Unit 3: Multiple Linear Regression

Vaibhav Singh

17 September 2021

1. An online retailer like Amazon wants to determine which products to promote based on reviews. They only want to promote products that are likely to sell. For each product, they have past sales as well as reviews. The reviews have both a numeric score (from 1 to 5) and text.

   (a) To formulate this as a machine learning problem, suggest a target variable that the online retailer could use.

   Solution (a):
   We can use "Sales" as our target variable, which would denote the number of items of a particular product sold. The reason for choosing this target variable is that, we would like to maximise our product sales.

   (b) For the predictors of the target variable, a data scientist suggests to combine the numeric score with frequency of occurrence of words that convey judgement like "bad", "good", and "doesn't work." Describe a possible linear model for this relation.

   Solution (b):
   $CombinedScore = n_f \times f$
   where $n_f = numeric\ score$ and $f = frequency$
   Linear model $\Rightarrow$
   $\hat{y} = \beta_0 + \beta_1 x$
   where $x = CombinedScore$

   (c) Now, suppose that some reviews have a numeric score from 1 to 5 and others have a score from 1 to 10. How would change your features?

   Solution (c):
   We can normalise the scores as $\hat{score} = score/MaxScore$
   So the scores on the scale $1-10$ would be $score/10$ and scores on the scale $1-5$ would be $score/5$

   (d) Now suppose the reviews have either (a) a score from 1 to 5; (b) a rating that is simply good or bad; or (c) no numeric rating at all. How would you change your features?
   Solution (d):

We can handle this by vectorizing our categorical features using one-hot encoding principle.

We can see that there are 8 ordinal variables

$$features = \begin{aligned} 1 &= [10000000] \\ 2 &= [01000000] \\ 3 &= [00100000] \\ 4 &= [00010000] \\ 5 &= [00001000] \\ Good &= [00000100] \\ Bad &= [00000010] \\ No\ Rating &= [00000001] \end{aligned}$$

(e) For the frequency of occurrence of a word such as "good", which variable would you suggest to use as a predictor: (a) total number of reviews with the word "good"; or (b) fraction of reviews with the word "good"?
Solution (e):

I think that option(b: fraction of reviews with the word "good") would be a better predictor, because $\frac{positives}{total}$ will give relative importance to positive reviews. As the model would be able to generalize better.
Eg. If we have 200 reviews, out of which 50 are positive and 150 are bad, then using only the figure 50, can be misleading in the model, since it gives no information as to how many negative reviews are there. But giving the fraction(50/200) gives us information that positive reviews are very less. This information would be very useful to the model.

2. Suppose we are given data:

x_{i1}	0	0	1	1
x_{i2}	0	1	0	1
y_i	1	4	3	7

(a) Write an equation for a linear model for y in terms of x_1 and x_2.
Solution (a):
Since this can be solved using Linear Regression we can write
$\hat{y_k} = \beta_0 + x_1\beta_1 + x_2\beta_2$
Now in matrix form, we have $\hat{Y} = X\boldsymbol{\beta}$
where

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} \\ 1 & x_{21} & x_{22} \\ 1 & x_{31} & x_{32} \\ 1 & x_{41} & x_{42} \end{bmatrix}$$

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

(b) Given the data compute the least-squares estimate for the parameters in the model.

Solution (b):

Now we have writen $\hat{Y} = X\boldsymbol{\beta}$ as our linear regression model. As derived in class, we know that $\beta = (X^\intercal X)^{-1}X^\intercal \hat{Y}$.

First we will compute $X^\intercal = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$

Now $X^\intercal X = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 4 & 2 & 2 \\ 2 & 2 & 1 \\ 2 & 1 & 2 \end{bmatrix}$

Now we need $(X^\intercal X)^{-1} = \begin{bmatrix} 0.75 & -0.5 & -0.5 \\ -0.5 & 1. & 0. \\ -0.5 & 0. & 1. \end{bmatrix}$

Now $(X^\intercal X)^{-1}X^\intercal = \begin{bmatrix} 0.75 & 0.25 & 0.25 & -0.25 \\ -0.5 & -0.5 & 0.5 & 0.5 \\ -0.5 & 0.5 & -0.5 & 0.5 \end{bmatrix}$

