# Problem Set #2

Joe Michail
DATA SCI 423 – Machine Learning
Northwestern University

April 25, 2020

## Question 4.1

### Part A

Let $\mathbf{z} = \mathbb{R}^{N\times 1}$ and $\mathbf{C} = \mathbb{R}^{N\times N}$. For a symmetric matrix, one can decompose it via eigenvalue decomposition $\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$. Here $\mathbf{Q}$ is an $N \times N$ matrix with the eigenvectors of $\mathbf{A}$ making the columns and $\mathbf{\Lambda}$ is an $N \times N$ matrix where the eigenvalues are on the diagonal.

$$\mathbf{z}^T\mathbf{C}\mathbf{z} \implies \mathbf{z}^T\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T\mathbf{z}.$$

If we define $\mathbf{V} \equiv \mathbf{z}^T\mathbf{Q}$, then $\mathbf{V}^T = \mathbf{Q}^T\mathbf{z}$:

$$\mathbf{V}\mathbf{\Lambda}\mathbf{V}^T. \tag{1}$$

Writing this in summation notation, we have:

$$\sum_{i,j} V_i \Lambda_{ij} V_j.$$

Since the eigenvalues only lie on the diagonal, indices where $i \neq j$ will not contribute to the sum, so it can be rewritten as:

$$\sum_{i=1}^{N} V_i \Lambda_{ii} V_i,$$
$$\sum_{i=1}^{N} V_i^2 \Lambda_{ii}.$$

Finally, inputting the condition that $\Lambda_{ii} \geq 0$ shows $\mathbf{z}^T\mathbf{C}\mathbf{z} \geq 0$ must be true since all $V_i^2$ are positive.

## Part B

Starting from the condition that $\mathbf{z}^T\mathbf{C}\mathbf{z} \geq 0$, one can show that equation (1) above is:

$$\mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T \geq 0.$$

Multiplying on the left by $\mathbf{V}^T$ and on the right by $\mathbf{V}$ yields:

$$\mathbf{V}^T\mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T\mathbf{V} \geq \mathbf{0}, \tag{2}$$

where both the left and right sides are now $N \times N$ matrices. The elements of a diagonal matrix are the eigenvalues of said matrix; therefore the determinant of a diagonal matrix is a product of the eigenvalues. In addition, the determinant of a product of matrices is the product of individual determinants:

$$\det(\mathbf{ABC}) = \det(\mathbf{A})\det(\mathbf{B})\det(\mathbf{C}).$$

Taking the determinant of equation (2):

$$\begin{aligned} \det(\mathbf{V}^T\mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T\mathbf{V}) &\geq \det(\mathbf{0}) \\ \det(\mathbf{V}^T\mathbf{V})\det(\boldsymbol{\Lambda})\det(\mathbf{V}^T\mathbf{V}) &\geq 0 \\ \det(\mathbf{V}^T\mathbf{V})^2\prod_{i=1}^{N}\lambda_{ii} &\geq 0. \end{aligned}$$

Since anything squared is positive, this implies:

$$\prod_{i=1}^{N}\lambda_{ii} \geq 0,$$

which holds for **ANY** N, odd or even. In the case of an odd N, a negative eigenvalue would ruin the inequality by making the entire product negative. Therefore, for this inequality to hold, **ALL** eigenvalues of $\mathbf{C}$ must be positive.

## Part C

The quadratic cost function is given by:

$$g(\mathbf{w}) = a + \mathbf{b}^T\mathbf{w} + \mathbf{w}^T\mathbf{C}\mathbf{w}.$$

The Hessian of this matrix is given by:

$$\nabla_{\mathbf{w}}g = 2\mathbf{C}.$$

$\mathbf{C}$ given in the problem is:

$$\mathbf{C} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}.$$

From the trace of the matrix, we find the sum of the eigenvalues is $\Sigma\lambda_i = 2$ and from the determinant we find the product of the matrix is $\Pi\lambda_i = 0$. Therefore, the two eigenvalues of $\mathbf{C}$ are $0, 2$. Since both are non-negative, this implies $\mathbf{w}^T\mathbf{C}\mathbf{w} \geq 0$, so this cost function is concave up for all points, which is expected for a quadratic cost function.

