PYTHON機器學習入門

UNIT 4：REGRESSION

授課教師：江尚瑀

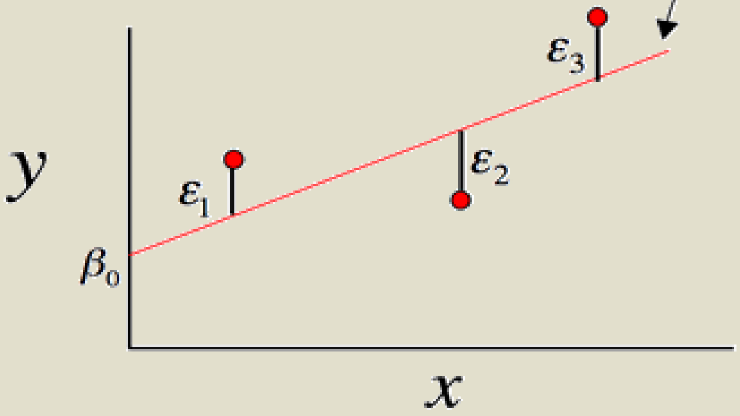
SUPERVISED LEARNING- LINEAR REGRESSION

REGRESSION 迴歸

- Definition: There exists parameters β_0, β_1, and σ^2, such that for any fixed value of the independent variable x, the dependent variable is related to x through the model equation

$$y = \beta_0 + \beta_1 x + \varepsilon$$

- ε is a rv assumed to be N(0, σ^2)



True Regression Line

$$y = \beta_0 + \beta_1 x$$

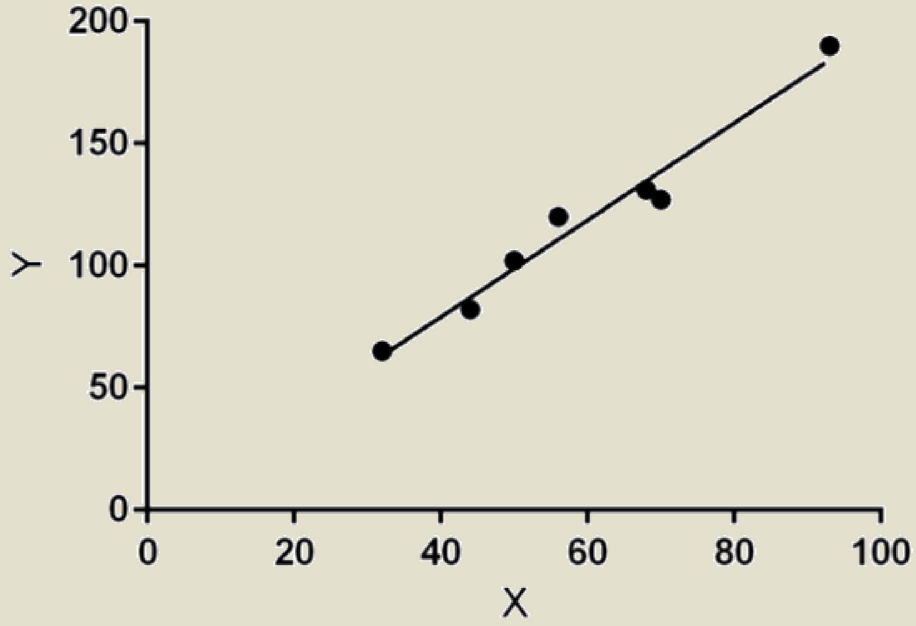
SINGLE FEATURE REGRESSION

- Single feature problem

$$\hat{Y} = \beta_0 + \beta_1 X$$

House Size (X)	House Price (Y)
50	102
70	127
32	65
68	131
93	190
44	82
56	120

house size is the single feature -> X
house price is the label -> Y



$\hat{Y} = -0.5243+1.987X$

$\hat{Y}(85)= 168.37$

MULTIREGRESSION

- Multi feature problem $\hat{Y} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3$

House Size (X_1)	Rooms (X_2)	Floor (X3)	House Price (Y)
50	2	5	102
70	2	3	127
32	1	3	65
68	3	7	131
93	4	10	190
44	2	6	82
56	3	1	120

構建多元線性回歸模型時，隨著解釋變量數目的增多，其中某兩個解釋變量之間產生多重共線性的機會就會大增。此時就需要考慮是否將其中某個變量從模型中剔除出去，甚至是重新考慮模型的構建。

共線性: 若X_1和X_2 非獨立變數,
則 β_1 也會影響 X_2

定義損失函數(LOSS FUNCTION)

好的**Regression Line** 就是誤差最小的那一條線

- 如何找出誤差最小的那條線?

