column with the mean of the observed values in that column. (Here, we'll just consider *unconditional* mean imputation.)

One of the observations we made in [Lecture 8](#) is that when performing mean imputation:

- the mean of the imputed column is **the same as** the mean of the observed values, pre-imputation, and
- the standard deviation of the imputed column is **less than** the standard deviation of the observed values, pre-imputation.

For clarity, let's illustrate that fact once more. Run the cell below to load in the same

`heights` DataFrame we used in Lecture 8.

```
In [2]: heights = pd.read_csv('data/heights-missing-2.csv')
heights.head()
```

```
Out[2]:
```

	father	mother	gender	child
0	78.5	67.0	male	NaN
1	78.5	67.0	female	69.2
2	78.5	67.0	female	69.0
3	78.5	67.0	female	69.0
4	75.5	66.5	male	NaN

The `'child'` column in `heights` has many missing values.

```
In [3]: # 169 missing values, 765 present values.
original_heights = heights['child']
original_heights.isna().value_counts()
```

```
Out[3]: child
False      765
True       169
Name: count, dtype: int64
```

When dropping all missing values, the mean and standard deviation of `heights` is below:

```
In [4]: original_heights = heights['child']
original_heights.mean()
```

```
Out[4]: 67.10339869281046
```

```
In [5]: # We'll talk about what ddof=0 does later in the question.
# For now, just interpret this result as the standard deviation.
original_heights.std(ddof=0)
```

```
Out[5]: 3.520474411620415
```

After filling in missing `'child'` heights with the mean of the observed heights, we have:

```
In [6]: imputed_heights = original_heights.fillna(original_heights.mean())

# The same as original_heights.mean()!
imputed_heights.mean()
```

```
Out[6]: 67.10339869281046
```

```
In [7]: # Smaller than original_heights.std(ddof=0)!
        imputed_heights.std(ddof=0)
```

```
Out[7]: 3.1860931691398657
```

Again, we see that the mean pre-imputation and post-imputation is the same, but the standard deviations are different. But is the mean always guaranteed to be the same after performing mean imputation? And what is the relationship between the standard deviation pre-imputation, ≈ 3.52047 , and the standard deviation post-imputation, ≈ 3.18609 ?

In this question, we will mathematically **prove** the relationships we're seeing above in more generality, to better understand the properties of mean imputation. This will also give us practice with manipulating equations involving summations and squares, which we'll need to do a lot to understand the machinery behind machine learning.

Question 1.1 2 Points

Consider a column of n numbers, y_1, y_2, \dots, y_n with mean M and standard deviation S , where the standard deviation is defined as follows:

$$S = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - M)^2}$$

Suppose we introduce k new values to the dataset, $y_{n+1}, y_{n+2}, \dots, y_{n+k}$, all of which are equal to M . (This is like mean imputation, if the full dataset had $n + k$ values, with n observed/not missing and k missing, and we imputed the k missing values with the mean of the observed, M .)

Let the new mean and standard deviation of all $n + k$ values be M' and S' , respectively.

Prove that $M' = M$.

Some guidance: It's not sufficient to provide a verbal argument. Instead, start with the definition of $M' = \frac{1}{n+k} \sum_{i=1}^{n+k} y_i$, and manipulate the sum to show that it's equal to M .

1.1

We want to show that $M' = M$.

We know that $M' = \frac{1}{n+k} \sum_{i=1}^{n+k} y_i$ and that $M = \frac{1}{n} \sum_{i=1}^n y_i$

Additionally we know that $y_{n+1} \dots y_{n+k}$ are all M .

So we may show;

$$M' = \frac{1}{n+k} \sum_{i=1}^{n+k} y_i \implies \frac{1}{n+k} (\sum_{i=1}^n y_i + \sum_{i=n+1}^{n+k} y_i)$$

Since we know that $y_{n+1} \dots y_{n+k}$ are all M and that $\sum_{i=1}^n y_i = Mn$, we may write:

$$\frac{1}{n+k} (Mn + Mk) \implies M \left(\frac{1}{n+k} \right) (n+k) = M$$

Thus we have shown that $M' = M$.

Question 1.2 4 Points

Using the same definitions as in Question 1.1, find S' in terms of M , n , k , and S . Show your work.

Some guidance:

- You may not need to use all of these variables in your answer.
- To verify your answer is correct, plug in $M = 67.10340$, $n = 765$, $k = 169$, and $S = 3.52047$. The answer you get should be approximately 3.18609, as we saw at the start of the question.