### Part D

Again, begin by defining $\mathbf{A} \equiv \mathbf{C} + \lambda\mathbf{I}$. Since both $\mathbf{C}$ and $\mathbf{I}$ are symmetric, their sum will be as well. Therefore, we can do an eigenvalue decomposition on $\mathbf{A}$:

$$\mathbf{A} = \mathbf{C} + \lambda\mathbf{I} = \mathbf{V}^T\mathbf{\Lambda}\mathbf{V}.$$

Taking the determinant of both sides:

$$\det(\mathbf{C} + \lambda\mathbf{I}) = \det(\mathbf{V}^T\mathbf{\Lambda}\mathbf{V}).$$

The determinant of a matrix product is the product of individual determinants, so:

$$\det(\mathbf{C} + \lambda\mathbf{I}) = \det\mathbf{V}^T \det\mathbf{\Lambda} \det\mathbf{V}.$$

The matrix $\mathbf{V}$ is orthogonal since it is composed of eigenvectors, so $\mathbf{V}^T = \mathbf{V}^{-1}$:

$$\det(\mathbf{C} + \lambda\mathbf{I}) = \det\mathbf{V}^{-1} \det\mathbf{\Lambda} \det\mathbf{V} = \det\mathbf{\Lambda} = \prod_i \lambda_i.$$

If we make the condition that the product of eigenvalues must be positive, but make no mention of the dimension of $\mathbf{C}$ initially, this places a constraint on the signs of each individual eigenvalue. In this case, all the eigenvalues must be positive for the product to always be positive. Therefore, given a large enough value of $\lambda$ will yield positive eigenvalues.
The smallest value of $\lambda$ required to make all the eigenvalues of $\mathbf{C}$ positive is $\lambda = \min \lambda_C$.

## Question 4.2

### Part A

Start by defining $\mathbf{A} \equiv \mathbf{x}\mathbf{x}^T, \mathbf{x} \in \mathbb{R}^{N\times 1}$. From problem 1 above, we know that since $\mathbf{A}$ is square, it can be split via eigenvalue decomposition:

$$\mathbf{x}\mathbf{x}^T = \mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T.$$

Multiplying on the right by $\mathbf{x}$ and on the left by $\mathbf{x}^T$ yields:

$$\mathbf{x}^T\mathbf{V}\mathbf{\Lambda}\mathbf{V}^T\mathbf{x} = \mathbf{x}^T\mathbf{x}\mathbf{x}^T\mathbf{x} = ||x||_2^4.$$

Define $\mathbf{Q} = \mathbf{x}^T\mathbf{V}$:

$$\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T = ||x||_2^4.$$

Converting this to summation notation:

$$\sum_{ij} Q_i \Lambda_{ij} Q_j = ||x||_2^4.$$

Since $\mathbf{\Lambda}$ is a diagonal matrix, only the terms where $i = j$ will survive:

$$\sum_i Q_i \Lambda_{ii} Q_i = \sum_i Q_i^2 \Lambda_{ii} = ||x||_2^4.$$

The right hand side must always be positive and $Q_i^2$ is as well. Since we cannot be guaranteed that $N$ will be even, all the eigenvalues $\lambda_i = \Lambda_{ii}$ must be individually positive so that one negative eigenvalue would not be able to break the equality.

