Introduction to Machine Learning
Problems Unit 3: Multiple Linear Regression

Prof. Sundeep Rangan

Willem Neuefeind Lessig's Solutions 9/17/25

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.

      i. **likely_to_sell** is a binary variable
      ii. 0: $<$ 50% chance of selling
      iii. 1: $>=$ 50% chance of selling

   (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.

      i. Logistic regression: $P = \frac{1}{1+e^{-z}}$
      ii. where $z = \beta_0 + \beta_1(Score1-5) + \beta_2(bad\_freq) + \beta_3(good\_freq) + \beta_4(doesn't\_work\_freq)$

   (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?

      i. Change $\beta_1(Score1-5)$ into $\beta_1(Score0-1)$ and turn each score into a fraction /10 or /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?

      i. Replace $\beta_1(Score1-5)$ with $\beta_1(Score1-0)$. Set the score as either
         A. The 1-5 score as a fraction
         B. 0 for a bad rating 1 for a good rating
         C. 0 if no rating

   (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"?

      i. b, the proportion of reviews saying good is a much more accurate predictor

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

(b) Given the data compute the least-squares estimate for the parameters in the model.
Normal Equations:
$\sum y = n\beta_0 + \beta_1 \sum x_1 + \beta_2 \sum x_2$
$\sum x_1 y = \beta_0 \sum x_1 + \beta_1 \sum x_1^2 + \beta_2 \sum x_1 x_2$
$\sum x_2 y = \beta_0 \sum x_2 + \beta_1 \sum x_1 x_2 + \beta_2 \sum x_2^2$
Solve for each variable
$n = 4$
$\sum y = 15$
$\sum x_{1i} = 2$
$\sum x_{2i} = 2$
$\sum x_{1i}^2 = 2$
$\sum x_{2i}^2 = 2$
$\sum x_1 x_2 = 0*0 + 0*1 + 1*0 + 1*1 = 1$
$\sum x_1 y = 0*1 + 0*4 + 1*3 + 1*7 = 10$
$\sum x_2 y = 0*1 + 1*4 + 0*3 + 1*7 = 11$
Put it together
$15 = 4\beta_0 + 2\beta_1 + 2\beta_2$
$10 = 2\beta_0 + 2\beta_1 + 1\beta_2$
$11 = 2\beta_0 + 1\beta_1 + 2\beta_2$
$\beta_0 = 0.75, \beta_1 = 2.5, \beta_2 = 3.5$
$\hat{y} = 0.75 + 2.5x_1 + 3.5x_2$

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) $\hat{y} = (a_1 x_1 + a_2 x_2)e^{-x_1 - x_2}$.

      i. $\hat{y} = a_1 \cdot (x_1 e^{-x_1 - x_2}) + a_1 \cdot (x_2 e^{-x_1 - x_2})$

      ii. $\boldsymbol{\beta} = \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$

      iii. $\boldsymbol{\phi}(\boldsymbol{x}) = \begin{pmatrix} \phi_1(x) \\ \phi_2(x) \end{pmatrix} = \begin{pmatrix} x_1 e^{-x_1 - x_2} \\ x_2 e^{-x_1 - x_2} \end{pmatrix}$

      iv. $\hat{y} = \beta^\mathsf{T}\phi(x) = \beta_1(x_1 e^{-x_1 - x_2}) + \beta_2(x_1 e^{-x_1 - x_2})$

      v. to revert, $a_1 = \beta_1, a_2 = \beta_2$

   (b) $\hat{y} = \begin{cases} a_1 + a_2 x & \text{if } x < 1 \\ a_3 + a_4 x & \text{if } x \geq 1 \end{cases}$

i. Use indicator functions

ii. $I_1(x) = \begin{cases} 1 & \text{if } x < 1 \\ 0 & \text{if } x \geq 1 \end{cases}$

iii. $I_2(x) = \begin{cases} 0 & \text{if } x < 1 \\ 1 & \text{if } x \geq 1 \end{cases}$

iv. $\hat{y} = (a_1 + a_2 x) \cdot I_1(x) + (a_3 + a_4 x) \cdot I_2(x)$

v. $\hat{y} = a_1 I_1(x) + a_2 x I_1(x) + a_3 I_2(x) + a_4 x I_2(x)$

vi. $\boldsymbol{\beta} = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \end{pmatrix} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix}$

vii. $\boldsymbol{\phi(x)} = \begin{pmatrix} \phi_1(x) \\ \phi_2(x) \\ \phi_2(x) \\ \phi_3(x) \end{pmatrix} = \begin{pmatrix} I_1(x) \\ x I_1(x) \\ I_2(x) \\ x I_2(x) \end{pmatrix}$

viii. $\hat{y} = \beta^{\mathsf{T}} \phi(x) = \beta_1 I_1(x) + \beta_2 x I_1(x) + \beta_3 I_2(x) + \beta_4 x I_2(x)$

ix. to revert, $\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \end{pmatrix}$

