Introduction to Machine Learning
Homework 2: Multiple Linear Regression

Prof. Sundeep Rangan

1. There is no one single correct answer to this problem. Below are possible ideas.

   (a) A possible target variable, y, is the total sales (in some units like dollars or units) for the product.

   (b) One way the features can be represented is to let x_1 be the numeric score and $x_2, x_3, \ldots, x_k$ be the frequency of occurrence for each of the $k-1$ words. The linear model could then be

   $$y = \beta_1 x_1 + \ldots + \beta_k x_k + \epsilon.$$

   (c) One way is to normalize the scores so that $x_1 = \text{score}/5$ when the score is out of five and $x_1 = \text{score}/10$ when it is out of ten.

   (d) For this case, you could use a variation of one-hot coding. Specifically, we replace the single variable x_1 with four variables $x_1, \ldots, x_4$ using the following encoding:

Review type	x_1	x_2	x_3	x_4
Score available with score $s = 1, \ldots, 5$	s	0	0	0
Good rating	0	1	0	0
Bad rating	0	0	1	0
No rating	0	0	0	1

   Then, each rating method obtains a different offset where the offset for the case of the score is proportional to the score.

   (e) Probably, fraction of reviews with "good" is a more likely to be a useful statistic than total number of reviews. For example, a very bad product could have a 1000 reviews with only 10 reviews saying "good", while a second product could have 8 good reviews out of 9. If we use total number of reviews with "good", the first product would have a higher score in that feature.

2. (a) Since there are two features,

   $$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \epsilon.$$

   (b) The transformed feature matrix and response vector is

   $$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} 1 \\ 4 \\ 3 \\ 7 \end{bmatrix},$$

The solution is $\hat{\boldsymbol{\beta}} = (\mathbf{A}^\mathsf{T}\mathbf{A})^{-1}\mathbf{A}^\mathsf{T}\mathbf{y}$. This can be solved in python with

```
A = np.array([[1,0,0],[1,0,1],[1,1,0],[1,1,1]])
y = np.array([1,4,3,7])
beta = np.linalg.lstsq(A,y)[0]
```

which returns the solution $\hat{\boldsymbol{\beta}} = [0.75, 2.5, 3.5]$.

3. An automobile engineer wants to model the relation between the accelerator control and the velocity of the car. The relation may not be simple since there is a lag in depressing the accelerator and the car actually accelerating. To determine the relation, the engineers measures the acceleration control input x_k and velocity of the car y_k at time instants $k = 0, 1, \ldots, T-1$. The measurements are made at some sampling rate, say once every 10 ms. The engineer then wants to fit a model of the form

$$y_k = \sum_{j=1}^{M} a_j y_{k-j} + \sum_{j=0}^{N} b_j x_{k-j} + \epsilon_k, \tag{1}$$

for coefficients a_j and b_j. In engineering this relation is called a *linear filter* and it statistics it is called an *auto-regressive moving average (ARMA)* model.

(a) The parameter vector would be

$$\boldsymbol{\beta} = [a_1, \ldots, a_M, b_0, \ldots, b_N]^\mathsf{T}.$$

Note the transpose to make this a column vector. There are $M + N + 1$ unknown parameters.

(b) We can rewrite the data as

$$\mathbf{A} = \left[\begin{array}{cccccc} y_{M-1} & \cdots & y_0 & x_M & \cdots & x_{M-N} \\ y_M & \cdots & y_1 & x_{M+1} & \cdots & x_{M-N+1} \\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ y_{T-2} & \cdots & y_{T-M-1} & x_{T-1} & \cdots & x_{T-N-1} \end{array}\right], \quad \mathbf{y} = \left[\begin{array}{c} y_M \\ \vdots \\ y_{T-1} \end{array}\right].$$

(c) First, we partition the matrix $\mathbf{A}$ into two parts:

$$\mathbf{A} = [\mathbf{A}_Y \;\; \mathbf{A}_X], \quad \mathbf{A}_Y = \left[\begin{array}{ccc} y_{M-1} & \cdots & y_0 \\ y_M & \cdots & y_1 \\ \vdots & \cdots & \vdots \\ y_{T-2} & \cdots & y_{T-M-1} \end{array}\right], \quad \mathbf{A}_X = \left[\begin{array}{ccc} x_M & \cdots & x_{M-N} \\ x_{M+1} & \cdots & x_{M-N+1} \\ \vdots & \cdots & \vdots \\ x_{T-1} & \cdots & x_{T-N-1} \end{array}\right].$$