1.2

To find S' we need to adjust the original S to accommodate k and make sure the error is accurately taken. That is, we need to tweak the sum so that it considers $y_{n+1} \dots y_{n+k}$. Additionally since we are adding more values into the dataset we also need to adjust the denominator so that it reflects the added values. M will remain the same as we proved in 1.1 we don't need to use S since that was derived without considering the extra M 's that we are adding. Finally we would result with:

$$S' = \sqrt{\frac{1}{n+k} \left(\sum_{i=1}^n (y_i - M)^2 + \sum_{i=n+1}^k (y_i - M)^2 \right)} \implies$$

$$S' = \sqrt{\frac{1}{n+k} \sum_{i=1}^{n+k} (y_i - M)^2}$$

While this form would give us S' we need to express S' in terms of M, n, k and S ,

So we may write:

$$S'^2 = \frac{1}{n+k} \left(\sum_{i=1}^n (y_i - M)^2 + \sum_{i=n+1}^k (y_i - M)^2 \right)$$

Since all y_i after n is all M , we may conclude that $\sum_{i=n+1}^k (y_i - M)^2 = 0$

Also from the question in 1.1 we may observe that $\sum_{i=1}^n (y_i - M)^2 = nS^2$

So we may write:

$$S'^2 = \frac{1}{n+k} (nS^2 + 0) \implies S'^2 = \frac{n}{n+k} (S^2) \implies S' = S \sqrt{\frac{n}{n+k}}$$

```
In [206... M = 67.10340
n = 765
k = 169
S = 3.52047

tsum = 0

for i in imputed_heights:
    tsum += (i-M)**2
s = math.sqrt((1/(n+k)) * tsum)
print(s)

s_1 = math.sqrt((n/(n+k))) * S
print(s_1)
```

```
3.1860931691401397
```

```
3.186089176543407
```

Question 2: Relative Squared Loss

In [Lecture 14](#), we introduced the "modeling recipe" for making predictions:

1. Choose a model.
2. Choose a loss function.
3. Minimize average loss to find optimal model parameters.

The first instance of this recipe saw us choose:

1. The constant model, $H(x) = h$.
2. The squared loss function: $L_{\text{sq}}(y_i, h) = (y_i - h)^2$.
3. The average squared loss across our entire dataset, then, was:

$$R_{\text{sq}}(h) = \frac{1}{n} \sum_{i=1}^n (y_i - h)^2$$

which, using calculus, we showed is minimized when:

$$h^* = \text{Mean}(y_1, y_2, \dots, y_n)$$

This means that using the squared loss function, the **best** constant prediction is

$$h^* = \text{Mean}(y_1, y_2, \dots, y_n).$$

In this question, you will find the best constant prediction when using a different loss function. In particular, here, we'll explore the **relative squared loss** function, $L_{\text{rsq}}(y_i, h)$:

$$L_{\text{rsq}}(y_i, h) = \frac{(y_i - h)^2}{y_i}$$

Throughout this question, assume that each of y_1, y_2, \dots, y_n is positive.

Question 2.1 2 Points

Determine $\frac{d}{dh} L_{\text{rsq}}(h)$, the derivative of the relative squared loss function with respect to h .

(Technically, this is a **partial** derivative, since there are other variables in the definition of $L_{\text{rsq}}(h)$.)

2.1

Since $L_{\text{rsq}}(h)$ is $\frac{(y_i - h)^2}{y_i}$ we may write

$$\frac{d}{dh} L_{\text{rsq}}(h) = \frac{d}{dh} \frac{(y_i - h)^2}{y_i}$$

Here we may ignore the constants and apply the power rule:

$$\frac{d}{dh} \frac{(y_i - h)^2}{y_i} \implies \frac{1}{y_i} * 2 * (y_i - h) * -1 \implies \frac{1}{y_i} * 2 * (h - y_i)$$

Here we may distribute the $\frac{1}{y_i}$:

$$\frac{2(h - y_i)}{y_i}$$

Question 2.2 4 Points

What value of h minimizes average loss when using the relative squared loss function – that is, what is h^* ? Your answer should only be in terms of the variables n, y_1, y_2, \dots, y_n , and any constants.

2.2

When the first derivative of $f(x)$ is 0 (when $f'(x) = 0$) we now that we have hit a maximum or minimum.

Thus we must find $R_{rsq}'(h) = 0$.

we know that $R'_{rsq}(h) = 2 * \sum_{i=1}^n (\frac{(h-y_i)}{y_i})$

so,

$$\frac{2}{n} * \sum_{i=1}^n (\frac{(h-y_i)}{y_i}) = 0 \implies$$

$$\sum_{i=1}^n (\frac{(h-y_i)}{y_i}) = 0 \implies$$

$$\sum_{i=1}^n (\frac{(h)}{y_i} - 1) = 0 \implies$$

$$h \sum_{i=1}^n (\frac{1}{y_i}) - n = 0 \implies$$

$$h = \frac{n}{\sum_{i=1}^n (\frac{1}{y_i})}$$

Question 2.3 3 Points

Let $C(y_1, y_2, \dots, y_n)$ be your minimizer h^* from Question 2.2. That is, for a particular dataset y_1, y_2, \dots, y_n , $C(y_1, y_2, \dots, y_n)$ is the value of h that minimizes empirical risk for relative squared loss on that dataset.

