HOMEWORK 5

HAMSALEKHA PREMKUMAR

Instructions: Although this is a programming homework, you only need to hand in a pdf answer file. There is no need to submit the latex source or any code. You can choose any programming language, as long as you implement the algorithm from scratch.
Use this latex file as a template to develop your homework. Submit your homework on time as a single pdf file to Canvas. Please check Piazza for updates about the homework.

Linear Regression (100 pts total, 10 each)

The Wisconsin State Climatology Office keeps a record on the number of days Lake Mendota was covered by ice at http://www.aos.wisc.edu/~sco/lakes/Mendota-ice.html. Same for Lake Monona: http://www.aos.wisc.edu/~sco/lakes/Monona-ice.html. As with any real problems, the data is not as clean or as organized as one would like for machine learning. Curate two clean data sets for each lake, respectively, starting from 1855-56 and ending in 2018-19. Let x be the year: for 1855-56, $x = 1855$; for 2017-18, $x = 2017$; and so on. Let y be the ice days in that year: for Mendota and 1855-56, $y = 118$; for 2017-18, $y = 94$; and so on. Some years have multiple freeze thaw cycles such as 2001-02, that one should be $x = 2001, y = 21$.

1. Plot year vs. ice days for the two lakes as two curves in the same plot. Produce another plot for year vs. $y_{Monona} - y_{Mendota}$.