$$min\ L(y_i, f(x_i))$$

- **$L(y_i, f(X_i))$** 叫做**Loss Function**或是**Cost Function. Loss**越小，就代表模型的配適性越好

不同問題或模型, 都必須明確定義出好的Loss Function

迴歸常用的LOSS FUNCTION

- MAE (Mean Absolute Error)

$$\frac{1}{m}\sum_{i=1}^{m}\left|(y_i - \hat{y}_i)\right|$$

- MSE (Mean Squared Error)

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

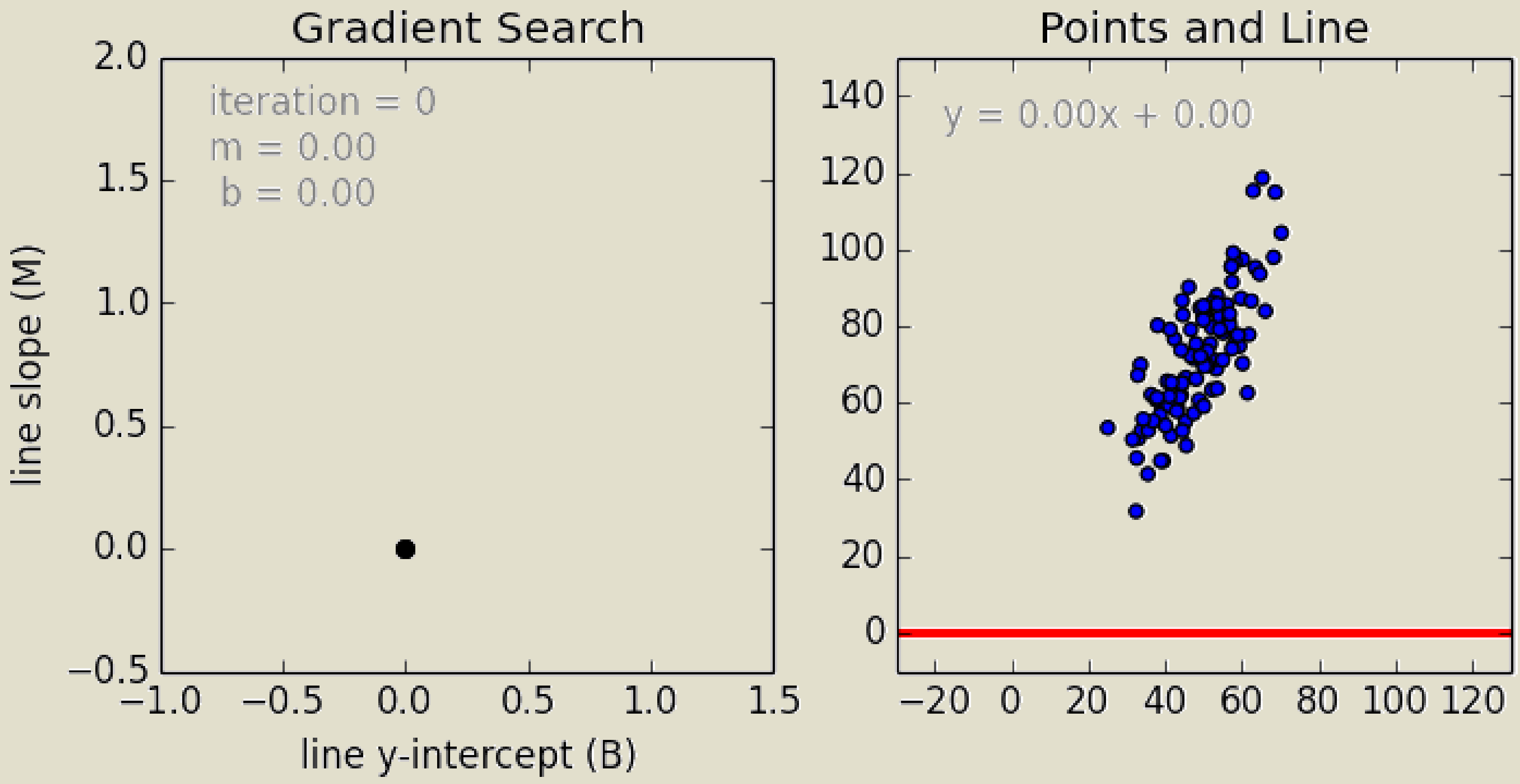
$$J(\theta) = 1/2m \sum_{i=1}^{m}(h(\theta)^{(i)} - y^{(i)})^2$$

- RMSE (Root Mean Squared Error)