What is the value of $\lim_{y_4 \rightarrow \infty} C(1, 3, 5, y_4)$ in terms of $C(1, 3, 5)$? Your answer should involve the function C and/or one or more constants.

Some guidance: To notice the pattern, evaluate $C(1, 3, 5, 100)$, $C(1, 3, 5, 10000)$, and $C(1, 3, 5, 1000000)$.

```
In [34]: def C(l):
          s = 0
          for i in l:
              s += 1/i
          h = (len(l)/s)
          return h
```

```
In [38]: l = [1,3,5,100]

          print("y4 = 100")
          print(C(l))
```



```

print("y4 = 10000")
l = [1,3,5,10000]
print(C(l))

print("y4 = 1000000")
l = [1,3,5,1000000]
print(C(l))

print("y4 = inf")
l = [1,3,5,math.inf]
print(C(l))
l = [1,3,5]
print(C(l)*(4/3))

```

```

y4 = 100
2.591792656587473
y4 = 10000
2.608525530943634
y4 = 1000000
2.6086939508517717
y4 = inf
2.608695652173913
2.608695652173913

```

2.3

$C(1,3,5,y_4)$ is equal to:

$\frac{4}{(\frac{1}{1}+\frac{1}{3}+\frac{1}{5}+\frac{1}{y_4})}$ so, as $y_4 \rightarrow \infty$ y_4 will make less and less of an impact on the sum until

we get:

$\frac{4}{(\frac{1}{1}+\frac{1}{3}+\frac{1}{5})}$ which is the same as $C(1,3,5) \frac{4}{3} = 2.608695652173913$

Thus, as $y_4 \rightarrow \infty$, $C(1,3,5,\text{inf}) = 2.608695652173913$

Question 2.4 2 Points

What is the value of $\lim_{y_4 \rightarrow 0} C(1, 3, 5, y_4)$? Again, your answer should involve the function C and/or one or more constants.

```

In [49]: print("y4 = 0.1")
l = [1,3,5,0.1]
print(C(l))

print("y4 = 0.0001")
l = [1,3,5,0.0001]
print(C(l))

```

```

y4 = 0.1
0.3468208092485549
y4 = 0.0001
0.00039993867606966934

```

In [56]: `4/(1+(1/3)+(1/5)+(math.inf))`

Out[56]: `0.0`

2.4

$C(1,3,5,y_4)$ is equal to:

$$\frac{4}{\left(\frac{1}{1} + \frac{1}{3} + \frac{1}{5} + \frac{1}{y_4}\right)}$$

so, as $y_4 \rightarrow y$, y_4 will make more and more of an impact on the sum until we get:

$$\frac{4}{\left(\frac{1}{1} + \frac{1}{3} + \frac{1}{5} + \frac{1}{0}\right)} \text{ which equals to}$$

$$\frac{4}{\left(\frac{1}{1} + \frac{1}{3} + \frac{1}{5} + \infty\right)} = 0$$

Thus, as $y_4 \rightarrow 0$, $C(1,3,5,y_4) = 0$

This is the equivalent to $C(1,3,5) = 0$

Question 2.5 2 Points

Based on the results of Questions 2.3 and 2.4, when is the prediction $C(y_1, y_2, \dots, y_n)$ robust to outliers? When is it not robust to outliers?

2.5

Based on 2.3 and 2.4, $C(y_1, y_2, \dots, y_n)$ is not very robust to outliers. When if any y_i too close to 0 or much larger than the other elements then the function returns either 0 or 2.61 respectively. This makes makes this particular function not ideal for datasets that don't have elements in some relative range from each other (that is to say that this function only really performs well when elements are closer together the larger the elements are).

Question 3.1 2 Points

What is the value of $\sum_{i=1}^n (y_i - \bar{y})$? Show your work.

3.1

$$\sum_{i=1}^n (y_i - \bar{y}) = \sum_{i=1}^n (y_i) - \sum_{i=1}^n (\bar{y})$$

We know that $\sum_{i=1}^n (y_i) = Mn$ (as we saw in 1.1) and $\sum_{i=1}^n (\bar{y}) = Mn$ as well thus, we get:

$$\bar{y}n - \bar{y}n = 0$$

Question 3.2 2 Points

Show that:

$$R_{sq}(h) = \frac{1}{n} \sum_{i=1}^n ((y_i - \bar{y})^2 + 2(y_i - \bar{y})(\bar{y} - h) + (\bar{y} - h)^2)$$

Some guidance:

- To proceed, start by rewriting $y_i - h$ in the definition of $R_{sq}(h)$ as $(y_i - \bar{y}) + (\bar{y} - h)$. Why is this a valid step?
- Make sure not to expand unnecessarily. Your work should only take ~3 lines.