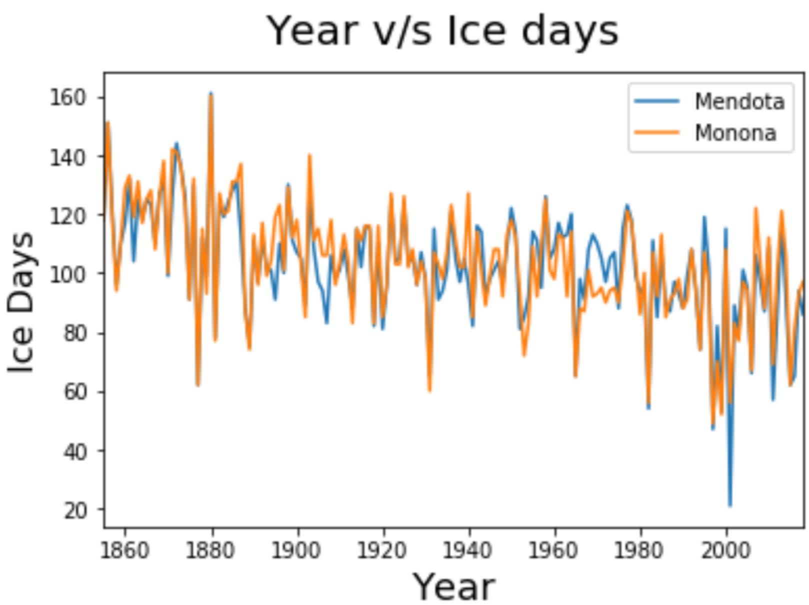

Fig: Plot of year v/s ice days for the two lakes

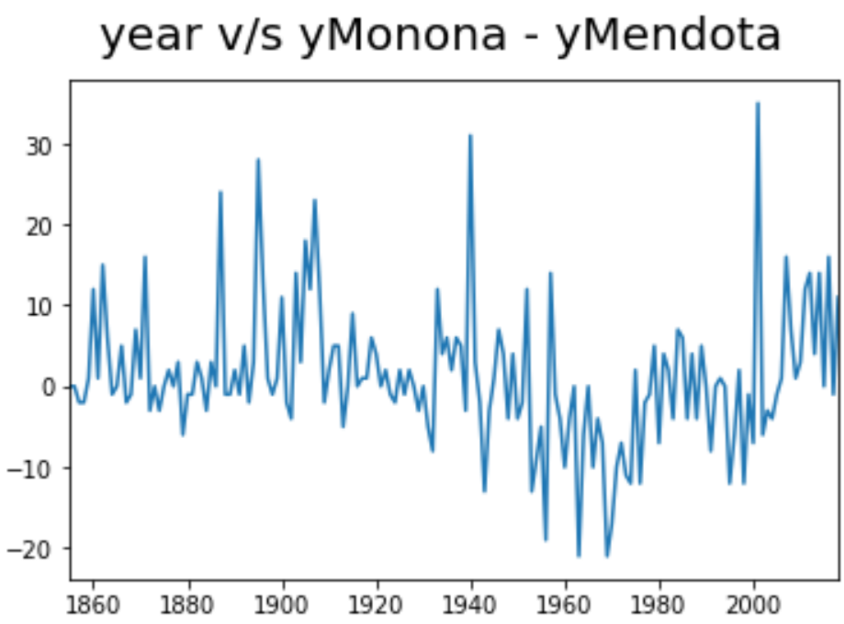

Fig: Plot of year v/s y_{Monana} - $y_{Mendota}$

2. Split the datasets: $x \le 1970$ as training, and $x > 1970$ as test. (Comment: due to the temporal nature this is NOT an iid split. But we will work with it.) On the training set, compute the sample mean $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ and the sample standard deviation $\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2}$ for the two lakes, respectively.

```
Lake Mendota
Mean 107.1896551724138
Standard Deviation 16.74666159754441
-----
Lake Monona
Mean 108.48275862068965
Standard Deviation 18.122521543826256
```

Fig: Sample Mean and sample deviations for lake Mendota and lake Monona

3. Using training sets, train a linear regression model

$$\hat{y}_{Mendota} = \beta_0 + \beta_1 x + \beta_2 y_{Monona}$$

to predict $y_{Mendota}$. Note: we are treating y_{Monona} as an observed feature. Do this by finding the closed-form MLE solution for $\beta = (\beta_0, \beta_1, \beta_2)^\top$ (no regularization):

$$\min_{\beta} \frac{1}{n}\sum_{i=1}^{n}(x_i^\top \beta - y_i)^2.$$

Give the MLE formula in matrix form (define your matrices), then give the MLE value of $\beta_0, \beta_1, \beta_2$.

$$\hat{\beta} = arg\min_{\beta} \frac{1}{n}\sum_{i=1}^{n}(x_i^\top \beta - y_i)^2$$

This can be represented in matrix notation form as,

$$\beta = arg\min_{\beta} \frac{1}{n}||\mathbf{X}^\top \beta - \mathbf{Y}||^2$$

where

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix}$$

,

$$\mathbf{X} = \begin{bmatrix} 1 & 1 & \dots & 1 \\ x_1 & x_2 & \dots & x_n \\ y_{mon_1} & y_{mon_2} & \dots & y_{mon_n} \end{bmatrix}$$

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

Let, $L = \frac{1}{n}||\mathbf{X}^\top \beta - \mathbf{Y}||^2$ be the loss function.
Find MLE of β by differentiating with respect to β and equating to zero.

$$\frac{\partial L}{\partial \beta} = \frac{2}{n} \frac{\partial[(\mathbf{X}^\top \beta - \mathbf{Y})^\top (\mathbf{X}^\top \beta - \mathbf{Y})]}{\partial \beta}$$

$$= \frac{2}{n} \frac{\partial[\beta^\top \mathbf{X}\mathbf{X}^\top \beta - \beta^\top \mathbf{X}\mathbf{Y} - \mathbf{Y}^\top \mathbf{X}^\top \beta + \mathbf{Y}^\top \mathbf{Y}]}{\partial \beta}$$

$$= \frac{2}{n}[2\mathbf{X}\mathbf{X}^\top \beta - \mathbf{X}\mathbf{Y} - \mathbf{Y}^\top \mathbf{X}^\top]$$

$$= \frac{2}{n}[2\mathbf{X}\mathbf{X}^\top \beta - 2\mathbf{X}\mathbf{Y}]$$

$$[2\mathbf{X}\mathbf{X}^\top \beta - 2\mathbf{X}\mathbf{Y}] = 0$$

$$\mathbf{X}\mathbf{X}^\top \beta = \mathbf{X}\mathbf{Y}$$

$$\beta = (\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{Y}$$

```
[[-6.41827663e+01]
 [ 4.12245664e-02]
 [ 8.52950638e-01]]
```

Fig: β from analytical solution

4. Using the MLE above, give the (1) mean squared error and (2) R^2 values on the Mendota test set. (You will need to use the Monona test data as observed features.)

$$R^2 = 1 - \frac{\sum_i (y_i - h(\vec{x_i}))^2}{\sum_i (y_i - \overline{y})^2}$$

Fig: R square formula used for calculation

```
mean square error 125.89537498517322
R squared error 0.709914254690508
```

Fig: Mean square and R^2 values for Mendota test set

5. “Reset” to Q3, but this time use gradient descent to learn the β’s. Recall our objective function is the mean squared error on the training set:

$$\frac{1}{n}\sum_{i=1}^{n}(x_i^\top \beta - y_i)^2.$$

Derive the gradient.