## Part B

Begin by defining $\mathbf{A} = \sum_p \delta_p \mathbf{x}_p \mathbf{x}_p^T$. Taking the trace of both sides:

$$\mathrm{Tr}\left(\sum_p \delta_p \mathbf{x}_p \mathbf{x}_p^T\right) = \mathrm{Tr}(\mathbf{A}) = \sum_i \lambda_i^A,$$

where $\lambda_i^A$ are the eigenvalues of $\mathbf{A}$. The trace of a matrix sum is the sum of traces, so:

$$\sum_p \mathrm{Tr}\left(\delta_p \mathbf{x}_p \mathbf{x}_p^T\right) = \sum_i \lambda_i^A.$$

Since $\delta_p$ is a number, it can be pulled out the trace. Using a trace identity: $\mathrm{Tr}(\mathbf{a}\mathbf{a}^T) = \mathbf{a}^T\mathbf{a} = ||a||_2^2$:

$$\sum_p \delta_p ||x_p||_2^2 = \sum_i^N \lambda_i^A.$$

Since both terms under the left sum are positive and we're not guaranteed an even $N$, all the eigenvalues $\lambda_i^A$ must be individually non-negative.

## Part C

As above, begin by defining $\mathbf{A} = \sum_p \delta_p \mathbf{x}_p \mathbf{x}_p^T + \lambda\mathbf{I}$. Taking the trace of each side:

$$\mathrm{Tr}\left(\sum_p \delta_p \mathbf{x}_p \mathbf{x}_p^T + \lambda\mathbf{I}\right) = \mathrm{Tr}(\mathbf{A}).$$

The trace of a sum of matrices is the sum of the traces:

$$\mathrm{Tr}\left(\sum_p \delta_p \mathbf{x}_p \mathbf{x}_p^T\right) + \mathrm{Tr}\left(\lambda\mathbf{I}\right) = \mathrm{Tr}(\mathbf{A}).$$

From above, we know that the trace of the first sum is $\sum_p \delta_p ||x_p||_2^2$. The trace of $\lambda\mathbf{I} = \lambda N$, where $N$ is the dimension of the identity, and the trace of $\mathbf{A}$ is the sum of the eigenvalues. Therefore, we find:

$$\sum_p \delta_p ||x_p||_2^2 + \lambda N = \sum_i \lambda_i^A.$$

Since $\delta_p$, $||x_p||_2^2$ and $\lambda \geq 0$, the sum of eigenvalues must also be non-negative. Because we're not guaranteed an even $N$ and a negative eigenvalue could break the inequality, it follows that the eigenvalues of $\mathbf{A}$ must all be independently non-negative.

## Question 4.5

### Part A

Given $g(\mathbf{w}) = \log(1 + \exp(\mathbf{w}^T\mathbf{w}))$, determine the stationary points using the first order condition.

$$\begin{aligned}
\nabla_{\mathbf{w}} g &= \nabla(\log(1 + \exp(w_1^2 + w_2^2))) \\
&= \left[\frac{\partial g}{\partial w_1}, \frac{\partial g}{\partial w_2}\right]^T \\
&= \frac{2\exp(w_1^2 + w_2^2)}{\exp(w_1^2 + w_2^2) + 1}\,[w_1, w_2]^T
\end{aligned}$$

The stationary point occurs where $\nabla g = \mathbf{0}$, therefore the stationary point occurs at the origin:

$$\boxed{\mathbf{w}^* = \mathbf{0}}$$

.

### Part B

The second-order definition of convexity requires computing the Hessian:

$$\nabla^2 g = \begin{bmatrix} \dfrac{\partial^2 g}{\partial w_1^2} & \dfrac{\partial^2 g}{\partial w_1 w_2} \\ \dfrac{\partial^2 g}{\partial w_1 w_2} & \dfrac{\partial^2 g}{\partial w_2^2} \end{bmatrix}.$$