3.2

Replace $(y_i - h) = (y_i - \bar{y}) + (\bar{y} - h)$ to get

$$R_{sq}(h) = \frac{1}{n} \sum_{i=1}^n (y_i - h)^2 \implies$$

Expand to get:

$$\frac{1}{n} \sum_{i=1}^n ((y_i - \bar{y}) + (\bar{y} - h))^2 \implies$$

Final:

$$\frac{1}{n} \sum_{i=1}^n ((y_i - \bar{y})^2 + 2(y_i - \bar{y})(\bar{y} - h) + (\bar{y} - h)^2)$$

Question 3.3 2 Points

Show that:

$$R_{sq}(h) = \frac{1}{n} \sum_{i=1}^n (y_i - \bar{y})^2 + (\bar{y} - h)^2$$

This is called the **bias-variance decomposition** of $R_{sq}(h)$, which is an idea we'll

revisit in the coming weeks.

Some guidance: At some point, you will need to use your result from Question 3.1.

3.3

$$\begin{aligned} \frac{1}{n} \sum_{i=1}^n ((y_i - \bar{y})^2 + 2(y_i - \bar{y})(\bar{y} - h) + (\bar{y} - h)^2) &\implies \\ \frac{1}{n} (\sum_{i=1}^n (y_i - \bar{y})^2 + 2(\bar{y} - h) \sum_{i=1}^n (y_i - \bar{y}) + (\bar{y} - h)^2 \sum_{i=1}^n (1)) &\implies \\ \frac{1}{n} \sum_{i=1}^n (y_i - \bar{y})^2 + \frac{1}{n} 2(\bar{y} - h) \sum_{i=1}^n (y_i - \bar{y}) + \frac{1}{n} (\bar{y} - h)^2 \sum_{i=1}^n (1) &\implies \\ \frac{1}{n} \sum_{i=1}^n (y_i - \bar{y})^2 + (\bar{y} - h)^2 & \end{aligned}$$

Question 3.4 1 Point

Why does the result in Question 3.3 prove that $h^* = \text{Mean}(y_1, y_2, \dots, y_n)$ minimizes $R_{\text{sq}}(h)$?

3.4

3.3 shows us that the $\frac{1}{n} \sum_{i=1}^n (y_i - \bar{y})^2$ can not be affected by the value of h but the only way to minimise the overall expression is to set $(\bar{y} - h)^2 = 0$. Here it is observable that the function will only be zero when $h = \bar{y}$ thus we may conclude that h^* must be $\text{Mean}(y_1, y_2, \dots, y_n)$.

Question 3.5 1 Point

In Question 3.3, you showed that:

$$R_{\text{sq}}(h) = \frac{1}{n} \sum_{i=1}^n (y_i - \bar{y})^2 + (\bar{y} - h)^2$$

Take a close look at the equation above, then fill in the blank below with **a single word**:

The value of $R_{\text{sq}}(h^*)$, when $h^* = \text{Mean}(y_1, y_2, \dots, y_n)$, is equal to the ____ of the data.

3.5

Variance

Question 4: Probability 🍌 Statistics

In Lecture 14, we discussed the relationship between probability and statistics:

- In probability questions, we're given some model of how the universe works, and it's our job to determine how various samples could turn out.

Example: If we have 5 blue marbles and 3 green marbles and pick 2 at random, what are the chances we see one marble of each?

- In statistics questions, we're given information about a sample, and it's our job to figure out how the universe – or **data generating process** works.

Example: Repeatedly, I picked 2 marbles at random from a bag with replacement. I don't know what's inside the bag. One time, I saw 2 blue marbles, then next time I saw 1 of each, the next time I saw 2 red marbles, and so on. What marbles are inside the bag?

In this question, we'll gain a deeper understanding of this relationship, through the lens of your probability knowledge from EECS 203. To do so, we'll introduce you to a key idea in machine learning and statistics, called **maximum likelihood estimation**.

Click [here](#) to read a lecture note (written by us) that introduces you to maximum likelihood estimation!

You *can* attempt the question without reading the note, but it'll be significantly more difficult.

Question 4.1 2 Points

When you step on campus, each person you see has a 0.1 chance of saying "Go Blue!" to you, independent of all other people.

Tomorrow, what's the probability that the first person to say "Go Blue!" to you is the **6th** person you see?

Leave your answer in unsimplified form. This question should not take very long; think back to the probability distributions you learned in EECS 203 (other than the binomial distribution).

4.1

1-5th person, don't say "Go Blue" so 0.9^5 and the 6th person does so :

$$0.9^5 * 0.1$$

Question 4.2 2 Points

Again, assume that the probability that each person you see has a 0.1 chance of saying "Go Blue!" to you, independent of all other people.

What's the probability that:

- the first person to say "Go Blue!" to you tomorrow is the **6th** person you see, **and**
- the first person to say "Go Blue!" to you the day after tomorrow is the **10th** person you see, **and**
- the first person to say "Go Blue!" to you the day after that is the **2nd** person you see?

Again, leave your answer in unsimplified form. Note that we're asking for a single probability, not three separate probabilities.