$$Loss = \frac{1}{n}\sum_{i=1}^{n}(\beta_0 + x_i\beta_1 + \beta_2 y_{monona} - y_i)^2$$

$$\frac{\partial L}{\partial \beta_0} = \frac{2}{n}\sum_{i=1}^{n}(\beta_0 + x_i\beta_1 + \beta_2 y_{monona} - y_i) * 1$$

$$\frac{\partial L}{\partial \beta_1} = \frac{2}{n}\sum_{i=1}^{n}(\beta_0 + x_i\beta_1 + \beta_2 y_{monona} - y_i) * x_i$$

$$\frac{\partial L}{\partial \beta_2} = \frac{2}{n}\sum_{i=1}^{n}(\beta_0 + x_i\beta_1 + \beta_2 y_{monona} - y_i) * y_{monona}$$

Compactly it can be written as,

$$\nabla L(\beta) = \frac{2}{n}\mathbf{X} * (\mathbf{X}^\top \beta - Y)$$

GD update will be $\beta_t = \beta_{t-1} - \eta * \nabla L(\beta)|_{\beta=\beta_{t-1}}$, t being the t^{th} iteration.

6. Implement gradient descent. Initialize $\beta_0 = \beta_1 = \beta_2 = 0$. Use a fixed stepsize parameter $\eta = 0.1$ and print the first 10 iteration's objective function value. Tell us if further iterations make your gradient descent converge, and if yes when; compare the β's to the closed-form solution. Try other η values and tell us what happens. **Hint:** Update $\beta_0, \beta_1, \beta_2$ simultaneously in an iteration. Don't use a new β_0 to calculate β_1, and so on.

```
Epoch: 001 | Objective: 6180350808297761.00000
Epoch: 002 | Objective: 333062678151436201782935552 0.00000
Epoch: 003 | Objective: 17948940285065871185752914798138821 18144.00000
Epoch: 004 | Objective: 96727876910417336587348324929806898446203310676 3776.00000
Epoch: 005 | Objective: 5212721209720435949265842972461314581193149066030645651911 47520.00000
Epoch: 006 | Objective: 2809165597156088516543526365145807499618188751145919517290430386971 77481216.00000
Epoch: 007 | Objective: 151387558143907265062140486337199351668721352086221462593079477236785839146894849212416.00000
Epoch: 008 | Objective: 815836303277264349581744890106133458396487766079846583430769837051263903150220208391533291393 51552.0000
0
Epoch: 009 | Objective: 439658900576500206559722098968400458987130117906296588940853205368663817124269007722034072311087406 3365
0528256.00000
Epoch: 010 | Objective: 236934723399337593739518677317937700744323214305196253070847909346910005624950083654230205554065759 7484
5831072762229161984.00000
```

Fig: For $\eta = 0.1$, the algorithm does not converge

It is observed that the algorithm diverges to infinity with increase in number of interations and hence corresponding β values are sub-optimal.

It is observed that there is a certain range of η values over which the algorithm converges. Above the range, it diverges and below the range,the algorithm converges extremely slowly, essentially appearing to stagnate. The Experiments are conducted as below. For $\eta = 0.00000001$, the algorithm appears to converge. Experiment was conducted with 2000 iterations. The Loss function appears to be decreasing with every iteration and it could take over 10000 iterations before it starts to appear to converge.

```
cost = train(model2,
             X, yMenTrain,
             num_epochs=2000,
             learning_rate=0.00000001)
```

```
Epoch: 001 | Objective: 10145.66786
Epoch: 002 | Objective: 8753.08021
Epoch: 003 | Objective: 7557.44788
Epoch: 004 | Objective: 6530.91517
Epoch: 005 | Objective: 5649.56607
Epoch: 006 | Objective: 4892.86704
Epoch: 007 | Objective: 4243.18862
Epoch: 008 | Objective: 3685.39473
Epoch: 009 | Objective: 3206.49000
Epoch: 010 | Objective: 2795.31701
Epoch: 011 | Objective: 2442.29636
```

```
Epoch: 1990 | Objective: 292.56467
Epoch: 1991 | Objective: 292.56132
Epoch: 1992 | Objective: 292.55797
Epoch: 1993 | Objective: 292.55463
Epoch: 1994 | Objective: 292.55128
Epoch: 1995 | Objective: 292.54793
Epoch: 1996 | Objective: 292.54458
Epoch: 1997 | Objective: 292.54124
Epoch: 1998 | Objective: 292.53789
Epoch: 1999 | Objective: 292.53454
Epoch: 2000 | Objective: 292.53119
```