Now,

$$\frac{1}{T}\mathbf{A}^\mathsf{T}\mathbf{A} = \left[\begin{array}{cc} \mathbf{A}_Y^\mathsf{T}\mathbf{A}_Y & \mathbf{A}_Y^\mathsf{T}\mathbf{A}_X \\ \mathbf{A}_X^\mathsf{T}\mathbf{A}_Y & \mathbf{A}_X^\mathsf{T}\mathbf{A}_X \end{array}\right].$$

The first term can be simplified as

$$\begin{aligned}
\frac{1}{T}(\mathbf{A}_Y^\intercal \mathbf{A}_Y)_{i,j} &\overset{(a)}{=} \frac{1}{T}\sum_k (\mathbf{A}_Y^\intercal)_{ik}(\mathbf{A}_Y)_{kj} \\
&\overset{(b)}{=} \frac{1}{T}\sum_k (\mathbf{A}_Y)_{ki}(\mathbf{A}_Y)_{kj} \\
&\overset{(c)}{=} \frac{1}{T}\sum_k y_{M+k-i}y_{M+k-j} \overset{(d)}{=} \frac{1}{T}\sum_k y_k y_{k+i-j} \approx R_{yy}(i-j),
\end{aligned}$$

where (a) follows from the definition of matrix multiplication; (b) uses the definition of a transpose matrix; (c) is the formula for the entries in $\mathbf{A}_Y$ (d) follows from a change of variables and (e) is the definition of the auto-correlation. The approximation is that there is a slight change in the end points of the summation which will not matter for large T. Similarly, one can show that

$$\begin{aligned}
\frac{1}{T}(\mathbf{A}_Y^\intercal \mathbf{A}_X)_{i,j} &= \frac{1}{T}\sum_k \sum_k y_{M+k-i}x_{M+k-j} \approx R_{yx}(i-j) \\
\frac{1}{T}(\mathbf{A}_X^\intercal \mathbf{A}_Y)_{i,j} &= \frac{1}{T}\sum_k \sum_k x_{M+k-i}y_{M+k-j} \approx R_{xy}(i-j) \\
\frac{1}{T}(\mathbf{A}_X^\intercal \mathbf{A}_X)_{i,j} &= \frac{1}{T}\sum_k \sum_k x_{M+k-i}x_{M+k-j} \approx R_{xx}(i-j).
\end{aligned}$$

Also,

$$\frac{1}{T}(\mathbf{A}_Y^\intercal \mathbf{y})_i = \frac{1}{T}\sum_k \sum_k y_{M+k-i}y_{M+k} \approx R_{yy}(M-i).$$

4. (a) Define

$$\mathbf{A} = \begin{bmatrix} \cos(\Omega_1(0)) & \cdots & \cos(\Omega_L(0)) & \sin(\Omega_1(0)) & \cdots & \sin(\Omega_L(0)) \\ \cos(\Omega_1(1)) & \cdots & \cos(\Omega_L(1)) & \sin(\Omega_1(1)) & \cdots & \sin(\Omega_L(1)) \\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ \cos(\Omega_1(T-1)) & \cdots & \cos(\Omega_L(T-1)) & \sin(\Omega_1(T-1)) & \cdots & \sin(\Omega_L(T-1)) \end{bmatrix}$$

and

$$\mathbf{x} = \begin{bmatrix} x_0 \\ \vdots \\ x_{T-1} \end{bmatrix}, \quad \boldsymbol{\beta} = \begin{bmatrix} a_1 \\ \vdots \\ a_L \\ b_1 \\ \vdots \\ b_L \end{bmatrix}.$$

Then $\mathbf{x} \approx \mathbf{A}\boldsymbol{\beta}$.

(b) If the frequencies Ω_ℓ are not known, then the model is nonlinear.