Finally we need $(X^\intercal X)^{-1}X^\intercal y = \begin{bmatrix} 0.75 & 0.25 & 0.25 & -0.25 \\ -0.5 & -0.5 & 0.5 & 0.5 \\ -0.5 & 0.5 & -0.5 & 0.5 \end{bmatrix} \begin{bmatrix} 1 \\ 4 \\ 3 \\ 7 \end{bmatrix}$

$$\boldsymbol{\beta} = \begin{bmatrix} 0.75 \\ 2.5 \\ 3.5 \end{bmatrix}$$

We can see that for simple data, we can compute the matrices and their inverses manually, but for large data, we should use numpy and LinearRegression library from sklearn. Following is the snippet.

```python
import numpy as np
from sklearn.linear_model import LinearRegression

x1 = np.array([[0,0],[0,1], [1,0], [1,1]])
y = np.array([1,4,3,7])

model = LinearRegression()
model.fit(x1,y)

print(" Beta parameters are ", model.coef_)
```

```
  print(" Intercept is ",  model.intercept_)

Output :
Beta parameters are  [2.5 3.5]
Intercept is  0.7500000000000004
```

3. Write each of the following models as transformed linear models. That is, find a parameter vector $\boldsymbol{\beta}$ in terms of the given parameters a_i and a set basis functions of functions $\boldsymbol{\phi}(\mathbf{x})$ such that $\hat{y} = \boldsymbol{\beta}^\mathsf{T}\phi(\mathbf{x})$. Also, show how to recover the original parameters a_i from the parameters β_j:

   (a) $\hat{y} = (a_1x_1 + a_2x_2)e^{-x_1-x_2}$.
   Solution (a):

   Let us define $\boldsymbol{\beta}^\mathsf{T} = [a_1 \quad a_2]$

   Let us also define basis function set

   $$\boldsymbol{\phi}(x) = \begin{array}{l} \boldsymbol{\phi}_1(x) = x_1e^{-x_1-x_2} \\ \boldsymbol{\phi}_2(x) = x_2e^{-x_1-x_2} \end{array}$$

   Now we can write
   $\hat{y} = a_1\boldsymbol{\phi}_1(x) + a_2\boldsymbol{\phi}_2(x)$

   $\therefore \hat{y} = \boldsymbol{\beta}^\mathsf{T}\phi(\mathbf{x})$

   (b) $\hat{y} = \begin{cases} a_1 + a_2x & \text{if } x < 1 \\ a_3 + a_4x & \text{if } x \geq 1 \end{cases}$
   Solution (b):
   Let us define $\boldsymbol{\beta}^\mathsf{T} = [a_1 \quad a_2 \quad a_3 \quad a_4]$

   Let us also define basis function set

   $$\boldsymbol{\phi}(x) = \begin{array}{l} \boldsymbol{\phi}_1(x) = 1 \quad if x < 1 \\ \boldsymbol{\phi}_2(x) = x \quad if x < 1 \\ \\ \boldsymbol{\phi}_3(x) = 1 \quad if x \geq 1 \\ \boldsymbol{\phi}_4(x) = x \quad if x \geq 1 \end{array}$$

   Now we can write
   $\hat{y} = a_1\boldsymbol{\phi}_1(x) + a_2\boldsymbol{\phi}_2(x) + a_3\boldsymbol{\phi}_3(x) + a_4\boldsymbol{\phi}_4(x)$

   $\therefore \hat{y} = \boldsymbol{\beta}^\mathsf{T}\phi(\mathbf{x})$

   (c) $\hat{y} = (1 + a_1x_1)e^{-x_2+a_2}$.
   Solution (c):
   $\hat{y} = (1 + a_1x_1)e^{-x_2+a_2}$

$= e^{-x_2}e^{a_2} + a_1 e^{a_2} x_1 e^{-x2}$

Now taking $\boldsymbol{\beta}_1 = e^{a_2}$, $\boldsymbol{\phi}_1(x) = e^{-x_2}$, $\boldsymbol{\phi}_2(x) = x_1 e^{-x_2}$

We can write as $\hat{y} = \boldsymbol{\beta}_1\boldsymbol{\phi}_1(x) + a_1\boldsymbol{\beta}_1\boldsymbol{\phi}_2(x)$

Now taking $\boldsymbol{\beta}_2 = a_1\boldsymbol{\beta}_1$ we can write

$\hat{y} = \boldsymbol{\beta}_1\boldsymbol{\phi}_1(x) + \boldsymbol{\beta}_2\boldsymbol{\phi}_2(x)$

$\boldsymbol{\beta}^{\mathsf{T}} = [\boldsymbol{\beta}_1 \quad \boldsymbol{\beta}_2]$

$\boldsymbol{\phi}(x) = \begin{matrix}\boldsymbol{\phi}_1(x) \\ \boldsymbol{\phi}_2(x)\end{matrix}$

$\therefore \hat{y} = \boldsymbol{\beta}^{\mathsf{T}}\phi(\mathbf{x})$

4. 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) Describe a vector $\boldsymbol{\beta}$ with the unknown parameters. How many unknown parameters are there?

