CSE 151A - Homework 03

Due: Wednesday, April 22, 2020

Write your solutions to the following problems by either typing them up or handwriting them on another piece of paper. Unless otherwise noted by the problem's instructions, show your work or provide some justification for your answer. Homeworks are due via Gradescope on Wednesday at 11:59 p.m.

Essential Problem 1.

You've been hired by a generic online retailer named after a rainforest named after a river. Your job is to build a model to predict whether or not a particular item will sell. You are provided with a dataset of outcomes for a collection of products:

Brand	Price Range	Condition	Sold
A	High	Used	No
A	High	New	Yes
B	Low	New	Yes
C	Medium	New	Yes
B	Low	Used	No
A	High	New	No
C	High	Used	Yes
A	Medium	Used	Yes
B	Medium	Used	No
C	Low	New	No
B	Low	Used	Yes

Using a Naïve Bayes classifier and the data above, predict if a product with Brand = B, Price Range = Medium, Condition = Used will sell or not. Show your calculations.

Solution:

We start by calculating the class conditional probabilities:

$$P(\text{Brand} = \text{B} \,|\, \text{Sold} = \text{Yes}) = \frac{2}{6}$$

$$P(\text{Brand} = \text{B} \,|\, \text{Sold} = \text{No}) = \frac{2}{5}$$

$$P(\text{Price Range} = \text{Medium} \,|\, \text{Sold} = \text{Yes}) = \frac{2}{6}$$

$$P(\text{Price Range} = \text{Medium} \,|\, \text{Sold} = \text{No}) = \frac{1}{5}$$

$$P(\text{Condition} = \text{Used} \,|\, \text{Sold} = \text{Yes}) = \frac{3}{6}$$

$$P(\text{Condition} = \text{Used} \,|\, \text{Sold} = \text{No}) = \frac{3}{5}$$

The prior probabilities are $P(\text{Sold} = \text{Yes}) = 6/11$ and $P(\text{Sold} = \text{No}) = 5/11$. Therefore:

$$\begin{aligned} &P(\text{Sold} = \text{Yes} \,|\, \text{Brand} = \text{B}, \text{Price Range} = \text{Medium}, \text{Condition} = \text{Used}) \\ &\quad \propto \frac{2}{6} \cdot \frac{2}{6} \cdot \frac{3}{6} \cdot \frac{6}{11} \\ &= \frac{72}{2376} \approx 0.03 \end{aligned}$$

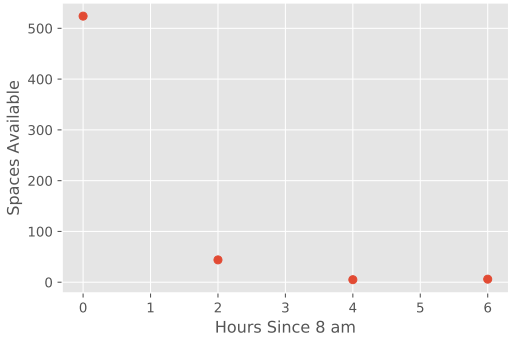
$$\begin{aligned} &P(\text{Sold} = \text{No} \,|\, \text{Brand} = \text{B}, \text{Price Range} = \text{Medium}, \text{Condition} = \text{Used}) \\ &\quad \propto \frac{2}{5} \cdot \frac{1}{5} \cdot \frac{3}{5} \cdot \frac{5}{11} \\ &= \frac{30}{1375} \approx 0.021 \end{aligned}$$

Because the former is larger, we predict that the product will be **sold**.

Essential Problem 2.

The table below and the accompanying plot show the total number of "S" parking spaces available on the UCSD campus at various times on a typical Tuesday:[1]

Hours Since 8 am	Spaces Available
0	524
2	44
4	5
6	6



a) Use least squares regression to find a prediction rule of the form $H(x) = w_1 x + w_0$ for the number of spaces available. The variable x represents the number of hours since 8 am. *Hint*: you can check whether your answer is reasonable by plotting.