First day $\rightarrow 0.9^5 * 0.1$

Second day $\rightarrow 0.9^9 * 0.1$

Third day $\rightarrow 0.9 * 0.1$

Overall the probability is $(0.9^5 * 0.1) * (0.9^9 * 0.1) * (0.9 * 0.1)$

Question 4.3 4 Points

Now, suppose that the probability that each person you see says "Go Blue!" to you is some **unknown parameter**, π . (That is, 0.1 will not appear in the rest of this question.)

Suppose you go to campus on n straight days, and you collect a dataset

x_1, x_2, \dots, x_n , where:

- On Day 1, the first person to say "Go Blue!" to you is the x_1 th person you saw (or "person x_1 "), **and**
- On Day 2, the first person to say "Go Blue!" to you is person x_2 , **and**,
- On Day 3, the first person to say "Go Blue!" to you is person x_3 , **and** so on.
- In general, for $i = 1, 2, \dots, n$, on Day i , the first person to say "Go Blue!" to you is person x_i .

For example, the dataset $x_1 = 5, x_2 = 10, x_3 = 2$ would mean that on Day 1, person 5 was the first to say "Go Blue!"; on Day 2, person 10 was the first to say "Go Blue!"; and on Day 3, person 2 was the first to say "Go Blue!".

Prove that $\log L(\pi)$, the log of the likelihood function for π , is:

$$\log L(\pi) = \log(1 - \pi) \sum_{i=1}^n (x_i - 1) + n \log \pi$$

Some guidance: Try and generalize the calculation you made in Question 4.2.

4.3

In general may construct the probability for a day we may write:

$$x_1 = (1 - \pi)^{n-k} (\pi)^k$$

so the total of a sequence of days is x_1, x_2, x_3 is $x_1 * x_2 * x_3$

so we may write

$$L(\pi) = (1 - \pi)^{\sum_{i=1}^n (x_i - 1)} (\pi)^n \text{ from this we may use log rules to write:}$$

$$\text{Log}(\pi) = \text{Log}((1 - \pi)^{\sum_{i=1}^n (x_i - 1)} (\pi)^n) \implies$$

$$\text{Log}(\pi) = \text{Log}((1 - \pi)^{\sum_{i=1}^n (x_i - 1)}) + \text{Log}((\pi)^n)$$

$$\text{Log}(\pi) = \text{Log}((1 - \pi))^{\sum_{i=1}^n (x_i - 1)} + n \text{Log}(\pi)$$

Thus we have proven the prompt above

Question 4.4 4 Points

Using the result to Question 4.3, find π^* , the maximum likelihood estimate of π given the dataset x_1, x_2, \dots, x_n . Once you've done that, give a brief English explanation of why the value of π^* makes intuitive sense.

4.4

To find π^* we need to take the derivative of the $\text{Log}(\pi)$ and set it to zero.

Thus we may write:

$$\frac{d}{d\pi} L(\pi) = \frac{d}{d\pi} (\text{Log}((1 - \pi))^{\sum_{i=1}^n (x_i - 1)} + n \text{Log}(\pi)) \implies$$

$$\frac{d}{d\pi} (\text{Log}((1 - \pi))^{\sum_{i=1}^n (x_i - 1)} + \frac{d}{d\pi} (n \text{Log}(\pi))) \implies$$

$$\sum_{i=1}^n (x_i - 1) \frac{d}{d\pi} \text{Log}((1 - \pi)) + n \frac{d}{d\pi} (\text{Log}(\pi)) \implies$$

$$\sum_{i=1}^n (x_i - 1) \frac{-1}{(1 - \pi)} + \frac{n}{\pi} \implies$$

$$\frac{-\pi \sum_{i=1}^n (x_i - 1)}{\pi(1-\pi)} + \frac{n(1-\pi)}{\pi(1-\pi)} \implies$$

$$\frac{-\pi \sum_{i=1}^n (x_i - 1) + n(1-\pi)}{\pi(1-\pi)} = 0 \implies$$

$$-\pi \sum_{i=1}^n (x_i - 1) + n(1 - \pi) = 0 \implies -\pi \sum_{i=1}^k (x_i - 1) + n - n\pi = 0 \implies$$

$$-\pi(\sum_{i=1}^n (x_i - 1) + n) + n = 0 \implies$$

$$\pi(\sum_{i=1}^n (x_i - 1) + n) = n \implies$$

$$\pi^* = \frac{n}{\sum_{i=1}^k (x_i - 1) + n}$$

This makes intuitive sense because this essentially evaluates to number of people who did say "Go Blue" divided by the number of people who didn't say "Go Blue". This is, the *probability* that someone in specified days would say "Go Blue"!