Figure 8: Fig: $\eta = 0.00000001$

For $\eta = 0.00000009$ and number of iterations = 2000 the algorithm appears to converge faster.

```
cost = train(model3,
             X, yMenTrain,
             num_epochs=2000,
             learning_rate=0.00000009)
```

```
Epoch: 001 | Objective: 1619.63405
Epoch: 002 | Objective: 451.27075
Epoch: 003 | Objective: 316.73033
Epoch: 004 | Objective: 301.21334
Epoch: 005 | Objective: 299.39947
Epoch: 006 | Objective: 299.16323
Epoch: 007 | Objective: 299.10862
Epoch: 008 | Objective: 299.07493
Epoch: 009 | Objective: 299.04365
Epoch: 010 | Objective: 299.01265
Epoch: 011 | Objective: 298.98169
```

```
Epoch: 1990 | Objective: 244.89771
Epoch: 1991 | Objective: 244.87371
Epoch: 1992 | Objective: 244.84972
Epoch: 1993 | Objective: 244.82573
Epoch: 1994 | Objective: 244.80175
Epoch: 1995 | Objective: 244.77776
Epoch: 1996 | Objective: 244.75378
Epoch: 1997 | Objective: 244.72981
Epoch: 1998 | Objective: 244.70583
Epoch: 1999 | Objective: 244.68186
Epoch: 2000 | Objective: 244.65790
```

Figure 9: Fig: $\eta = 0.00000009$

Experiment was conducted with $\eta = 0.000000001, 0.0000000000001, 0.000000000009$. The loss function decreased slowly,taking too many iterations before it appears to converge, if not stagnate.

```
cost = train(model4,
             X, yMenTrain,
             num_epochs=100,
             learning_rate=0.0000000000001)
```

```
Epoch: 001 | Objective: 11767.63833
Epoch: 002 | Objective: 11767.62150
Epoch: 003 | Objective: 11767.60466
Epoch: 004 | Objective: 11767.58782
Epoch: 005 | Objective: 11767.57098
Epoch: 006 | Objective: 11767.55415
Epoch: 007 | Objective: 11767.53731
Epoch: 008 | Objective: 11767.52047
Epoch: 009 | Objective: 11767.50363
Epoch: 010 | Objective: 11767.48679
```

Fig: $\eta = 0.00000000000001$

```
cost = train(model4,
             X, yMenTrain,
             num_epochs=100,
             learning_rate=0.000000000009)
```

```
Epoch: 001 | Objective: 11766.13981
Epoch: 002 | Objective: 11764.62465
Epoch: 003 | Objective: 11763.10969
Epoch: 004 | Objective: 11761.59493
Epoch: 005 | Objective: 11760.08037
Epoch: 006 | Objective: 11758.56600
Epoch: 007 | Objective: 11757.05184
Epoch: 008 | Objective: 11755.53788
Epoch: 009 | Objective: 11754.02412
Epoch: 010 | Objective: 11752.51056
```

Fig: $\eta = 0.000000000009$

For higher values, it appears starts to diverge. Experiments were conducted with $\eta = 0.00009, 0.0001$

```
cost = train(model4,
             X, yMenTrain,
             num_epochs=100,
             learning_rate=0.00009)
```

```
Epoch: 001 | Objective: 4990955442.16160
Epoch: 002 | Objective: 2172036175091787.75000
Epoch: 003 | Objective: 945258168532183810048.00000
Epoch: 004 | Objective: 411371143548821926437715968.00000
Epoch: 005 | Objective: 179026453701472184168715439308800.00000
Epoch: 006 | Objective: 77911325642416111910545426945394343936.00000
Epoch: 007 | Objective: 33906579378937279650872019777978798385922048.00000
Epoch: 008 | Objective: 14755956411994163184964017936902154709038102740992.00000
Epoch: 009 | Objective: 6421710877975801895818172749013294380984783133797777408.00000
Epoch: 010 | Objective: 2794693169925112363795734614673064101058686243573265732730880.00000
```

Fig: $\eta = 0.00009$

```
cost = train(model4,
             X, yMenTrain,
             num_epochs=100,
             learning_rate=0.0001)
```

```
Epoch: 001 | Objective: 6163541503.94858
Epoch: 002 | Objective: 3312533871817740.00000
Epoch: 003 | Objective: 1780288348615921369088.00000
Epoch: 004 | Objective: 956798247765122483451068416.00000
Epoch: 005 | Objective: 514221692029907304462276202856448.00000
Epoch: 006 | Objective: 276363328603223468036066892070197919744.00000
Epoch: 007 | Objective: 148528719383956255980843445764859997439655936.00000
Epoch: 008 | Objective: 79825281426939311895970834626676230095924812054528.00000
Epoch: 009 | Objective: 42901302733364588172430848023529230267218273776464560128.00000
Epoch: 010 | Objective: 23056878013068419676187823104808742442547927767361808385441792.00000
```