Computing the second derivatives:

$$\frac{\partial^2 g}{\partial w_1^2} = \frac{2w_1^2 + 1 + \exp(w_1^2 + w_2^2)}{2\cosh^2\left(\dfrac{w_1^2 + w_2^2}{2}\right)}$$

$$\frac{\partial^2 g}{\partial w_2^2} = \frac{2w_2^2 + 1 + \exp(w_1^2 + w_2^2)}{2\cosh^2\left(\dfrac{w_1^2 + w_2^2}{2}\right)}$$

$$\frac{\partial^2 g}{\partial w_1 \partial w_2} = \frac{w_1 w_2}{2\cosh^2\left(\dfrac{w_1^2 + w_2^2}{2}\right)}$$

Evaluating the Hessian at $\mathbf{w}^* = \mathbf{0}$ yields:

$$\nabla^2 g|_{\mathbf{w}^*} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Since the determinant of the Hessian is positive and the diagonal terms are positive, $\mathbf{w}^*$ is a minimum.

## Part C

See attached pages for this plot.

## Part D

See attached pages for this plot. Calculating the Hessian at $\mathbf{w} = [4,4]^T$ yields:

$$\nabla^2 g|_{[4,4]^T} \approx \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

whereas at $\mathbf{w} = [1,1]^T$:

$$\nabla^2 g|_{[1,1]^T} = \begin{bmatrix} 2.18 & 0.42 \\ 0.42 & 2.18 \end{bmatrix}.$$

In the case where we start at $[4,4]^T$, the steps taken are only in the $w_1$ and $w_2$ directions. In the initial $[1,1]^T$ spot, the off-diagonal terms show that we're zigzagging around the minimum initially, which takes longer to find it.

# Question 5.2

## Part A

See attached scripts for plots.

## Part B

Begin with the log fit:

$$w_0 + w_1 \log(x_p) \approx \log(y_p).$$

Exponentiating both sides to $e$:

$$\begin{aligned} \exp(\log(y_p)) &= \exp(w_0 + w_1 \log(x_p)) \\ y_p &= \exp(w_0) x_p^{w_1} \end{aligned}$$

Plugging in $(w_0, w_1) = (6.81, 0.65)$ from the linear regression yields:

$$\boxed{y_p(\text{kJ/day}) \approx 906.9 x_p(\text{kg})^{0.65}}$$

### Part C

A 10 kg animal requires about 4051 kJ per day, which yields a calorie rate of about 968212 calories or 968 kcal in more familiar units.

## Question 5.9

### Boston Housing

I was not able to verify either of the metrics for the Boston data set as stated in the book. However, as the professor mentioned, these values are likely not correct and are probably typos. For the RMSE, I got 4.7 and for the MAD I got 3.1. Since the MAD is less than the RMSE, there is probably a lot of scatter or outliers in the data.

### Automobile MPGs

I was able to verify the RMSE and MAD values for this dataset.

# HW 2

April 24, 2020

```
[1]: import numpy as np
     import matplotlib.pyplot as plt
     import scipy.stats as s
     import scipy.optimize as o
```