Question 5: More and More Losses

As we mentioned in Questions 2 and 3, $h^* = \text{Mean}(y_1, y_2, \dots, y_n)$ is the constant prediction that minimizes mean squared error, i.e. average squared loss:

$$R_{\text{sq}}(h) = \frac{1}{n} \sum_{i=1}^n (y_i - h)^2$$

Related, $h^* = \text{Median}(y_1, y_2, \dots, y_n)$ is the constant prediction that minimizes mean absolute error, i.e. average absolute loss:

$$R_{\text{abs}}(h) = \frac{1}{n} \sum_{i=1}^n |y_i - h|$$

You may notice that the formulas for $R_{\text{sq}}(h)$ and $R_{\text{abs}}(h)$ look awfully similar – they're nearly identical besides the exponent. More generally, for any positive integer p , define the L_p loss as follows:

$$L_p(y_i, h) = |y_i - h|^p$$

With this definition, L_2 loss is the same as squared loss and L_1 loss is the same as absolute loss. The corresponding average loss, for any value of p , is then:

$$R_p(h) = \frac{1}{n} \sum_{i=1}^n |y_i - h|^p$$

Written in terms of $R_p(h)$, we know – from the top of this question – that:

- The minimizer of $R_1(h)$ is $\text{Median}(y_1, y_2, \dots, y_n)$:

$$\text{Median}(y_1, y_2, \dots, y_n) = \underset{h}{\operatorname{argmin}} R_1(h)$$

- The minimizer of $R_2(h)$ is $\text{Mean}(y_1, y_2, \dots, y_n)$:

$$\text{Mean}(y_1, y_2, \dots, y_n) = \underset{h}{\operatorname{argmin}} R_2(h)$$


But what constant prediction h^* minimizes $R_3(h)$, or $R_{10}(h)$, or $R_{10000}(h)$? In this question, we'll explore this idea – more specifically, we'll study how h^* changes as p (the exponent on $|y_i - h|$) increases.

[Lecture 14](#) worked through how to solve for constant prediction h^* that minimized average squared loss (i.e. minimized $R_2(h)$), and we linked to a [video](#) that works through a similar derivation for average absolute loss (i.e. $R_1(h)$). Unfortunately, $p = 1$ and $p = 2$ are the only cases in which we can solve for the minimizer to $R_p(h)$ by hand.

For all other values of p , there is no closed-form solution (i.e. no "formula" for the best constant prediction), and so we need to approximate the solution using the computer. Later in the class, we'll learn how to minimize functions using code we write ourselves (the idea is called gradient descent if you're curious), but for now, we're going to use `scipy.optimize.minimize`, which does the hard work for us.

The `minimize` function is a versatile tool from the `scipy` library that can help us find the input that minimizes the output of a function. Let's test it out.

```
In [3]: from scipy.optimize import minimize
```

Below, we've defined and plotted a quadratic function. We can see  that it's minimized when $x = -4$.

```
In [4]: def f(x):
        return (x + 4) ** 2 - 1
```

```
In [5]: xs = np.linspace(-20, 20)
        ys = f(xs)
        px.line(x=xs, y=ys)
```

But Python doesn't have eyes, so it can't see that the graph is minimized at $x = -4$. `minimize`, though, magically **can** do this minimization!

```
In [6]: # To call minimize, we have to provide an array of initial "guesses"
```

```
# as to where the minimizing input might be.
# For our purposes, using 0 as an initial guess will work fine.
minimize(f, x0=[0])
```

```
Out[6]: message: Optimization terminated successfully.
        success: True
        status: 0
         fun: -0.9999999999999999
          x: [-4.000e+00]
         nit: 3
         jac: [-7.451e-09]
        hess_inv: [[ 5.000e-01]]
         nfev: 8
         njev: 4
```

Above, the `x` attribute of the output tells us that the minimizing input to `f` is `-4.0000`, which is what we were able to see ourselves! Cool.

In this question, we'll deal with the following example array of values, `vals`:

For context, let's see what the distribution of `vals` looks like:

```
In [71]: pd.Series(vals).hist(nbins=40)
```

To reiterate, the constant prediction h^* that minimizes $R_1(h)$ for `vals` is:

```
In [72]: np.median(vals)
```

```
Out[72]: 2.0
```

And the constant prediction h^* that minimizes $R_2(h)$ for `vals` is:

```
In [73]: np.mean(vals)
```

```
Out[73]: 5.714285714285714
```

Question 5.1 0 Points

Complete the implementation of the function `h_star`, which takes in a positive integer `p` and an array `vals` and returns the value of the constant prediction h^* that minimizes average L_p loss for `vals`, i.e. the value of h^* that minimizes $R_p(h)$ for `vals`. Example behavior is given below.

```
>>> h_star(1, vals)
2.0
```

```
>>> h_star(2, vals)
5.714285345730987
```

Some guidance:

- Your solution should use `minimize`, and will likely involve defining a helper function inside.
- It's okay if your example values are slightly different than those above, but they should be roughly the same. (So, it's fine if `h_star(1, vals)` gives you `1.9999999920558864` or something similar.)