Fig: $\eta = 0.0001$

7. As preprocessing, normalize your year and Monona features (but not $y_{Mendota}$). Then repeat Q6.

```
array([[107.18965517],
       [  1.38040755],
       [ 15.39084445]])
```

Fig:β for normalised data with $\eta = 0.1$ and 2000 iterations. The β values are obtained after 2000 itserations as the objective doesn't decrease further.The β values are far from the ones obtained from the analytical solution.

```
cost = train(modelNormalised1,
             X_norm, yMenTrain,
             num_epochs=2000,
             learning_rate=0.1)

Epoch: 001 | Objective: 7545.36373
Epoch: 002 | Objective: 4849.54746
Epoch: 003 | Objective: 3126.82634
Epoch: 004 | Objective: 2025.07152
Epoch: 005 | Objective: 1319.92599
Epoch: 006 | Objective: 868.28897
Epoch: 007 | Objective: 578.80499
Epoch: 008 | Objective: 393.10799
Epoch: 009 | Objective: 273.88331
Epoch: 010 | Objective: 197.26029
```

(a) First 10 iterations with normalized data.

```
Epoch: 1990 | Objective: 57.50964
Epoch: 1991 | Objective: 57.50964
Epoch: 1992 | Objective: 57.50964
Epoch: 1993 | Objective: 57.50964
Epoch: 1994 | Objective: 57.50964
Epoch: 1995 | Objective: 57.50964
Epoch: 1996 | Objective: 57.50964
Epoch: 1997 | Objective: 57.50964
Epoch: 1998 | Objective: 57.50964
Epoch: 1999 | Objective: 57.50964
Epoch: 2000 | Objective: 57.50964
```

(b) The objective value stagnates but has achieved a lower objective than the previous experiments

Figure 14: Normalised data with $\eta = 0.1$

```
cost = train(modelNormalised2,
             X_norm, yMenTrain,
             num_epochs=1000,
             learning_rate=0.5)

Epoch: 001 | Objective: 98.23538
Epoch: 002 | Objective: 65.03077
Epoch: 003 | Objective: 58.89862
Epoch: 004 | Objective: 57.76615
Epoch: 005 | Objective: 57.55701
Epoch: 006 | Objective: 57.51839
Epoch: 007 | Objective: 57.51125
Epoch: 008 | Objective: 57.50994
Epoch: 009 | Objective: 57.50969
Epoch: 010 | Objective: 57.50965
```

Fig:Experiment conducted with $\eta = 0.5$. The objective reduces drastically in the first few iterations and then stagnates.

```
cost = train(modelNormalised2,
             X_norm, yMenTrain,
             num_epochs=1000,
             learning_rate=0.01)
```

```
Epoch: 001 | Objective: 11302.93686
Epoch: 002 | Objective: 10856.72329
Epoch: 003 | Objective: 10428.27489
Epoch: 004 | Objective: 10016.88179
Epoch: 005 | Objective: 9621.86260
Epoch: 006 | Objective: 9242.56328
Epoch: 007 | Objective: 8878.35604
Epoch: 008 | Objective: 8528.63825
Epoch: 009 | Objective: 8192.83147
Epoch: 010 | Objective: 7870.38042
```

Fig:For $\eta = 0.01$, the objective decreases slowly and stagnates

```
cost = train(modelNormalised2,
             X_norm, yMenTrain,
             num_epochs=1000,
             learning_rate=0.9)
```

```
Epoch: 001 | Objective: 7758.36688
Epoch: 002 | Objective: 5623.93128
Epoch: 003 | Objective: 5199.48890
Epoch: 004 | Objective: 7259.15993
Epoch: 005 | Objective: 14349.70862
Epoch: 006 | Objective: 33180.08427
Epoch: 007 | Objective: 80619.72375
Epoch: 008 | Objective: 198602.66026
Epoch: 009 | Objective: 491063.69669
Epoch: 010 | Objective: 1215414.17427
```

Fig:For higher values such as $\eta = 0.9$, it starts to diverge

8. "Reset" to Q3 (no normalization, use closed-form solution), but train a linear regression model without using Monona:

$$\hat{y}_{Mendota} = \gamma_0 + \gamma_1 x.$$

(a) Interpret the sign of γ_1.