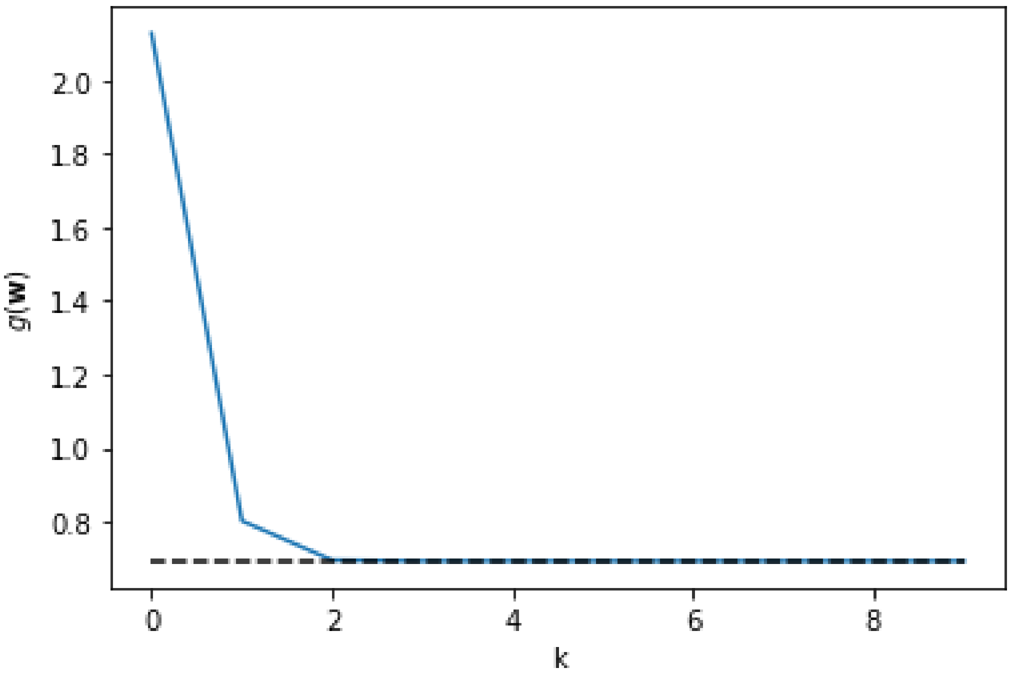
## 1 Problem 4.5

```
[2]: def g(w):
         return np.log(1 + np.exp(np.sum(w**2.0)))

     def grad(w):
         return 2 * np.exp(np.sum(w**2.0)) / (np.exp(np.sum(w**2.0)) + 1) * w

     def hessian(w):
         d2w1 = (2 * w[0]**2.0 + 1 + np.exp(np.sum(w**2.0))) / (2.0 * (np.cosh(np.
      ↪sum(w**2.0) / 2))**2.0)
         d2w2 = (2 * w[1]**2.0 + 1 + np.exp(np.sum(w**2.0))) / (2.0 * (np.cosh(np.
      ↪sum(w**2.0) / 2))**2.0)
         d2w1w2 = (w[0] * w[1]) / (np.cosh(np.sum(w**2.0) / 2))**2.0

         hess = np.zeros((2, 2))
         hess[0, 0], hess[1, 1] = d2w1, d2w2
         hess[0, 1], hess[1, 0] = d2w1w2, d2w1w2

         return hess

     def newton(w, steps):
         cost = []

         for i in range(steps):
             cost.append(g(w))

             step = np.matmul(np.linalg.inv(hessian(w)), grad(w))

             w = w - step
```

```
    return cost
```

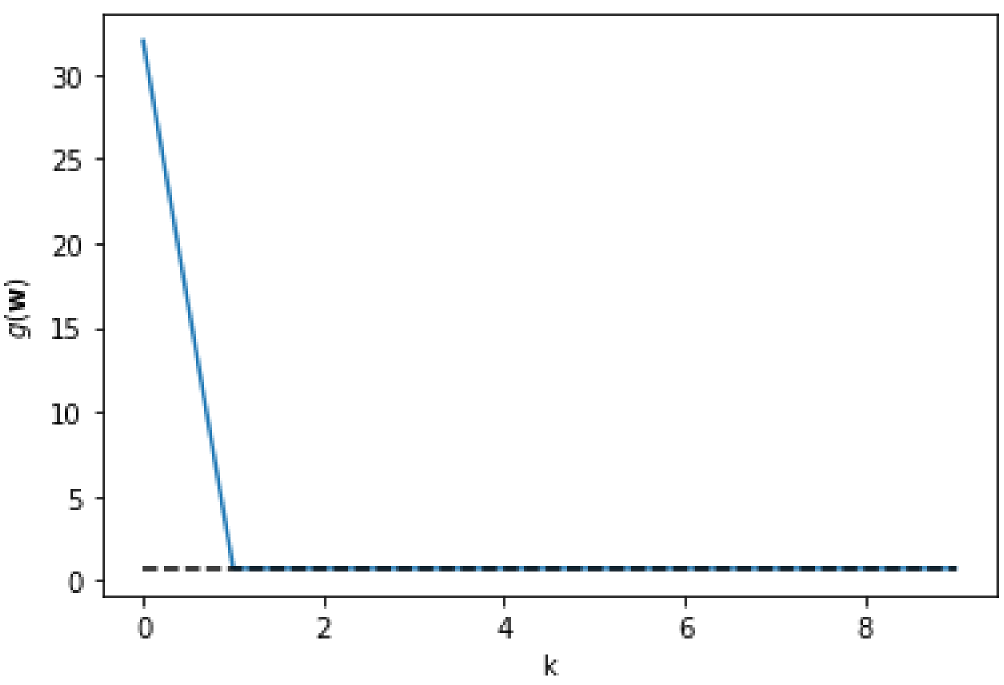
[3]:
```
#Part C
w0 = np.ones((2, 1))
cost = newton(w0, 10)

plt.plot(cost)
plt.plot(range(10), np.log(2)*np.ones((10)), 'k--')
plt.xlabel("k")
plt.ylabel("$g(\\mathbf{w})$")
```