Solution: We'll use the familiar formulas for the slope and intercept of the linear least squares regression line:

$$w_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$w_0 = \bar{y} - b_1 \bar{x}$$

In this case, the x_i are the hours since 8 am, and the y_i are the number of spaces available. Therefore: $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i = 3$ and $\bar{y} = \sum_{i=1}^{n} y_i = 144.75$.

[1] Parking availability for January 14, 2020 was scraped from the UCSD transportation office website. Lot P701 was excluded.

$x_i - \bar{x}$	$y_i - \bar{y}$	$(x_i - \bar{x})(y_i - \bar{y})$	$(x_i - \bar{x})^2$
-3.0	379.25	-1137.75	9.0
-1.0	-100.75	100.75	1.0
1.0	-139.75	-139.75	1.0
3.0	-138.75	-416.25	9.0
	sum:	-1593.00	20.0

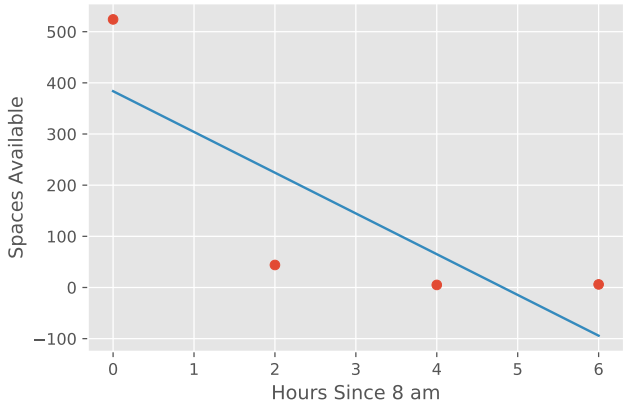
This gives $w_1 = -1593/20 = -79.65$. Plugging this into the formula for w_0, we find:

$$w_1 = \bar{y} - w_1\bar{x} = 383.7$$

So our prediction rule is:

$$H(x) = -79.65x + 383.7$$

The plot below shows our linear prediction rule:



You might like to know that `numpy` has a function which performs least squares regression. It is called `np.polyfit`. Here's a demo:

```
>>> x = [0, 2, 4, 6]
>>> y = [524, 44, 5, 6]
>>> import numpy as np
>>> np.polyfit(x, y, deg=1)
array([-79.65, 383.7 ])
```

b) Use your prediction rule from above to predict the number of parking spots available at 9 am. Use it again to predict the number of parking spaces available at 4 pm. Do you believe that your prediction rule makes good predictions? Why or why not?

Solution: Our prediction rule was $H(x) = -79.65x + 383.7$, where x is the number of hours past 8am. Our prediction for 9 am is found by calculating $H(1)$:

$$H(1) = -79.65 + 383.7 = 304.05$$

Since 4 pm is 8 hours after 8 am, our prediction for 4 pm is:

$$H(8) = -79.65 \cdot 8 + 383.7 = -253.5$$

It seems that our linear prediction rule does not make good predictions. For one, it is predicting

a negative number of parking spaces at 4 pm, which is not possible. Second, even when its predictions are positive, they don't match the data particularly well.

c) Use least squares regression to find a prediction rule of the form $H(x) = \frac{w_1}{x+1} + w_0$.

Solution: We wish to fit a function of the form

$$H(x) = \frac{w_1}{x+1} + w_0.$$

If we define $z(x) = 1/(x+1)$, then our prediction rule becomes:

$$H(x) = w_1 z(x) + w_0.$$

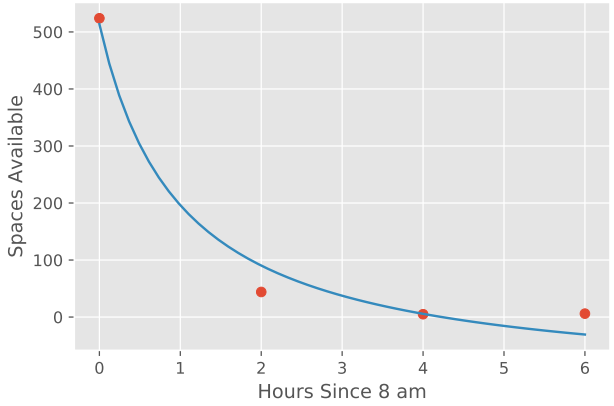
This is a linear function of w_1 and w_0, so our formulas apply. We start by making a table of the various quantities involved:

z_i	$z_i - \bar{z}$	$y_i - \bar{y}$	$(z_i - \bar{z})(y_i - \bar{y})$	$(z_i - \bar{z})^2$
1.00	0.58	379.25	220.33	0.34
0.33	-0.09	-100.75	8.64	0.01
0.20	-0.22	-139.75	30.61	0.05
0.14	-0.28	-138.75	38.32	0.08
		sum:	297.90	0.47

Here, $z_i = z(x_i) = 1/(x_i+1)$. Using our formulas: $w_1 = 297.90/0.47 = 635.02$ and $w_0 = -121.35$. So our prediction rule is:

$$H(x) = \frac{635.02}{x+1} - 121.35$$

Here is our prediction rule in action:



It certainly looks better than the linear prediction rule, but it still predicts negative parking spaces.

d) It looks like the number of parking spaces decreases exponentially as the day goes on. A better prediction rule might be $H_{\text{exp}}(x) = 524 \cdot e^{-wx}$, where w is a parameter that we want to learn from data. Write down the general formula for computing the mean squared error of this prediction rule as a function of w, using x_i for the hours since 8 a.m. and y_i for the number of parking spaces, and n for the number of data points.

Hint: the formula for the mean squared error of a linear prediction rule, $H(x) = w_1 x + w_0$, is:

$R(w_1, w_0) = \frac{1}{n}\sum_{i=1}^{n}\left((w_1 x_i + w_0) - y_i\right)^2$.

Solution: In general, the mean squared error of a prediction rule $H(x)$ is:

$$R(H) = \frac{1}{n}\sum_{i=1}^{n}\left(H(x_i) - y_i\right)^2$$

in this case, $H(x) = 524 \cdot e^{-wx}$. Therefore, the MSE of this prediction rule is:

$$R(w) = \frac{1}{n}\sum_{i=1}^{n}\left(524 \cdot e^{-wx_i} - y_i\right)^2$$

Essential Problem 3.

Suppose that the age and net worth of each representative in the US House of Representatives is collected, and least squares is used to fit a linear prediction function $H(x) = w_1 x + w_0$ for predicting a representative's worth from their age.

Now suppose that the world's richest person, Jeff Bezos (net worth \$139 billion), has decided to run for the US House of Representatives in a special election! Assume that Bezos replaces a congressperson who is the same age as him: 56 years old. The net worth of the replaced representative was \$1 million. Assume, too, that that the *variance* in the age of the House of Representatives is 10 years.[2] If a new linear predictor $H'(x) = w_1' x + w_0'$ is fit using the new data with Bezos included, what is the difference between the new slope w_1' and the old? Show your calculations. You may assume that Bezos is the only new member of the House of Representatives.

For this problem, you'll need to use the fact that the US House of Representatives has 435 members and that their average age is 58 years.

Solution: We consider age and net worth as x, y respectively. Let the index of the replaced congressperson be j.

$$w_1' - w_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} - \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

We can separate j from our summations in the numerator. Fortunately, we notice that the $x_j' = x_j$, so we can write them equivalently. Likewise, $\bar{x}$ and our denominator remain unchanged. To make notation easier, define $\bar{y}_{i\neq j} := \frac{1}{n-1}\sum_{i\neq j} y_i$.

$$= \frac{\sum_{i\neq j}(x_i - \bar{x})(y_i - \bar{y}_{i\neq j}) + (x_j - \bar{x})(y_j' - \bar{y}')}{\sum_{i=1}^{n}(x_i - \bar{x})^2} - \frac{\sum_{i\neq j}(x_i - \bar{x})(y_i - \bar{y}_{i\neq j}) + (x_j - \bar{x})(y_j - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

The denominator can be rewritten as $n \cdot \mathrm{Var}(x)$, and we can factor it out. Our shared terms cancel.

$$= \frac{(x_j - \bar{x})(y_j' - \bar{y_j}') - (x_j - \bar{x})(y_j - \bar{y})}{n \cdot \mathrm{Var}(x)}$$
$$= \frac{(x_j - \bar{x})(y_j' - \bar{y}' - y_j + \bar{y})}{n \cdot \mathrm{Var}(x)}$$

[2] recall that the variance is $\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$.