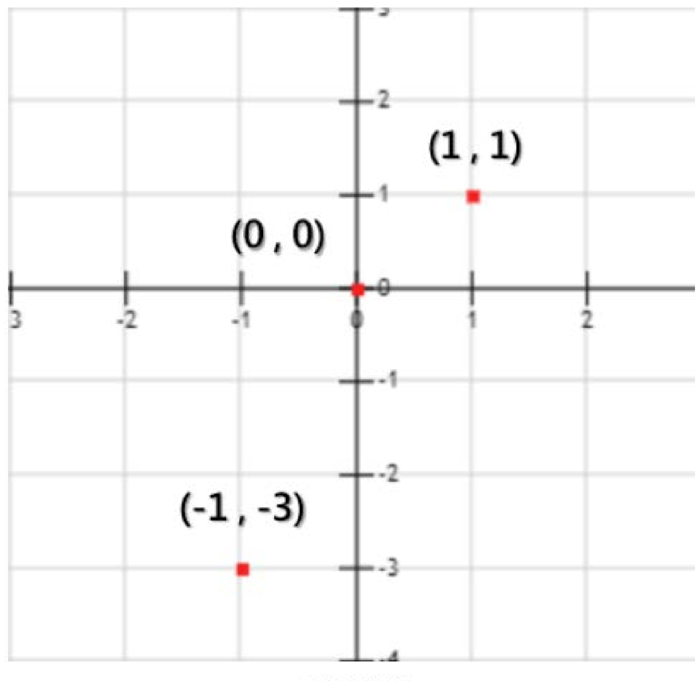
$$\sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2}$$

如何找出最佳參數使得LOSS 最小

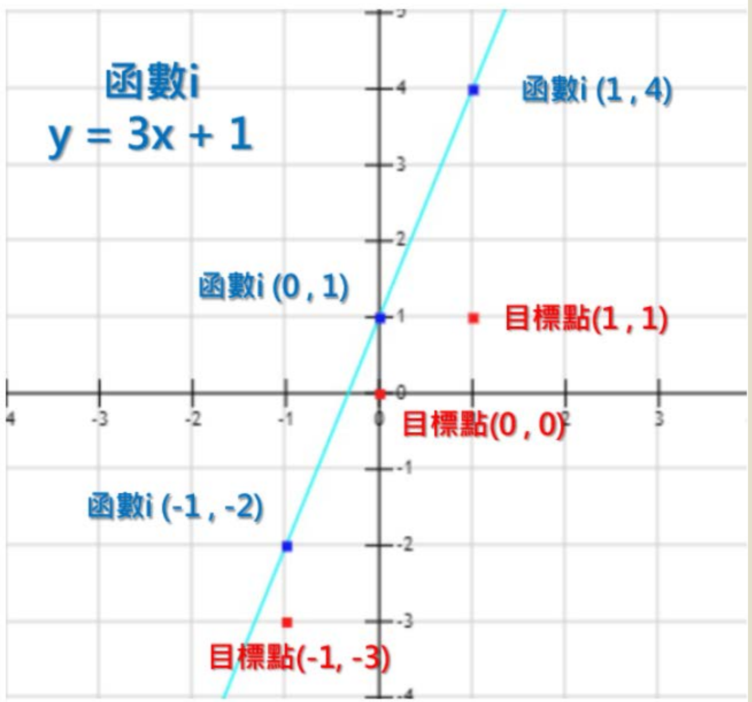
可以利用 **Gradient Descent** 方法找出最適參數B_0,B_1,B_2..., 使得Loss 最小