We're not autograding Question 5.1, and it's not worth any points. But, you need to do it in order to answer Question 5.2, which is worth points (and which you will answer on paper!).

```
In [8]: def g(x,deg,vals):
        return np.sum(np.abs(vals - x)**deg)
```

```
In [9]: def h_star(deg, vals):
        return minimize(lambda x:g(x,deg, vals), x0=[0]).x[0]

# Feel free to change this input to make sure your function works correct
h_star(77, vals)
```

```
Out[9]: 64.84039484083391
```

Before proceeding, make sure that the following cells both say `True`, otherwise you did something incorrectly:

```
In [10]: np.isclose(h_star(1, vals), np.median(vals))
```

```
Out[10]: True
```

```
In [11]: np.isclose(h_star(2, vals), np.mean(vals))
```

```
Out[11]: True
```

```
In [17]: #vals = np.array([1, 1, 1, 1, 1, 1, 2, 2, 2, 4, 5, 10, 10, 39])
        vals = np.array([ 40, 40, 40, 40, 40, 40, 60, 60, 60, 80, 80, 100])
```

Once you have a working implementation of `h_star`, run the cell below.

```
In [32]: vals = np.array([2,3,4,5,6,7,8,100])
        ps = np.arange(1, 75)
        hs = [h_star(p, vals) for p in ps]
        px.line(x=ps, y=hs).update_layout(xaxis_title=r'p', yaxis_title=r'h^* = m
```

It seems like as p increases, the value of h^* that minimizes $R_p(h)$ approaches some

fixed value. But what is that value? For context, look at `vals` again:

```
In [198... vals
Out[198... array([ 30,  30,  40,  40,  40,  40,  40,  40,  60,  60,  60,  80,  80,
          100])
```

Question 5.2 4 Points

Use the plot above to answer the following prompts:

1. In the `vals` dataset, as p increases, what does the value of h^* that minimizes $R_p(h)$ approach?
2. In any general dataset of values y_1, y_2, \dots, y_n , as p increases, what does the value of h^* that minimizes $R_p(h)$ approach? Why?

Put another way, we're asking you to evaluate the following limit, but using your plot, not calculus (you're welcome 😊):

$$\lim_{p \rightarrow \infty} \left(\operatorname{argmin}_h \frac{1}{n} \sum_{i=1}^n |y_i - h|^p \right)$$

Some guidance:

- To answer the second prompt, try calling `h_star` with different arrays that you create. Try and see if you can find a pattern in the values that `h_star` returns when `p` is very large.
- If you experiment in the way we're suggesting above, you may run into *overflow* errors, where the numbers you're dealing with are too big for Python to compute. (e.g., something like $|100 - 90|^{2000}$ is far too big to be represented).

5.2

1. As p increases h^* approaches 20
2. From the values we may noticed that the limit approaches the mean between the two extremes.

Thus we get $\frac{v_{min} + v_{max}}{2}$

This is because as p increases the differences between all values are exagurated more and more. This effect will impact the outliers the most thus, minimizing the function as p increases means minimising the difference of h with it's extremes. Thus we take the average of the minimum and maximum values in the set.

Question 6: Algebra, Too

In the coming lectures, we'll start formulating the problem of making predictions about future data given past data in terms of matrices and vectors. Why? The answer is simple: doing so will allow us to build models that use multiple input variables (i.e. features) in order to make predictions.


This question serves to review the key linear algebra knowledge you'll need to be familiar with as we start using matrices and vectors in lecture. If any of this feels foreign – and it's totally fine if it does! – review [LARDS: Linear Algebra Review for Data Science](#). We'll link to specific sections in LARDS for each part of this question.

Throughout this question, consider the following vectors in \mathbb{R}^3 , where $\beta \in \mathbb{R}$ is a scalar:

$$\vec{v}_1 = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \quad \vec{v}_2 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \quad \vec{v}_3 = \begin{bmatrix} \beta \\ 1 \\ 2 \end{bmatrix}$$

Question 6.1 2 Points

For what value(s) of β are \vec{v}_1, \vec{v}_2 , and \vec{v}_3 linearly **independent**?

 To review, read LARDS [Section 5](#).

Since $\vec{v}_1 + \vec{v}_2 = \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix}$ Any value besides 1. The reason why no other number makes

\vec{v}_3 not linearly independent is because any other linear combination would affect both the first and the third element, thus any other value for beta will result in \vec{v}_3 being a linearly independent vector.

Question 6.2 1 Point

For what value(s) of β are \vec{v}_1 and \vec{v}_3 orthogonal?

 To review, watch LARDS [Section 2](#).

6.2

For \vec{v}_1 and \vec{v}_3 to be orthogonal we need to have the dot product = 0.

so, $\beta 0 + 1 \cdot 1 + 1 \cdot 2 = 0$

since $0 \neq -3$ there is not value of β such that \vec{v}_1 and \vec{v}_3 would be orthogonal

Question 6.3 1 Point

For what value(s) of β are \vec{v}_2 and \vec{v}_3 orthogonal?

 To review, watch LARDS [Section 2](#).