[3]:
```
Text(0, 0.5, '$g(\\mathbf{w})$')
```



[4]:
```
#Part D
w0 = 4*np.ones((2, 1))
cost = newton(w0, 10)

plt.plot(cost)
plt.plot(range(10), np.log(2)*np.ones((10)), 'k--')
plt.xlabel("k")
plt.ylabel("$g(\\mathbf{w})$")
```

[4]:
```
Text(0, 0.5, '$g(\\mathbf{w})$')
```



## 2 Problem 5.2

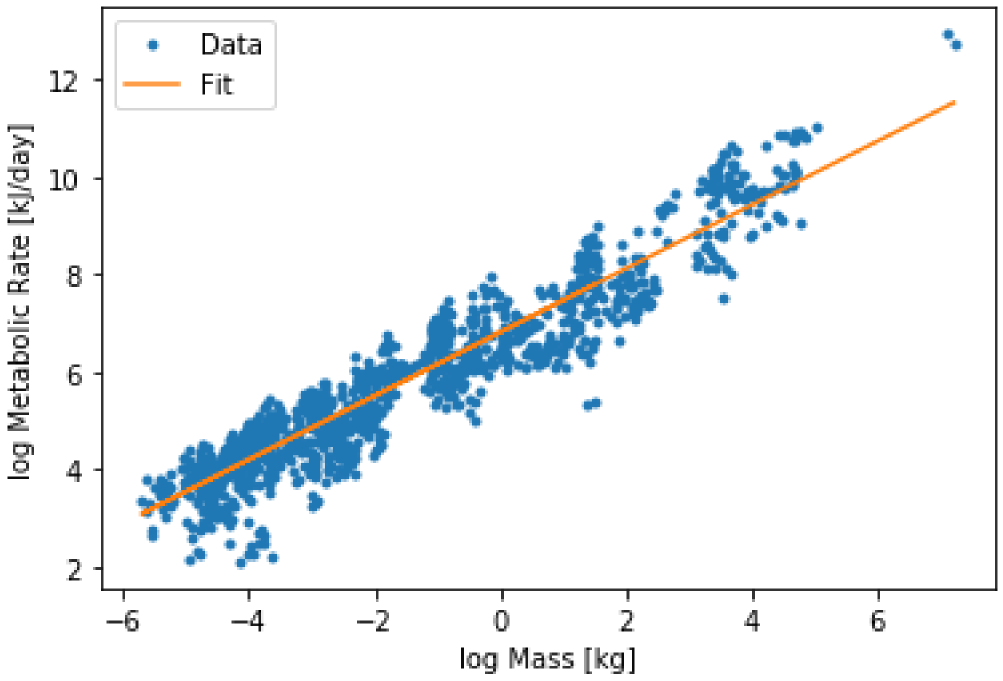
```
[5]: kleiber = np.genfromtxt("kleiber.csv", delimiter=',')

     #Fix first mass for some reason
     kleiber[0, 0] = 1370
```

```
[6]: #Part A
     w1, w0, _, _, _ = s.linregress(np.log(kleiber[0, :]), np.log(kleiber[1, :]))
     print("w0 = %0.2f, w1 = %0.2f" % (w0, w1))
```

```
w0 = 6.81, w1 = 0.65
```

```
[7]: plt.plot(np.log(kleiber[0, :]), np.log(kleiber[1, :]), '.', label="Data")
     plt.plot(np.log(kleiber[0, :]), w1 * np.log(kleiber[0, :]) + w0, label="Fit")
     plt.xlabel("log Mass [kg]")
     plt.ylabel("log Metabolic Rate [kJ/day]")
     plt.legend()
```

```
[7]: <matplotlib.legend.Legend at 0x1e48a2a6308>
```