γ_1 is the coefficient corresponding to the year. The negative sign can be interpreted as a trend where the number of ice days reduces by 0.156 day on average with every passing year.

```
[[ 4.06111060e+02]
 [-1.56298774e-01]]
```

Fig: $\gamma = (\gamma_0, \gamma_1)^\top$

(b) Some analysts claim that because β_1 the closed-form solution in Q3 is positive, fixing all other factors, as the years go by the number of Mendota ice days will increase, namely the model in Q3 indicates a cooling trend. Discuss this viewpoint, relate it to question 8(a).

Signs of the coefficients indicate the direction of the relationship between a predictor variables and the prediction. When all else is constant, year influences the prediction positively. Hence with every passing year, the number of ice days increases and thus results in the cooling trend. It can be seen from 8a) that the negative sign for the coefficient implies that number of ice days reduces by 0.156 day on average with every passing year.

9. Of course, Weka has linear regression. Reset to Q3. Save the training data in .arff format for Weka. Use classifiers / functions / LinearRegression. Choose "Use training set." Bring up Linear Regression options, set "ridge" to 0 so it does not regularize. Run it and tell us the model: it is in the output in the form of "β_1 * year + β_2 * Monona + β_0."

$$0.0412 * year + 0.853 * Monona + -64.1828$$

Thus $\beta_0 = -64.1828$, $\beta_1 = 0.0412$, $\beta_2 = 0.853$

```
Linear Regression Model

IceDays =

      0.0412 * year +
      0.853  * Monona +
    -64.1828
```

Fig: Weka linear regression model results with $\lambda = 0$

10. Ridge regression.

 (a) Then set ridge to 1 and tell us the resulting Weka model.
 Linear + ridge Model is:

$$0.0387 * year + 0.8436 * Monona + -58.3961$$

```
IceDays =

      0.0387 * year +
      0.8436 * Monona +
    -58.3961
```

Fig: Weka linear regression model results with ridge set to 1, $\lambda = 1$

 (b) Meanwhile, derive the closed-form solution in matrix form for the ridge regression problem:

$$\min_{\beta} \left(\frac{1}{n} \sum_{i=1}^{n} (x_i^\top \beta - y_i)^2 \right) + \lambda \|\beta\|_A^2$$

where

$$\|\beta\|_A^2 := \beta^\top A \beta$$

and

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

This A matrix has the effect of NOT regularizing the bias β_0, which is standard practice in ridge regression. Note: Derive the closed-form solution, do not blindly copy lecture notes.
In matrix notation,our minimization problem is $\min_\beta (X^\top \beta - Y)^2 + \lambda \|\beta\|_2^2$.
From linear Algebra we know that,

$$||a||_2^2 + ||b||_2^2 = a^T a + b^T b = \left|\left|\begin{bmatrix} a \\ b \end{bmatrix}\right|\right|_2^2$$

Let us take $a = (\mathbf{X^T}\beta - Y)$ and $b = \sqrt{\lambda}\beta$
Thus, the minimization problem becomes $\min_\beta \left|\left|\begin{bmatrix} \mathbf{X^T} \\ \sqrt{\lambda} \end{bmatrix}\beta - \begin{bmatrix} Y \\ 0 \end{bmatrix}\right|\right|_2^2$. This is like minimizing $||\tilde{A}\beta - \tilde{Y}||_2^2$ whose solution is $(\tilde{\mathbf{A}^T}\tilde{\mathbf{A}})^{-1}\tilde{\mathbf{A}}^T\tilde{Y}$ where $\tilde{\mathbf{A}} = \begin{bmatrix} \mathbf{X^T} \\ \sqrt{\lambda} \end{bmatrix}$ and $\tilde{Y} = \begin{bmatrix} Y \\ 0 \end{bmatrix}$.
Thus,

$$\beta = \left(\begin{bmatrix} \mathbf{X} & \sqrt{\lambda} I \end{bmatrix} \begin{bmatrix} \mathbf{X^T} \\ \sqrt{\lambda} I \end{bmatrix} \right)^{-1} \mathbf{X}Y = (\mathbf{X}\mathbf{X^T} + \lambda I)^{-1}\mathbf{X}Y$$

(c) Let $\lambda = 1$ and tell us the value of β from your ridge regression model.
$\beta_0 = -58.3961, \beta_1 = 0.0387, \beta_2 = 0.8436$

Extra Credit: Multinomial Naïve Bayes [10 pts]

Consider the Multinomial Naïve Bayes model. For each point $(\mathbf{x}, y)$, $y \in \{0, 1\}$, $\mathbf{x} = (x_1, x_2, \ldots, x_M)$ where each x_j is an integer from $\{1, 2, \ldots, K\}$ for $1 \le j \le M$. Here K and M are two fixed integer.
Suppose we have N data points $\{(\mathbf{x}^{(i)}, y^{(i)}) : 1 \le i \le N\}$, generated as follows.