如何找出最佳參數使得LOSS 最小



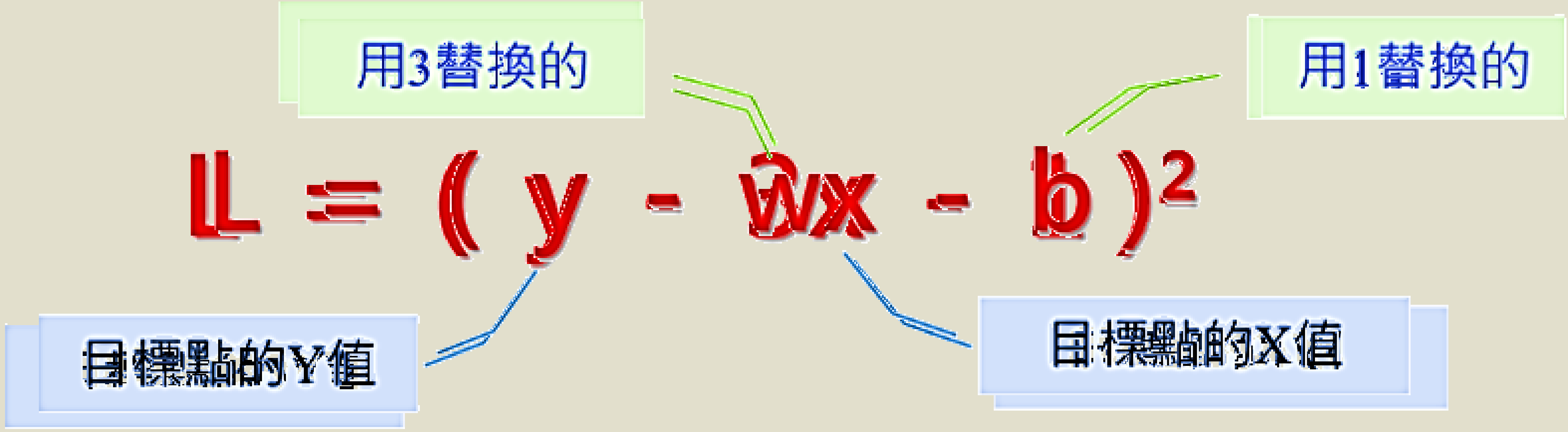
三個目標點



目標點 X值	目標點 Y值	函數i : Y = 3X + 1, X使用目標點X值帶入得到	距離 (目標點Y值-函數i 輸出值) 取平方
-1	-3	-2	1
0	0	1	1
1	1	4	9
總距離: 1 + 1 + 9 = 11			

如何找出最佳參數使得LOSS 最小

- 某目標點距離 = (某目標點的Y值 - 函數i對應某目標點X值的輸出值)²
- 總距離 = (每個目標點的Y值 - 函數i對應每個目標點X值的輸出值)²



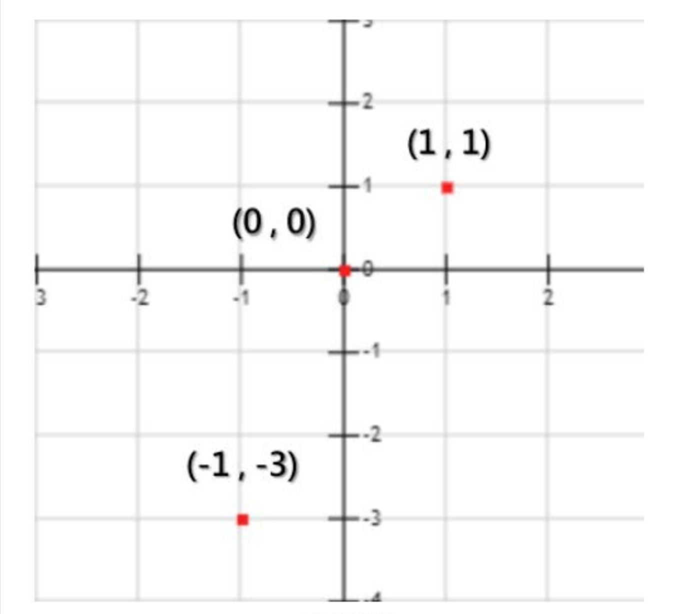
結論1:損失函數L可以幫助我們找到接近目標的新函數

https://medium.com/@arlen.mg.lu/%E6%B7%B1%E5%BA%A6%E5%AD%B8%E7%BF%92%E8%AC%9B%E4%B8%AD%E6%96%87-gradient-descent-b2a658815c72