Solution (a):

The $\boldsymbol{\beta}$ vector would be of this form $\boldsymbol{\beta} = \begin{bmatrix} a_1 \\ a_2 \\ \cdots \\ \cdots \\ a_M \\ b_0 \\ \cdots \\ \cdots \\ a_N \end{bmatrix}$

This is a vector of dimension $(M + N + 1) \times 1$.

(b) Describe the matrix $\mathbf{A}$ and target vector $\mathbf{y}$ so that we can rewrite the model (1) in matrix form,

$$\mathbf{y} = \mathbf{A}\boldsymbol{\beta} + \boldsymbol{\epsilon}.$$

Your matrix $\mathbf{A}$ will have entries of y_k and x_k in it.

Solution (b):

The matrix $\mathbf{A}$ will be of the following form

$$\begin{bmatrix} 0 & 0 & 0 & \dots\dots & 0 & x_0 & 0 & 0 & \dots\dots & 0 \\ y_0 & 0 & 0 & \dots\dots & 0 & x_1 & x_0 & 0 & \dots\dots & 0 \\ y_1 & y_0 & 0 & \dots\dots & 0 & x_2 & x_1 & x_0 & \dots\dots & 0 \\ \dots\dots\dots & & & & & & & & & \\ y_{(T-2)} & y_{(T-3)} & y_{(T-4)} & \dots\dots & y_{(T-2-M)} & x_{(T-1)} & x_{(T-2)} & x_{(T-3)} & \dots\dots & x_{(T-1-N)} \end{bmatrix}$$

$\mathbf{A}$ has a dimension of $T \times (M + N + 1)$

Target vector will be a column vector of dimension $T \times 1$

$$\begin{bmatrix} y_0 \\ y_1 \\ . \\ . \\ . \\ y_{T-1} \end{bmatrix}$$

Now we can easily write our solution as

$$\mathbf{y} = \mathbf{A}\boldsymbol{\beta} + \boldsymbol{\epsilon}.$$

(c) (Graduate students only) Show that, for $T \gg N$ and $T \gg M$, the coefficients of $(1/T)\mathbf{A}^\mathsf{T}\mathbf{A}$ and $(1/T)\mathbf{A}^\mathsf{T}\mathbf{y}$ can be approximately computed from the so-called auto-correlation functions

$$R_{xy}(\ell) = \frac{1}{T}\sum_{k=0}^{T-1} x_k y_{k+\ell}, \quad R_{yy}(\ell) = \frac{1}{T}\sum_{k=0}^{T-1} y_k y_{k+\ell}, \quad R_{xx}(\ell) = \frac{1}{T}\sum_{k=0}^{T-1} x_k x_{k+\ell},$$

In the sum, we take $x_k = 0$ or $y_k = 0$ whenever $k < 0$ or $k \geq T$.

Solution (c):

The matrix $\mathbf{A}^\mathsf{T}\mathbf{A}$ can be seen as follows:

$$\begin{bmatrix} 0 & y_0 & y_1 & \dots\dots & y_{T-2} \\ 0 & 0 & y_0 & \dots\dots & y_{T-3} \\ \dots\dots\dots & & & & \\ x_0 & x_1 & x_2 & \dots\dots & x_{T-1} \\ 0 & x_0 & x_1 & \dots\dots & x_{T-2} \\ \dots\dots\dots & & & & \\ 0 & 0 & 0 & \dots\dots & x_{T-1-N} \end{bmatrix} \begin{bmatrix} 0 & 0 & \dots\dots & 0 & x_0 & 0 & \dots\dots & 0 \\ y_0 & 0 & \dots\dots & 0 & x_1 & x_0 & \dots\dots & 0 \\ y_1 & y_0 & \dots\dots & 0 & x_2 & x_1 & \dots\dots & 0 \\ \dots\dots\dots & & & & & & & \\ y_{T-2} & y_{T-3} & \dots\dots & y_{T-2-M} & x_{T-1} & x_{T-2} & \dots\dots & x_{T-1-N} \end{bmatrix}$$

Now we can easily see that the resultant will look something like

$$\begin{bmatrix} R_{yy}(0) & R_{yy}(1) & .... & R_{xy}(0) & R_{xy}(1) & ...... & R_{xy}(N-1) \\ R_{yy}(1) & R_{yy}(0) & .... & .... & R_{xy}(0) & ...... & R_{xy}(N-2) \\ .... \\ R_{xy}(N-1) & R_{xy}(N-2) & .... & R_{xx}(N) & R_{xx}(N-1) & ...... & ..... \end{bmatrix}$$

Similarly the $\mathbf{A}^{\mathsf{T}}y$ can be seen as follows

$$\begin{bmatrix} 0 & y_0 & y_1 & .... & y_{T-2} \\ 0 & 0 & y_0 & .... & y_{T-3} \\ .... \\ x_0 & x_1 & x_2 & .... & x_{T-1} \\ 0 & x_0 & x_1 & .... & x_{T-2} \\ .... \\ 0 & 0 & 0 & .... & x_{T-1-N} \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ .. \\ .. \\ y_{T-1} \end{bmatrix}$$