6.3

For \vec{v}_2 and \vec{v}_3 to be orthogonal we need to have the dot product = 0.

$$1 \cdot \beta + 0 \cdot 1 + 1 \cdot 2 = 0 \implies \beta + 2 = 0$$

$$\beta = -2$$

Question 6.4 2 Points

Regardless of your answers to the previous three parts, in this part, let $\beta = 3$.

Is the vector $\begin{bmatrix} 3 \\ 5 \\ 8 \end{bmatrix}$ in $\text{span}(\vec{v}_1, \vec{v}_2, \vec{v}_3)$? Why or why not?

 To review, watch LARDS [Section 4](#) and read [Section 5](#).

6.4

For the vector above to be in the span, we must show that there exist some coefficients for $\vec{v}_1, \vec{v}_2, \vec{v}_3$ such that

$$a\vec{v}_1 + b\vec{v}_2 + c\vec{v}_3 = \begin{bmatrix} 3 \\ 5 \\ 8 \end{bmatrix}$$

$$\text{Remember that: } \vec{v}_1 = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \quad \vec{v}_2 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \quad \vec{v}_3 = \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix}$$

so we may write:

$$a \cdot 0 + b \cdot 1 + c \cdot 3 = 3$$

$$a + b + c = 5$$

$$a + b + c = 8$$

so, we know that $b = 3 - 3c$ and $a = 5 - c$, thus using the third equation we may write $5 - c + 3 - 3c + 2c = 8 \implies -2c = 0 \implies c = 0$.

so, we may show that $b = 3$ and $a = 5$.

Since these coefficients exist,

Thus $\begin{bmatrix} 3 \\ 5 \\ 8 \end{bmatrix}$ is in the $\text{span}(\vec{v}_1, \vec{v}_2, \vec{v}_3)$

Question 6.5 3 Points

What is the projection of the vector $\begin{bmatrix} 3 \\ 15 \\ 21 \end{bmatrix}$ onto \vec{v}_1 ? Give your answer in the form of a vector.

 To review, watch LARDS [Section 6](#).

6.5

Let's refer to $\begin{bmatrix} 3 \\ 15 \\ 21 \end{bmatrix}$ as \vec{v}_p

Since we are projecting \vec{v}_p to \vec{v}_1 we may find the coefficient using:

$$\frac{\vec{v}_p \cdot \vec{v}_1}{\vec{v}_1 \cdot \vec{v}_1}$$

so:

$$\frac{0 \cdot 3 + 15 \cdot 1 + 21 \cdot 1}{1^2 + 1^2} = \frac{36}{2}$$

Thus, the projection of \vec{v}_p onto $\text{span}(\vec{v}_1)$ is $18\vec{v}_1$

Question 6.6 4 Points

What is the orthogonal projection of the vector $\begin{bmatrix} 3 \\ 15 \\ 21 \end{bmatrix}$ onto $\text{span}(\vec{v}_1, \vec{v}_2)$?

The answer is a vector, \vec{z} , which can be written in the form:

$$\vec{z} = \lambda_1 \vec{v}_1 + \lambda_2 \vec{v}_2$$

Your job is to find the values of scalars λ_1 and λ_2 , and then, the vector \vec{z} . As done in LARDS [Section 8](#), one of the intermediate steps in answering this question involves defining a particular matrix X and computing $(X^T X)^{-1} X^T$.

 To review, watch LARDS [Sections 6-8](#).

$$X = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \text{ which is } [1 \quad 0]$$

From the slides we can find that the coefficients can be found using $(X^T X)^{-1} X^T \vec{v}_p$

$$\vec{w}^* = \left(\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 15 \\ 21 \end{bmatrix}$$

$$\vec{w}^* = \left(\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 15 \\ 21 \end{bmatrix}$$

$$\vec{w}^* = \left(\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \right)^{-1} \begin{bmatrix} 24 \\ 36 \end{bmatrix}$$

$$\vec{w}^* = \begin{bmatrix} \frac{2}{3} + & \frac{-1}{3} \\ \frac{-1}{3} + & \frac{2}{3} \end{bmatrix} \begin{bmatrix} 24 \\ 36 \end{bmatrix} = \begin{bmatrix} 24\frac{2}{3} + & 36\frac{-1}{3} \\ 24\frac{-1}{3} + & 36\frac{2}{3} \end{bmatrix}$$

$$\vec{w}^* = \begin{bmatrix} 16 + & -12 \\ -8 + & 24 \end{bmatrix} = \begin{bmatrix} 4 \\ 16 \end{bmatrix}$$

Thus we may conclude that $\lambda_1 = 16$ and $\lambda_2 = 4$

so,

$$\vec{z} = 16\vec{v}_1 + 4\vec{v}_2$$

Finish Line 

Congratulations! You're ready to submit Homework 7.

Remember, you'll submit Homework 7 as a **PDF** to the **Homework 7 (PDF)** assignment on Gradescope. You won't submit this notebook anywhere; Homework 7 will be entirely manually graded.