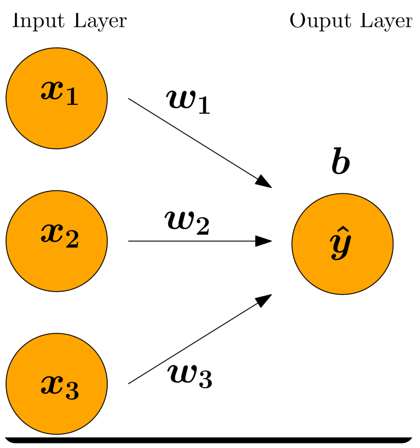
CH3.2 Multiple Linear Regression



Multiple linear regression:

$$\boxed{\hat{y} = w_1 x_1 + w_2 x_2 + w_3 x_3 + b}$$

$w_1, w_2, w_3 \rightarrow$ weights

$b \rightarrow$ bias

Consider the input points in the form $(x_1^{(i)}, x_2^{(i)}, x_3^{(i)})$ and the output points y_i for $i = 1, 2, \ldots N$. Our data may look like

X_1	X_2	X_3	Y
2.1	3.5	-0.5	13
6.21	12.1	4	16.5
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$x_1^{(i)}$	$x_2^{(i)}$	$x_3^{(i)}$	y_i $\rightarrow$ i^{th}-row.
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$x_1^{(N)}$	$x_2^{(N)}$	$x_3^{(N)}$	y_N

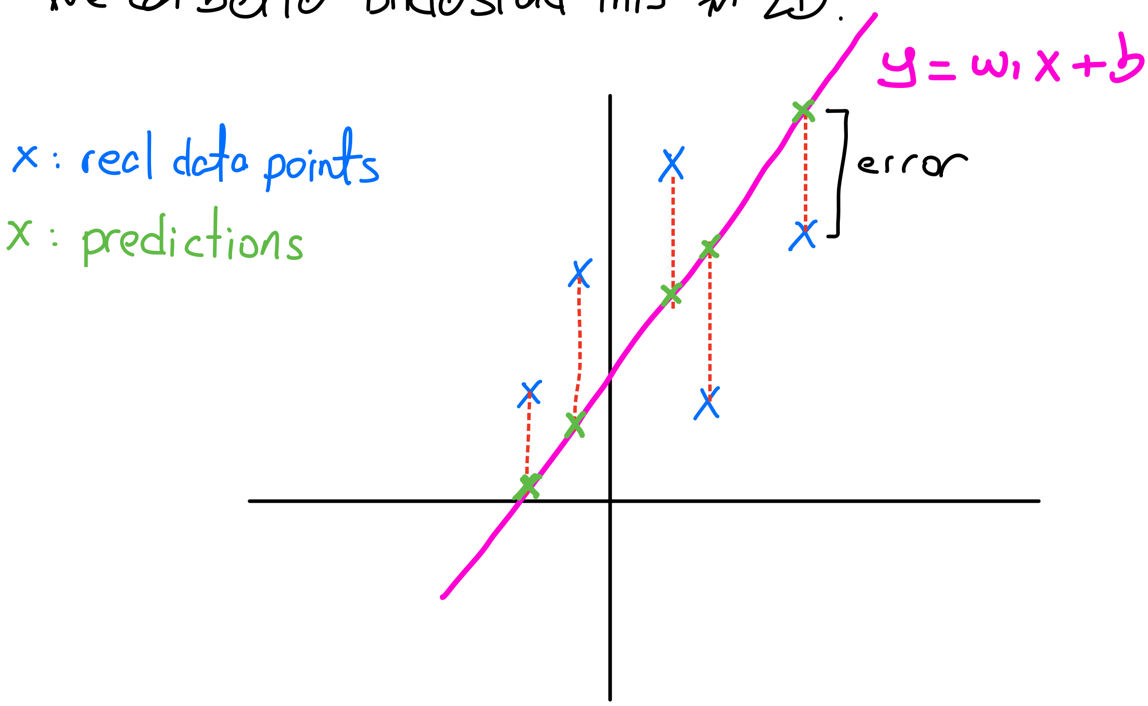
real (labeled) values

Define a loss function to measure the error between the prediction and the real labeled value. The most common one could be "the mean-square" loss function

$$C(\hat{y}) := \frac{1}{N}\sum_{i=1}^{N}(\overbrace{\hat{y}_i}^{\text{prediction}} - \underbrace{y_i}_{\text{real value}})^2$$

$$= \frac{1}{N}\sum_{i=1}^{N}\Big(\underbrace{w_1 x_1^{(i)} + w_2 x_2^{(i)} + w_3 x_3^{(i)} + b}_{\text{prediction}} - y_i\Big)^2 ,$$

Remarks: (1) Notice that we measure the error for each point for $i = 1, 2, \ldots N$ and average them out. We can better understand this in 2D.