for $i \in \{1, \ldots, N\}$:
  $y^{(i)} \sim$ Bernoulli(ϕ)
  for $j \in \{1, \ldots, M\}$:
    $x_j^{(i)} \sim$ Multinomial$(\theta_{y^{(i)}}, 1)$

Here $\phi \in \mathbb{R}$ and $\theta_k \in \mathbb{R}^K (k \in \{0, 1\}$ are parameters. Note that $\sum_l \theta_{k,l} = 1$ since they are the parameters of a multinomial distribution.
Derive the formula for estimating the parameters ϕ and θ_k, as we have done in the lecture for the Bernoulli Naïve Bayes model. Show the steps.
Since $y^{(i)}$ is Bernoulli(ϕ), If $y^{(i)} = 0$, then $x_j^{(i)}$ is Multinomial$(\theta_0, 1)$ and if $y^{(i)} = 1$, then $x_j^{(i)}$ is Multinomial$(\theta_1, 1)$
$P(y^{(i)}) = (\phi)^{y^{(i)}}(1-\phi)^{(1-y^{(i)})}$

$$P(Y) = \prod_{i=1}^{N} (\phi)^{y^{(i)}}(1-\phi)^{(1-y^{(i)})}$$

$$l(\phi) = \prod_{i=1}^{N} (\phi)^{y^{(i)}}(1-\phi)^{(1-y^{(i)})}$$

$$log(l(\phi)) = \sum_{i=1}^{N} y^{(i)} log(\phi) + \sum_{i=1}^{N} (1 - y^{(i)}) log(1-\phi)$$

Differentiating wrt ϕ and solving for it results in:

$$\frac{\partial}{\partial\phi} log(l(\phi)) = \sum_{i=1}^{N} \frac{y^{(i)}}{\phi} + \sum_{i=1}^{N} (1 - y^{(i)}) \frac{-1}{(1-\phi)}$$

$$\frac{\partial}{\partial\phi} log(l(\phi)) = \sum_{i=1}^{N} \frac{y^{(i)}}{\phi} + \sum_{i=1}^{N} (y^{(i)} - 1) \frac{1}{(1-\phi)}$$

Setting $\frac{\partial}{\partial\phi} log(l(\phi)) = 0$, and solving for ϕ, we get,

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

Thus $\phi_{MLE} = \frac{\sum_{i=1}^{N} y^{(i)}}{N}$
Now, $P(X_k|Y)$ is Multinomial$(\phi_Y, 1)$. Hence

$$P(X_k|Y) = \prod_{j=1}^{M_i} \theta_{y,x_j}$$

Thus, $p_{\phi,\theta} = (\phi)^y(1-\phi)^{(1-y)} \prod_{j=1}^{M_i} \theta_{y,x_j}$
To find $P(Y)$, find class conditional MLE by,
$\phi_{MLE} = \frac{\sum_{i=1}^{N} 1(y^{(i)}=1)}{N}$

$$l(\theta_{yj}) = \prod_{j=1}^{M_i} \theta_{yj}^{x_j}$$

$$logl(\theta_{yj}) = \sum_{j=1}^{M_i} x_j log(\theta_{yj})$$

, Using The fact that $\sum_l \theta_{k,l} = 1$ and Lagrange Multiplier λ,
$logl(\theta_{yj}) = \sum_{j=1}^{M_i} x_j log(\theta_{yj}) + \lambda \left(1 - \sum_j \theta_{y,j} \right)$

$$\frac{\partial}{\partial\theta_{yj}} logl(\theta_{yj}) = \sum_{j=1}^{M_i} \frac{x_j}{\theta_{yj}} + \lambda \frac{\partial}{\partial\theta_{yj}} \left(1 - \sum_j \theta_{yj} \right) = 0$$

Thus,

$$\frac{x_j}{\theta_{y,j}} - \lambda = 0$$
$$\theta_{y,j} = \frac{x_j}{\lambda}$$

and since $\sum_{j=1}^{M_i} \theta_{y,j} = 1$, $\sum_{j=1}^{M_i} x_j = \lambda$
Thus, $\theta_{y,j} = \frac{x_j}{\lambda} = \frac{x_j}{\sum_{j=1}^{M_i}} x_j$

Extra Credit: Logistic Regression [10 pts]

(1) Suppose for each class $i \in \{1, \ldots, K\}$, the class-conditional density $p(\mathbf{x}|y = i)$ is normal with mean $\mu_i \in \mathbb{R}^d$ and the same covariance $\mathbf{\Sigma} \in \mathbb{R}^{d \times d}$:

$$p(\mathbf{x}|y = i) = N(\mathbf{x}|\mu_i, \mathbf{\Sigma}).$$

Compute $p(y = i|\mathbf{x})$. Can it be represented as a softmax over a linear transformation of $\mathbf{x}$? Show the calculation steps.