如何找出最佳參數使得LOSS 最小

- 損失函數**L**可以幫助我們找到接近目標的新函數

$$L = (y - wx - b)^2$$



三個目標點

- **微分**某變數後產生的函數，可以指出**原函數在每點的變化**。

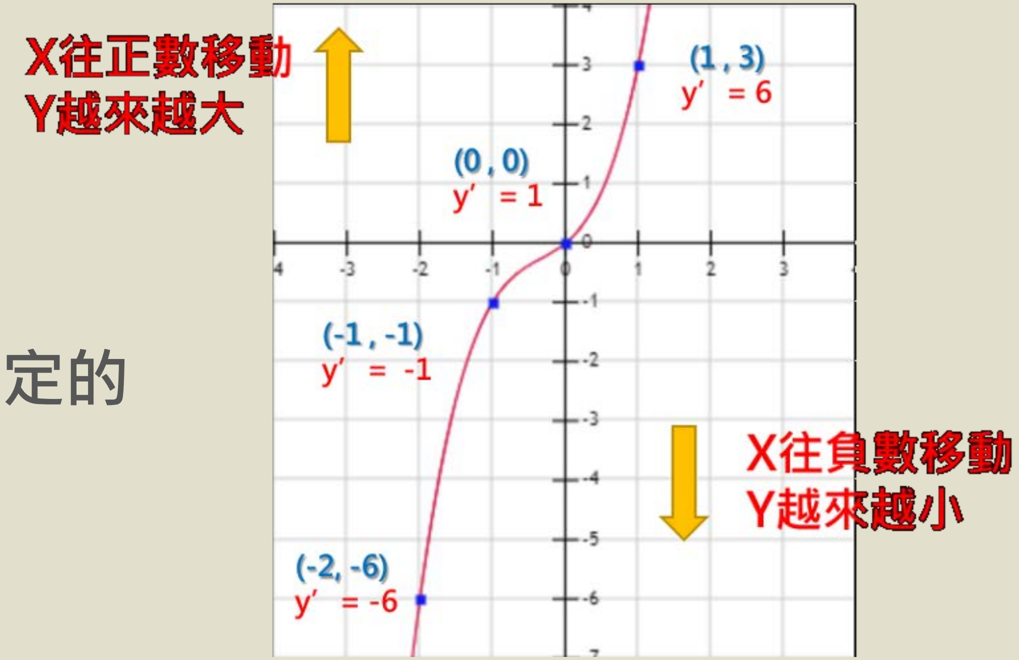
https://medium.com/@arlen.mg.lu/%E6%B7%B1%E5%BA%A6%E5%AD%B8%E7%BF%92%E8%AC%9B%E4%B8%AD%E6%96%87-gradient-descent-b2a658815c72

如何找出最佳參數使得LOSS 最小

- 微分某變數後產生的函數，可以指出原函數在每點的變化
  - 數值移動的方向的判定方法:
    將x帶入微分後的函數**y’**
  - 我們會將帶入**x**後的**y’**產生的數值
    稱之為梯度