We can reason that $\bar{y}' = \bar{y} - \frac{1\times10^6}{435} + \frac{139\times10^9}{435}$, which simplifies to $\bar{y}' = \bar{y} + \frac{138.999\times10^9}{435}$

$$= \frac{(56-58)(139\times10^9 - (\bar{y} + \frac{138.999\times10^9}{435}) - 1\times10^6 + \bar{y})}{435\cdot10}$$
$$= \frac{(-2)(138.999\times10^9 - \frac{138.999\times10^9}{435})}{4350}$$
$$\approx -63,760,672 \text{ or -\$63.8 million}$$

Plus Problem 1. (6 plus points)

Consider again the prediction rule $H_{\text{exp}}(x) = 524 \cdot e^{-wx}$ from the parking regression problem above. Unlike the case of linear prediction rules, there's no formula for the minimizer of R_{exp}. Instead, we have to minimize the mean squared error numerically by using gradient descent or some other method. Use the numerical minimization tool of your choice to find the value of w which minimizes the mean squared error of the exponential prediction rule given the data in the table from Essential Problem 2. Submit the value of w that you find, along with your code as a PDF (we will not run your code).

Hint: The Python package `scipy` has a function called `scipy.optimize.minimize` which numerically minimizes a function. There is a short demo notebook on using SciPy for Numerical Optimization Demo. It will show you how to use `scipy.optimize.minimize`, and you can perform your analysis in the notebook itself.

Hint: if you're not using Python, other languages should have something like `scipy.optimize.minimize`. For instance, in Java, the Apache Commons Math library's `BrentOptimizer` will do the same thing; in C++, `dlib`'s `find_min` will work.

Solution: Here is part of the documentation of the `scipy.optimize.minimize` function:

scipy.optimize.**minimize**(*fun, x0, args=(), method=None, jac=None, hess=None, hessp=None, bounds=None, constraints=(), tol=None, callback=None, options=None*) [source]

Minimization of scalar function of one or more variables.

Parameters:

fun : *callable*

The objective function to be minimized.

```
fun(x, *args) -> float
```

where x is an 1-D array with shape (n,) and *args* is a tuple of the fixed parameters needed to completely specify the function.

x0 : *ndarray, shape (n,)*

Initial guess. Array of real elements of size (n,), where 'n' is the number of independent variables.

To use it, we must define a function and specify a starting location. Here is a Python function that computes the mean squared error of the exponential prediction rule:

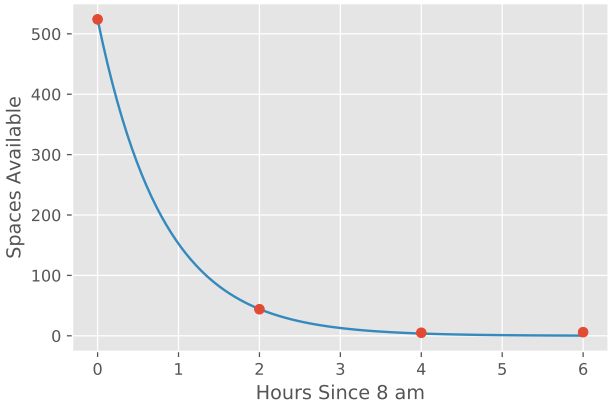
```
x = np.array([0, 2, 4, 6])
y = np.array([524, 44, 5, 6])

def mse(w):
    return ((524 * np.exp(-w*x) - y)**2).mean()
```

Running `scipy.optimize.minimize(mse, 1)` finds an optimizer of $w \approx 1.235$. Our prediction rule is therefore:

$$H(x) = 524 \cdot e^{-1.235\,x}$$

The prediction rule is plotted below.