The resultant would be

$$\begin{bmatrix} R_{yy}(1) \\ R_{yy}(2) \\ .. \\ R_{yy}(M-1) \\ R_{xy}(0) \\ R_{xy}(1) \\ .. \\ R_{xy}(N-1) \end{bmatrix}$$

5. In audio processing, one often wants to find tonal sounds in segments of the recordings. This can be formulated as follows: We are given samples of an audio segment, x_k, $k = 0, \dots, N-1$, and wish to fit a model of the form,

$$x_k \approx \sum_{\ell=1}^{L} a_\ell \cos(\Omega_\ell k) + b_\ell \sin(\Omega_\ell k), \tag{2}$$

where L are a number of tones present in the audio segment; Ω_ℓ are the tonal frequencies and a_ℓ and b_ℓ are the coefficients.

(a) Show that if the frequencies Ω_ℓ are given, we can solve for the coefficients a_ℓ and b_ℓ using linear regression. Specifically, rewrite the model (2) as $\mathbf{x} \approx \mathbf{A}\boldsymbol{\beta}$ for appropriate $\mathbf{x}$, $\mathbf{A}$ and $\boldsymbol{\beta}$. Then describe exactly how we obtain the coefficients a_ℓ and b_ℓ from this model.

Solution (a):
Let the β vector be $[a_1\, a_2\, ...a_L\, b_1\, b_2\, ...\, b_L]$ This is a vector of length $2L$. Since Ω_l and k are given to us, they form the constituents of data matrix X.
Using the parameterized form of Linear Regression, we will choose basis functions as $\Phi_1^k(\Omega_l k) = cos(\Omega_l k)$ and $\Phi_2^k(\Omega_l k) = sin(\Omega_l k)$ where the k represents the k_{th} sample from the data. Now our data matrix A would look like

$$\begin{bmatrix} \Phi_1^0 & \Phi_1^0 & \cdots\cdots & \Phi_2^0 & \Phi_2^0 \\ \Phi_1^1 & \Phi_1^1 & \cdots\cdots & \Phi_2^1 & \Phi_2^1 \\ \Phi_1^2 & \Phi_1^2 & \cdots\cdots & \Phi_2^2 & \Phi_2^2 \\ \cdots\cdots & & & & \\ \Phi_1^{N-1} & \Phi_1^{N-1} & \cdots\cdots & \Phi_2^{N-1} & \Phi_2^{N-1} \end{bmatrix}$$

This matrix is of dimensions $N \times 2L$.

Now we can write $X = A\beta$ as our linear regression model. As derived in class, we know that $\beta = (A^T A)^{-1} A^T X$. Solving this equation with the help of python's numpy and scikit learn library routines, we can solve for β

(b) Now suppose the frequencies Ω_ℓ were not known. If we had to solve for the parameters a_ℓ, b_ℓ and Ω_ℓ, would the problem be a linear regression problem?

Solution (a):

No, then the problem would not be that of Linear Regression, since the coefficients a_ℓ, b_ℓ and Ω_ℓ have multiplicative relationship rather than linear.

6. *Python broadcasting.* Rewrite the following code without for-loops using vectorization and python broadcasting.

(a) Given a data matrix `X` and vector `beta` compute a vector `yhat`:

```
n = X.shape[0]
yhat = np.zeros(n)
for i in range(n):
    yhat[i] = beta[0]*X[i,0] + beta[1]*X[i,1] + beta[2]*X[i,1]*X[i,2]
```

Solution (a):

```
yhat = np.zeros(n)
X[:, −1] = X[:, −1] * X[:,−2]
yhat = X*beta
yhat = np.sum(y, axis = 1)
```

(b) Given vectors `x`, `alpha`, and `beta` computes a vector `yhat`:

```
n = len(x)
m = len(alpha)
yhat = np.zeros(n)
for i in range(n):
    for j in range(m):
        yhat[i] += alpha[j]*np.exp(-beta[j]*x[i])
```

Solution (b):

```
exp_ = np.exp(-x[:,None] * beta[None,:])
yhat = alpha * exp_
yhat = np.sum(yhat, axis=1)
```

(c) Given arrays `x` and `y`, find the squared distances `dist`:

```
n,d = x.shape
m,d = y.shape
dist = np.zeros((n,m))
for i in range(n):
    for j in range(m):
        for k in range(d):
            dist[i,j] += (x[i,k]-y[j,k])**2
```

Solution (c):

```
dist = np.zeros((n,m))
diff_ = x[:,None,:] - y[None,:,:]
dist = np.sum(diff_**2, axis = 1)
```