$y = x^3 + x^2 + x$

$y' = 3x + 2x + 1$



結論2:數值移動的方向是可以從梯度判定的

如何找出最佳參數使得LOSS 最小

$$L = (y - wx - b)^2$$

- 假設x = -3 y = -1，我們該怎麼找到w跟b呢?
  - 需要分別對w跟b做微分，來得知w跟b怎麼移動才會讓$(-3 + w - b)^2$越來越小

    對W取微分後得到$(-3 + w - b)^2 = 9 - 6w + 6b - 2wb + w^2 + b^2$，$L'w = -6 - 2b + 2w$

    對b取微分後得到$L'b : 6 - 2w + 2b$

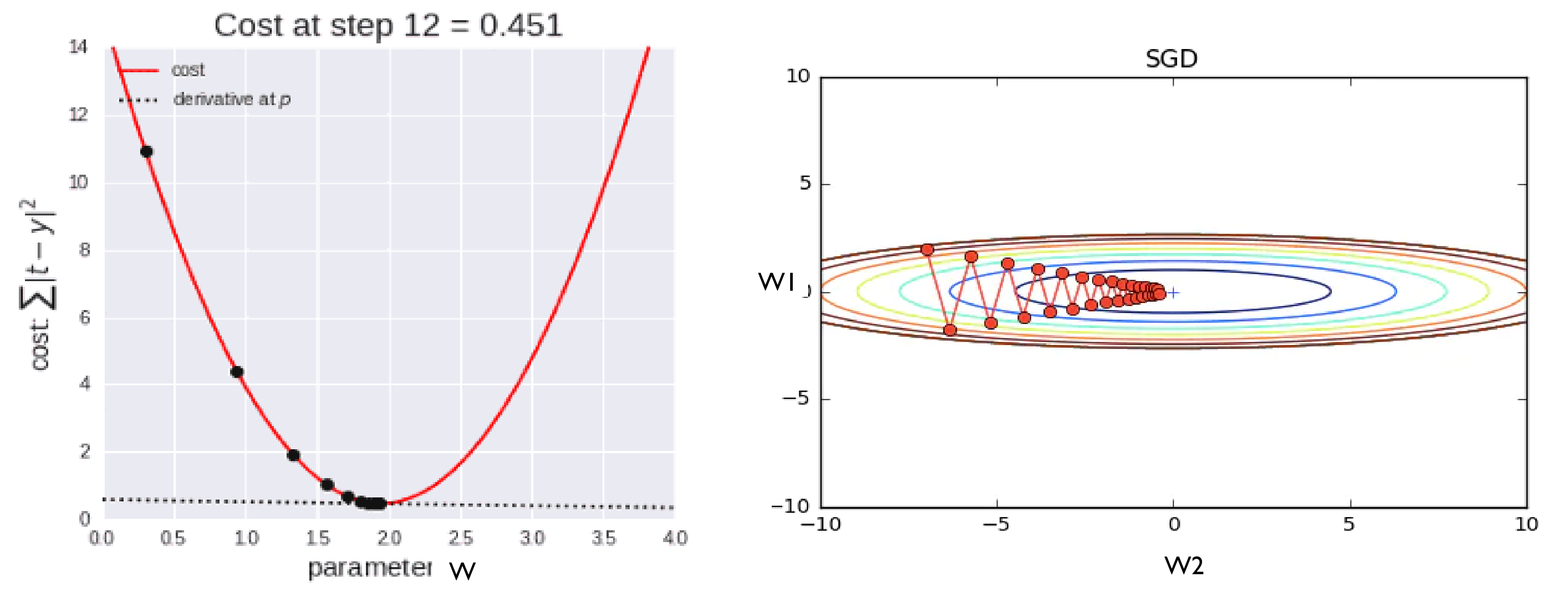
w越往**負數**方向移動，損失函數會**越小**

b越往**正數**方向移動，損失函數會**越大**

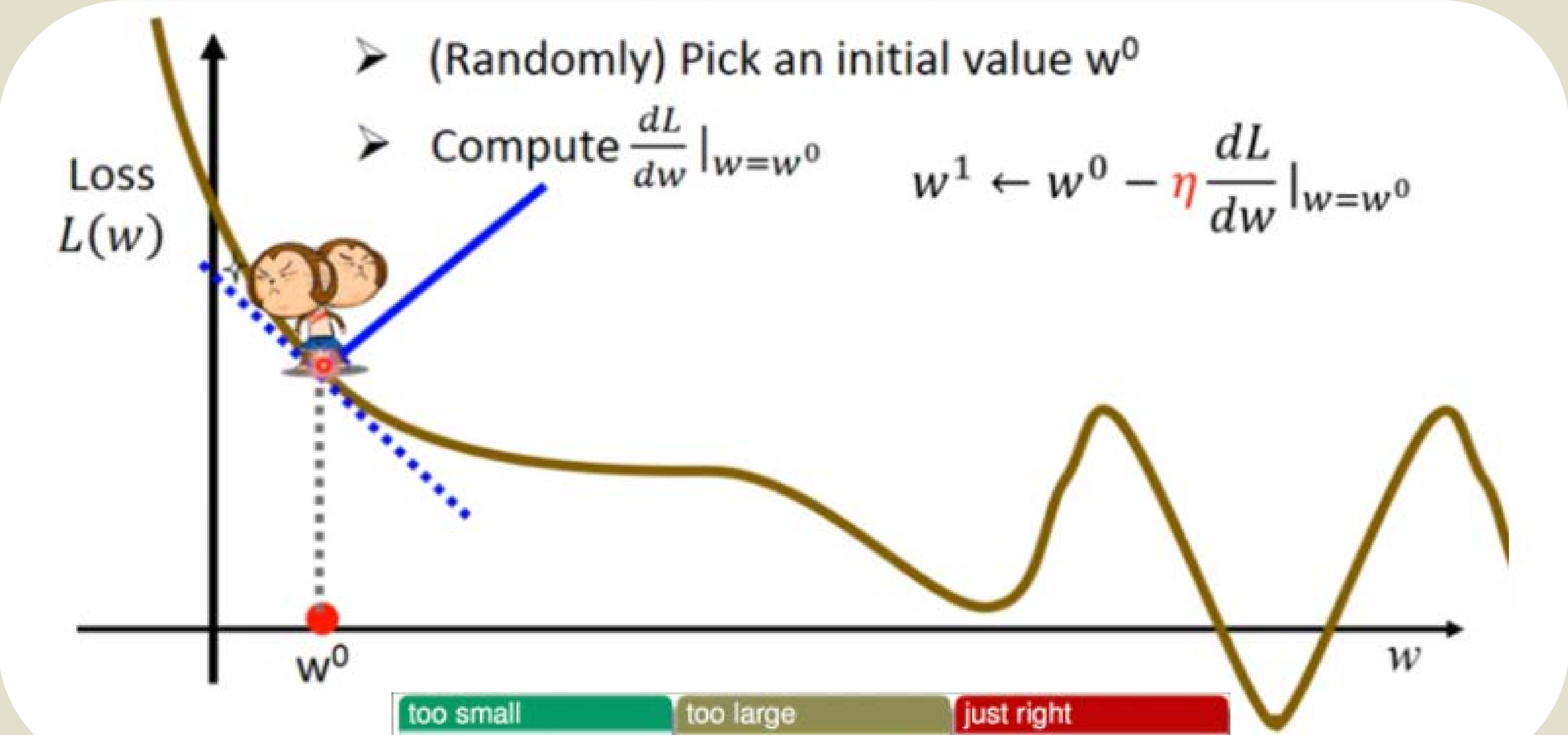