Looks pretty good!

One advantage of this prediction rule is that it never predicts a negative number of available spaces. As $x \to \infty$, our prediction will go to zero. Of course, we do expect the number of parking spaces to increase once again as people go home for the night, so this model only works for the day time. If we had data for before 8 am and in the evening, we might expect to see two sigmoid functions.

Plus Problem 2. (8 plus points)

This plus problem is a **competition** to see who can achieve the highest accuracy on a machine learning problem. We'll grade the problem as usual, but the person who obtains the highest accuracy will receive 6 extra plus points; second place will receive 4 extra plus points, and third place will receive 2 extra plus points.

An outdoor recreation store wants to start selling surfboards. The company has a large data set containing information about their customers and their hobbies, but it doesn't include whether each person surfs. To predict whether a customer is a surfer (and thus should be sent an advertisement), you will build a machine learning model that predicts whether or not a person surfs given their other hobbies.

A group of 700 of the store's customers were surveyed and asked whether they surf. Their responses, along with other information about them are included in the file: `http://cse151a.com/data/surfer/train.csv`. The columns of this file are:

`id`	the customer identification number
`age`	their age
`name`	their name
`salary`	their salary
`city`	the city/neighborhood they live in
`favorite_color`	their favorite color
`jog`	whether they like to jog (yes = 1, no = 0)
`hike`	whether they like to hike (yes = 1, no = 0)
`paddleboard`	whether they like to paddleboard (yes = 1, no = 0)
`camp`	whether they like to camp (yes = 1, no = 0)
`surf`	whether they like to surf (yes = 1, no = 0)

The file: `http://cse151a.com/data/surfer/test.csv` contains a data set of customers whose surfer status is unknown. Using any of the classification methods we have learned so far in this class: kNN classification or Bayes classifiers (naïve or otherwise), predict whether or not each person in the test data set is a surfer.

Submit your predictions as a zip file to the Gradescope assignment named "Homework 03 - Plus Problem 02". This zip file should contain the following:

- a file named `predictions.csv` containing your predictions (see below).

- a file name `description.md` containing a short (4 sentence maximum) description of your model and what is unique about it (if anything).
- a folder called `code` containing your code files (they can be named whatever you'd like – we will not run your code).

When you are zipping up your files, make sure to zip the files, and not the folder containing the files. That is, if your solutions are in a folder named `my_solution`, run `zip -r upload.zip *` *inside* of `my_solution` instead of `zip -r upload.zip my_solution/`. The autograder will do a rudimentary check to make sure that your uploaded zip has the right structure.

The `predictions.csv` file should have two columns: `id` (the customer ID from the test set), and `surf` (whether or not they surf, 1 meaning yes and 0 meaning no). For instance:

```
id,surf
700,0
701,0
702,0
703,0
704,0
...
```

See http://cse151a.com/data/surfer/predictions.sample.csv for an example of how your `predictions.csv` should look. The linked file is the output of a dummy method which predicts that nobody surfs.

Once you've uploaded your code, you'll be able to see how it performs relative to other submissions by viewing the leaderboard (the button is in the top-right). The accuracy reported on the leaderboard is your accuracy on a *subset* of the test data. The final standings will be determined by computing your accuracy on the *full* test set. As a result, the standing you see on the leaderboard may differ from your final standing!

Note: for this problem, you can use whatever code from whatever open source libraries you'd like: `sklearn`, `scipy`, `JavaML`, `eigen`, `dlib`, etc. For instance, you do **not** need to re-implement Naïve Bayes from scratch. You can also engineer new features from those given to you, make modeling assumptions, whatever you'd like.

We will implement a very simple model to achieve a baseline level of accuracy, and you'll receive full credit if you match or exceed this accuracy. You'll receive partial credit for making predictions that perform better than random.