Since $p(x) =$, and $p(x|y=i) =$ Using the fact that $\mathbf{\Sigma}$ is same, it can be cancelled out in Nr and Dr.

$$\begin{aligned}
p(y=i|\mathbf{x}) &= \frac{p(x|y=i)p(y=i)}{p(x)} \\
&= \frac{p(x|y=i)p(y=i)}{\sum_{i=1}^{K} p(x|y=i)p(y=i)} \\
&= \frac{exp\{-0.5(x-\mu_i)^\top \Sigma^{-1}(x-\mu_i)\}exp\{log(p(y=i))\}}{\sum_{i=1}^{K} exp\{-0.5(x-\mu_i)^\top \Sigma^{-1}(x-\mu_i)\}exp\{log(p(y=i)))\}} \\
&= \frac{exp\{-0.5(x^T\Sigma^{-1}x)\}exp\{-0.5(-x^T\Sigma^{-1}\mu_i - \mu_i^T\Sigma^{-1}x + \mu_i^T\Sigma^{-1}\mu_i) + log(p(y=i))\}}{exp\{-0.5(x^T\Sigma^{-1}x)\}\sum_{i=1}^{K} exp\{-0.5(-x^T\Sigma^{-1}\mu_i - \mu_i^T\Sigma^{-1}x + \mu_i^T\Sigma^{-1}\mu_i) + log(p(y=i))\}} \\
&= \frac{exp\{0.5x^T\Sigma^{-1}\mu_i + 0.5\mu_i^T\Sigma^{-1}x - (0.5\mu_i^T\Sigma^{-1}\mu_i - log(p(y=i)))\}}{\sum_{i=1}^{K} exp\{0.5x^T\Sigma^{-1}\mu_i + 0.5\mu_i^T\Sigma^{-1}x - (0.5\mu_i^T\Sigma^{-1}\mu_i - log(p(y=i)))\}}
\end{aligned} \tag{1}$$

By appropriately choosing $\mathbf{W}$ and $\mathbf{X}$, this can be represented as $\frac{exp\{\mathbf{WX}\}}{\sum_{i=1}^{K} exp\{\mathbf{WX}\}}$ which is the softmax function. It can be seen that $\mathbf{WX}$ is a linear transformation of $\mathbf{X}$¿ Hence it is possible to represent $p(y=i|\mathbf{x})$ as a softmax over a linear transformation of $\mathbf{x}$.

(2) Suppose $p(\mathbf{x}|y=i)$ has different covariances $\mathbf{\Sigma}_i \in \mathbb{R}^{d\times d}$:

$$p(\mathbf{x}|y=i) = N(\mathbf{x}|\mu_i, \mathbf{\Sigma}_i).$$

Again, compute $p(y=i|\mathbf{x})$. Can it be represented as a softmax over a linear transformation of $\mathbf{x}$? Show the calculation steps.

$$\begin{aligned}
p(y=i|\mathbf{x}) &= \frac{p(x|y=i)p(y=i)}{\sum_{i=1}^{K} p(x|y=i)p(y=i)} \\
&= \frac{exp\{-0.5(x-\mu_i)^\top \Sigma_i^{-1}(x-\mu_i)\}exp\{log(p(y=i))\}}{\sum_{i=1}^{K} exp\{-0.5(x-\mu_i)^\top \Sigma_i^{-1}(x-\mu_i)\}exp\{log(p(y=i)))\}} \\
&= \frac{exp\{-0.5(x^T\Sigma_i^{-1}x)\}exp\{-0.5(-x^T\Sigma^{-1}\mu_i - \mu_i^T\Sigma^{-1}x + \mu_i^T\Sigma_i^{-1}\mu_i) + log(p(y=i))\}}{\sum_{i=1}^{K} exp\{-0.5(x^T\Sigma_i^{-1}x)\}exp\{-0.5(-x^T\Sigma_i^{-1}\mu_i - \mu_i^T\Sigma_i^{-1}x + \mu_i^T\Sigma_i^{-1}\mu_i) + log(p(y=i))\}}
\end{aligned} \tag{2}$$

Since the quadratic term is dependent on i, it cannot be factored out and cancelled. And hence $p(y=i|\mathbf{x})$ can no longer be expressed as a linear transformation of $\mathbf{x}$.