結論3：損失函數L移動的方向剛好跟梯度的方向相反

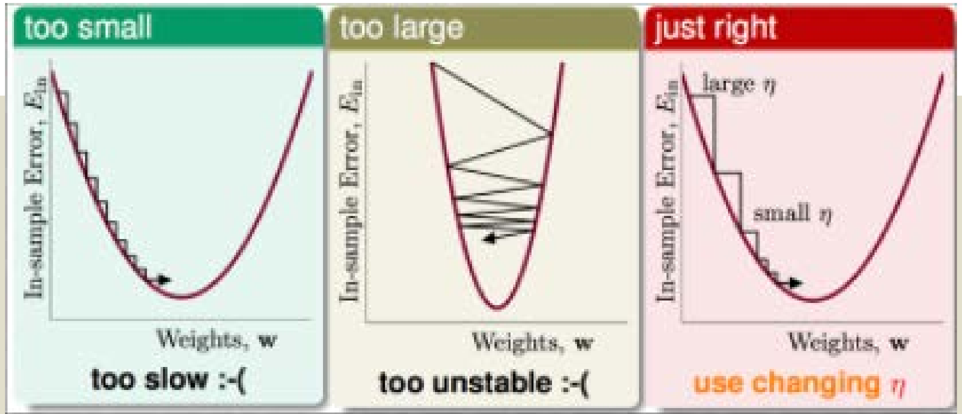
SGD(STOCHASTIC GRADIENT DESCENT)

- Choose an initial vector of parameters **W** and learning rate **η**
- Repeat until an approximate minimum is obtained:
  - Randomly shuffle samples in the training set.
  - For each sample do $W \leftarrow W - \eta \frac{\partial L}{\partial W}$



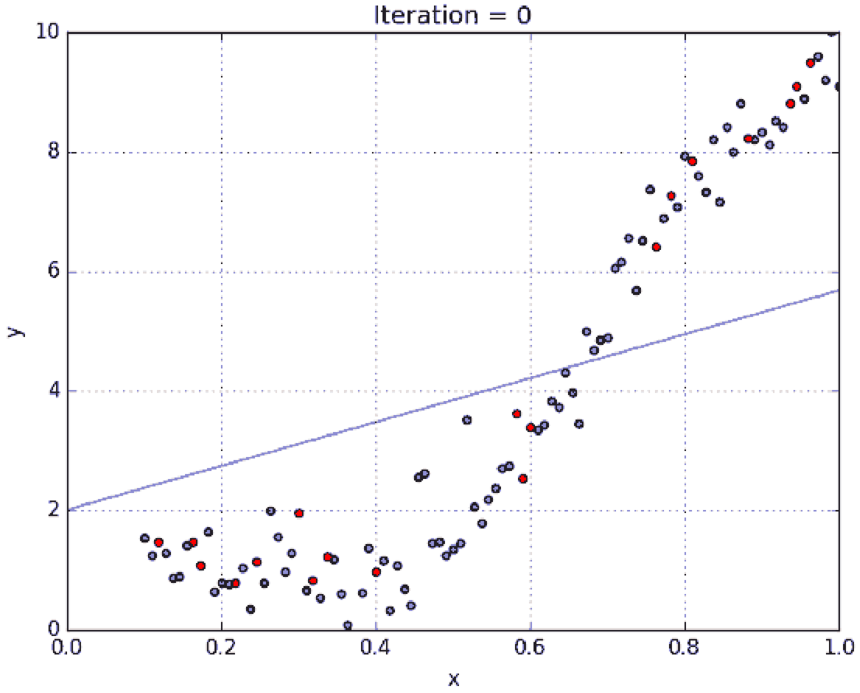
SGD(STOCHASTIC GRADIENT DESCENT)





