Chapter 3

Multivariate Linear Regression

3.1 Multiple Features

We now revisit the Housing price model now with more features, such as number of bedrooms, number of floors, age of the house, see Table below: And we use the following

No. (m)	**Size sqft** (x_1)	**# of Bedrm** (x_2)	**# of Floors** (x_3)	**Age** (x_4)	**Price in $1k** (y)
1	2104	5	1	45	460
2	1416	3	2	40	232
3	1534	3	2	30	315
4	852	2	1	36	178
...	...	...	...	...	...

Table 3.1: Multiple Features in the Housing Price Prediction

notations to represent the data above:

- x_j = j-th feature of the input (here x_1 through x_4)
- n = number of features (here $n = 4$)
- $x^{(i)}$ = a vector of features of the i-th training example (e.g. $x^{(2)} = [1416, 3, 2, 40]$ is a vector)
- $x_j^{(i)}$ = value of feature j in i-th training example (e.g. $x_2^{(3)} = 3$)

3.1.1 Multivariate Model Representation

Previously for univariate linear model, we have $f_{w,b}(x) = wx + b$, now we need to update our model to represent multi-variables:

$$f_{w,b}(x) = w_1x_1 + w_2x_2 + w_3x_3 + \cdots + w_nx_n + b$$

We can then use $\mathbf{W} = [w_1, w_2, w_3, \cdots, w_n]$ and $\mathbf{X} = [x_1, x_2, \cdots, x_n]$ and rewrite f in the following form:

$$f_{\mathbf{W},b}(\mathbf{X}) = \mathbf{W} \cdot \mathbf{X} + b \tag{3.1}$$

This is called the "**multiple variable linear regression model**", where "·" is the dot product of **W** and **X**.

3.2 Vectorization

When use `Python`, we have two ways to implement the model function $f_{\mathbf{W},b}(\mathbf{X}) = \mathbf{W} \cdot \mathbf{X} + b = \sum_{j=1}^{n}(w_jx_j) + b$:

1. use `for` loop (**sequential add-up**)
2. use `numpy.dot` (**parallel vector computation, effective use of GPU**)

For the first method, we will have the following code:

```
import numpy as np
w = np.array([..., ..., ..., ...])
for j in range n:          # we can replace the for loop here by vectorization
    f += w[j] * x[j]
f += b
```

For the second method, we will have the one liner to replace the `for` loop:

```
f = np.dot(w, x) + b
```

3.3 Gradient Descent for Multivariate Linear Regression Using Vectorization

For recap, we have the following two sets of notations in representing our models so far:

1. original notation with itemized features and weights (in coding, used `for loop`)
2. vectorized notation with vector $\mathbb{X}$ as features vector and $\mathbb{W}$ as weights (in coding, used `np.dot()`)

We summarized in the following table with the notations for model representation:

	Original Notation	**Vectorized Notation**
Model	$f_{\mathbf{W},b}((\mathbf{X})) = w_1x_1 + \cdots + w_nx_n + b$	$f_{\mathbf{W},b}(\mathbf{X}) = \mathbf{W} \cdot \mathbf{X} + b$
Cost Function	$J(w_1, \ldots, w_n, b)$	$J(\mathbf{W}, b)$

Table 3.2: Previous Notation v. Vectorization Notation

Now we further look at the gradient descent algorithm for one feature versus multiple features:

1. One feature, only one x, and m total examples:

$$w \leftarrow w - \alpha \cdot \frac{1}{2m} \sum_{i=1}^{m} (wx^{(i)} + b - y^{(i)}) \cdot x^{(i)} \tag{3.2}$$

$$b \leftarrow b - \alpha \cdot \frac{1}{2m} \sum_{i=1}^{m} (wx^{(i)} + b - y^{(i)}) \tag{3.3}$$

2. Multiple (n) features, x_1 through x_n, still m total examples:

$$w_1 \leftarrow w_1 - \alpha \cdot \frac{1}{2m} \sum_{i=1}^{m} (\mathbf{W} \cdot \mathbf{X}^{(\mathbf{i})} + b - y^{(i)}) \cdot x_1^{(i)} \tag{3.4}$$

$$\vdots \tag{3.5}$$

$$w_n \leftarrow w_n - \alpha \cdot \frac{1}{2m} \sum_{i=1}^{m} (\mathbf{W} \cdot \mathbf{X}^{(\mathbf{i})} + b - y^{(i)}) \cdot x_n^{(i)} \tag{3.6}$$

$$b \leftarrow b - \alpha \cdot \frac{1}{2m} \sum_{i=1}^{m} (\mathbf{W} \cdot \mathbf{X}^{(\mathbf{i})} + b - y^{(i)}) \tag{3.7}$$

where we will need to simultaneously update w_j for $j = 1, \ldots, n$ (will need a `for loop` in range n) and b.