## 3 Problem 5.9

[8]:
```python
def RMSE(w, x, y):
    cost = 0
    for p in range(y.size):
        x_p = x[:, p]
        y_p = y[p]

        a = w[0] + np.dot(x_p.T, w[1:])
        cost += (a.T - y_p)**2.0

    return np.sqrt(cost / float(y.size))

def MAD(w, x, y):
    cost = 0
    for p in range(y.size):
        x_p = x[:, p][:, np.newaxis]
        y_p = y[p]

        a = w[0] + np.dot(x_p.T, w[1:])
        cost += np.abs(a.T - y_p)

    return cost / float(y.size)
```

[9]:
```python
#Boston Data: Ex. 5.5
boston = np.genfromtxt("boston_housing.csv", delimiter=',')
x, y = boston[:-1, :], boston[-1:, :].flatten()

print("Input x shape: ", x.shape)
print("Input y shape: ", y.shape)

#Calculating means and standard deviations in rows
means = np.nanmean(x, axis=1)
stds  = np.nanstd(x, axis=1)

#Now normalize the data
transformed = np.zeros(x.shape)

for i in range(means.size):
    transformed[i, :] = (x[i, :] - means[i]) / stds[i]

#Run the conjugate gradient method, find minimum weights and print out cost
print("RMSE: %0.1f" % o.minimize(RMSE, np.ones(14), args=(transformed, y),␣
 ↪method='CG')['fun'])
print("MAD: %0.1f" %  o.minimize(MAD,  np.ones(14), args=(transformed, y),␣
 ↪method='CG')['fun'])
```

```
Input x shape:  (13, 506)
Input y shape:  (506,)
RMSE: 4.7
MAD: 3.1
```

[10]:
```python
#Automobile Data: Ex 5.6
auto = np.genfromtxt("auto_data.csv", delimiter=',')
x, y = auto[:-1, :], auto[-1:, :]

print("Input x shape: ", x.shape)
print("Input y shape: ", y.shape)

#Calculating means and standard deviations in rows
means = np.nanmean(x, axis=1)
stds  = np.nanstd(x, axis=1)

#Now normalize the data
transformed = np.zeros(x.shape)

for i in range(means.size):
    transformed[i, :] = (x[i, :] - means[i]) / stds[i]

#Remove the 6 NaN values in the x-data
indices = np.any(np.isnan(transformed), axis=0)
```

```
transformed = transformed[:, ~indices]
y = y.flatten()[~indices]

#Run the conjugate gradient method, find minimum weights and print out cost
print("RMSE: %0.1f" % o.minimize(RMSE, np.ones(8), args=(transformed, y),␣
 ↪method='CG')['fun'])
print("MAD: %0.1f"  % o.minimize(MAD,  np.ones(8), args=(transformed, y),␣
 ↪method='CG')['fun'])
```

```
Input x shape:  (7, 399)
Input y shape:  (1, 399)
RMSE: 3.3
MAD: 2.4
```

[ ]:

```

```