(c) $\hat{y} = (1 + a_1 x_1) e^{-x_2 + a_2}$.

i. $\hat{y} = e^{-x_2 + a_2} + a_1 x_1 e^{-x_2 + a_2}$

ii. $\hat{y} = e^{-x_2} e^{a_2} + a_1 x_1 e^{-x_2} e^{a_2}$

iii. $\beta = \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} e^{a_2} \\ a_1 e^{a_2} \end{pmatrix}$

iv. $\boldsymbol{\phi(x)} = \begin{pmatrix} \phi_1(x) \\ \phi_2(x) \end{pmatrix} = \begin{pmatrix} e^{-x^2} \\ x_1 e^{-x_2} \end{pmatrix}$

v. $\hat{y} = \beta^{\mathsf{T}} \phi(x) = \beta_1 (e^{-x_2}) + \beta_2 (x_1 e^{-x_2})$

vi. to revert, $\begin{pmatrix} a_2 \\ a_1 \end{pmatrix} = \begin{pmatrix} \ln(\beta_1) \\ \frac{\beta_2}{\beta_1} \end{pmatrix}$ $(\beta_1 > 0)$

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?

$$\beta = \begin{pmatrix} a_1 \\ \cdots \\ a_M \\ b_0 \\ \cdots \\ b_N \end{pmatrix}, \text{ total} = M + N + 1$$

(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.

$$A = \begin{pmatrix} y_{k-1} \\ y_{k-2} \\ \cdots \\ y_{k-M} \\ x_k \\ x_{k-1} \\ \cdots \\ x_{k-N} \end{pmatrix}$$

(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 $k \geq T$.

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, $k = 0, \ldots, N-1$, 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.

$$\mathbf{x} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{N-1} \end{pmatrix}$$

$$\boldsymbol{\beta} = \begin{pmatrix} a_1 \\ b_1 \\ a_2 \\ b_2 \\ \vdots \\ a_L \\ b_L \end{pmatrix}$$

$$A = \begin{pmatrix} \cos(\Omega_1 \cdot 0) & \sin(\Omega_1 \cdot 0) & \cdots & \cos(\Omega_L \cdot 0) & \sin(\Omega_L \cdot 0) \\ \cos(\Omega_1 \cdot 1) & \sin(\Omega_1 \cdot 1) & \cdots & \cos(\Omega_L \cdot 1) & \sin(\Omega_L \cdot 1) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \cos(\Omega_1 \cdot (N-1)) & \sin(\Omega_1 \cdot (N-1)) & \cdots & \cos(\Omega_L \cdot (N-1)) & \sin(\Omega_L \cdot (N-1)) \end{pmatrix}$$

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

$$\hat{\boldsymbol{\beta}} = (A^T A)^{-1} A^T \mathbf{x}$$

$$\hat{\boldsymbol{\beta}} = \begin{pmatrix} \hat{a}_1 \\ \hat{b}_1 \\ \hat{a}_2 \\ \hat{b}_2 \\ \vdots \\ \hat{a}_L \\ \hat{b}_L \end{pmatrix}$$

$$\hat{a}_\ell = \hat{\beta}_{2\ell-1} \quad \text{for } \ell = 1, 2, \ldots, L$$

$$\hat{b}_\ell = \hat{\beta}_{2\ell} \quad \text{for } \ell = 1, 2, \ldots, L$$

(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?

The model wouldn't be linear because for $\mathbf{x} \approx A\boldsymbol{\beta}$ A must be a known function, but if Ω_ℓ is unknown then A depends on itself.

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:`

```
yhat = (beta[0] * X[:, 0] +
        beta[1] * X[:, 1] +
        beta[2] * X[:, 1] * X[;, 2])
```

(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:

```
np.sum(alpha * np.exp(-beta * x[:, None]), 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:

```
diff = x[:,None,:]-y[None,:,:]
d = np.sum(diff**2,axis=2)
```