(2) There are various types of loss functions (cross entropy, L_1 loss, Huber loss etc.). They all serve the same purpose.

Then we can state our problem as follows

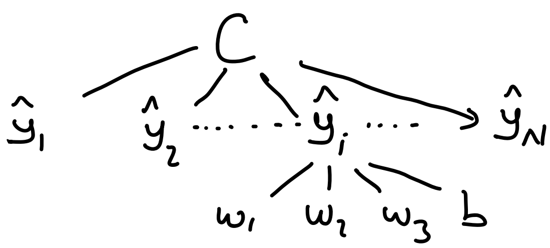
$$\boxed{\text{find } w_1, w_2, w_3 \text{ and } b \text{ that minimizes } C(\hat{y})}$$

We will use gradient decent so we need partial derivatives

$$\frac{\partial C}{\partial w_1}, \frac{\partial C}{\partial w_2}, \frac{\partial C}{\partial w_3} \text{ and } \frac{\partial C}{\partial b}$$

From (*) C is a function of $\hat{y}_1, \hat{y}_2, \dots \hat{y}_N$ and every each of these prediction are function of w_1, w_2, w_3, b

We can display this as



$$C = \frac{1}{N}\left((\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 + \dots (\hat{y}_i - y_i)^2 + \dots (\hat{y}_N - y_N)^2\right)$$

where $\hat{y}_i = w_1 x_1^{(i)} + w_2 x_2^{(i)} + w_3 x_3^{(i)} + b$

$$\frac{\partial C}{\partial w_1} = \frac{\partial C}{\partial \hat{y}_1} \cdot \frac{\partial \hat{y}_1}{\partial w_1} + \frac{\partial C}{\partial \hat{y}_2} \frac{\partial \hat{y}_2}{\partial w_1} + \dots \frac{\partial C}{\partial \hat{y}_n} \frac{\partial \hat{y}_n}{\partial w_1} = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial C}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial w_1}$$

$$= 2(\hat{y}_1 - y_1) x_1^{(1)} + 2(\hat{y}_2 - y_2) x_1^{(2)} + \dots 2(\hat{y}_N - y_N) \cdot x_1^{(N)}$$

$$= \frac{1}{N} \sum_{i=1}^{N} 2 (\hat{y}_i - y_i) \, x_1^{(i)} = \frac{2}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i) \, x_1^{(i)}$$

Repeat the same procedure for w_2, w_3 and b to obtain

$$\frac{\partial C}{\partial w_1} = \frac{2}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)x_1^{(i)}$$

$$\frac{\partial C}{\partial w_2} = \frac{2}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)x_2^{(i)}$$

$$\frac{\partial C}{\partial w_3} = \frac{2}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)x_3^{(i)}$$

$$\frac{\partial C}{\partial b} = \frac{\partial C}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial b} = \frac{2}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)\cdot 1$$

Note that we can have a more compact notation

$$\nabla C := \begin{bmatrix} \partial C/\partial w_1 \\ \partial C/\partial w_2 \\ \partial C/\partial w_3 \\ \partial C/\partial b \end{bmatrix} = \frac{2}{n}\sum_{i=1}^{n}\begin{bmatrix} (\hat{y}_i - y_i)\, x_1^{(i)} \\ (\hat{y}_i - y_i)\, x_2^{(i)} \\ (\hat{y}_i - y_i)\, x_3^{i} \\ (\hat{y}_i - y_i)\cdot 1 \end{bmatrix}$$

Then we can express the gradient decent as follows.

For $W^{(i)} = (w_1^{(i)}, w_2^{(i)}, w_3^{(i)}, b)$ and learning rate r

$$W^{(i+1)} = W^{(i)} - r\nabla C(W^{(i)}) \quad , \quad i = 0, 1, 2, \dots N$$

Ex: Consider the data set

x_1	x_2	Y
3	3	27
8	6	68
8	1	58
3	4	29
1	6	19
4	9	46

Build a linear regression model of the form

$$\boxed{\hat{y} = w_1 x_1 + w_2 x_2 + b}$$

for this data set. Do not include the last data point in your model and keep it for testing. If your model is good enough, given the inputs $x_1 = 4$, $x_2 = 9$ the output $\hat{y}$ should be close to 46. Let's pick $r = 0.01$

Sol: Let's do this step by step. Let's perform gradient descent for the first data point

$$x_1^{(1)} = 3, \quad x_2^{(1)} = 3, \quad y_1 = 27$$

1) Initialize the weights and the bias (only once)

$$w_1 = 5, \quad w_2 = 2, \quad b = 3$$

2) Forward Pass:

(a) Feed the data : inputs: $(x_1^{(1)}, x_2^{(1)})$ output: y_1

(b) Get the predictions for $\hat{y}_1$

$$\hat{y}_1 = w_1 x_1^{(1)} + w_2 x_2^{(1)} + b = 5 \cdot 3 + 2 \cdot 3 + 3 = 24$$

(c) Evaluate the cost $C(\hat{y})$:

$$C(\hat{y}) = (\hat{y}_1 - y_1)^2 = (24-27)^2 = 9$$

3) Backpropagation

(a) Evalute the gradients

$$\frac{\partial C}{\partial w_1} = \frac{\partial C}{\partial \hat{y}_1} \frac{\partial \hat{y}_1}{\partial w_1} = 2(\hat{y}_1 - y_1) x_1^{(1)} = 2(24-27) \cdot 3 = -18$$

$$\frac{\partial C}{\partial w_2} = \frac{\partial C}{\partial \hat{y}_1} \frac{\partial \hat{y}_1}{\partial w_2} = 2(\hat{y}_1 - y_1) x_2^{(1)} = 2(24-27) \cdot 3 = -18$$

$$\frac{\partial C}{\partial b} = \frac{\partial C}{\partial \hat{y}_1} \frac{\partial \hat{y}_1}{\partial b} = 2(\hat{y}_1 - y_1) \cdot 1 = 2(24-27) \cdot 1 = -6$$

(b) Update the weights and the bias using gradient descent.

$$w_1 = w_1 - r \frac{\partial C}{\partial w_1} = 5 - 0.01(-18) = 5.18$$

$$w_2 = w_2 - r\, \partial C / \partial w_2 = 2 - 0.01(-18) = 2.18$$

$$b = b - r \cdot \partial C/\partial b = 3 - 0.01(-6) = 3.06$$

Repeat the same procedure for the other data points except the last one. Once we are done, this will make up our first epoch. Repeat the same procedure for several epochs.

When we train our model for 100 epochs, we obtain

$\boxed{w_1 = 6.944,\ w_2 = 2.012,\ b = 0.372}$, loss = 0.07

Let's test the model for the last data point
$(4, 9) \longrightarrow 46$

$$\hat{y} = 6.944 \times 4 + 2.012 \times 9 + 0.372 \approx \boxed{46.256}$$

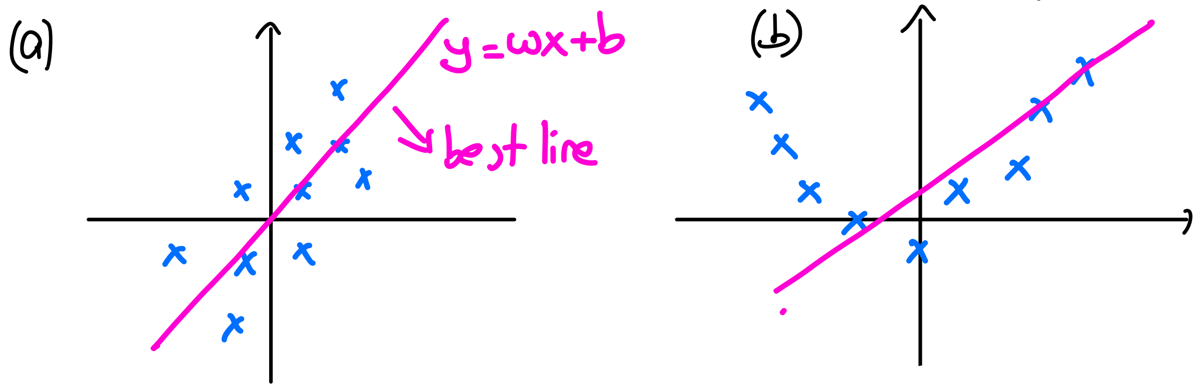
This means our model is doing a good job.

Note: (a) This is actually a toy dataset. The output is 7 times the first column + 2 times the second column + 0. In other words,

$$7X_1 + 2X_2 + 0 = Y$$

Remarks: (1) We repeat the same operations in this procedure, which begs for a more modular approach such as "automatic differentiation".

(2) As the name suggests, we can only describe "linear" or "near-linear" data with multiple linear regression:



In (a), $y=wx+b$ is a good representative of the data. In (b), no line can describe the data as it looks like $y=ax^2+bx+c$ quadratic data.

Linear regression is missing two ingredients

(1) Few number of parameters: In our original example, we have only 4 parameters (w_1, w_2, w_3 and b). Imagine our data has 1M rows. There is no way we can describe 1M data points with 4 parameters

(2) Linearity: life itself is NOT linear. As we observe

above even a couple of data points resulting from a quadratic function is a big challenge for linear regression.

These two weaknesses will be the basis of our motivation to study neural networks.

$\Downarrow$

CH3.4 Neural Networks

$$\boxed{\text{More parameters + nonlinear activation}}$$