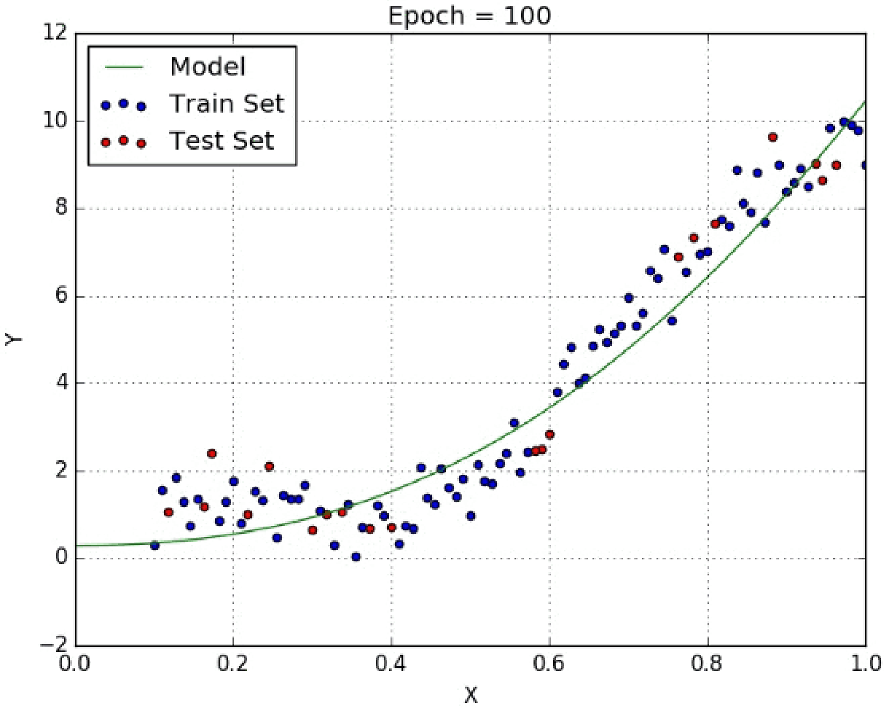
POLYNOMIAL LINEAR REGRESSION

Linear Regression



$$\hat{y} = xw + b = w_1 x_1 + b$$

Polynomial regression



$$\hat{y} = xw + b = w_1 x_1 + w_2 x_1^2 + b$$

模型泛化程度

- 泛化(**generalize**)：指**ML**模型「對未知資料集」的預測能力。

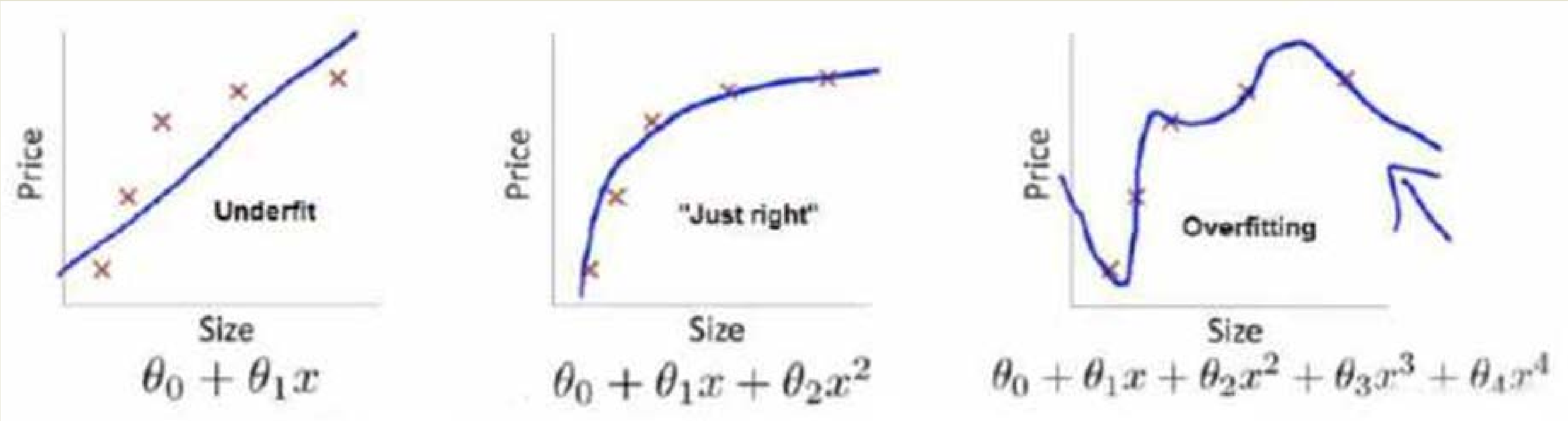
如果你的 **model** 在 **training dataset** 表現不錯，但是在 **testing dataset** 表現卻很差，那就是 **overfitting**

Overfitting

– 常常發生在 **model** 很複雜、有很多參數的時候或是 **dataset** 裡有很多 **noise** 或 **outlier**。使得表現在 **training set** 的準確率很高，但是在 **testing set** 的準確率卻很低。

Underfitting

– 通常發生在 **model** 太簡單的時候，其表現就算在 **training set** 上的錯誤率也很高。

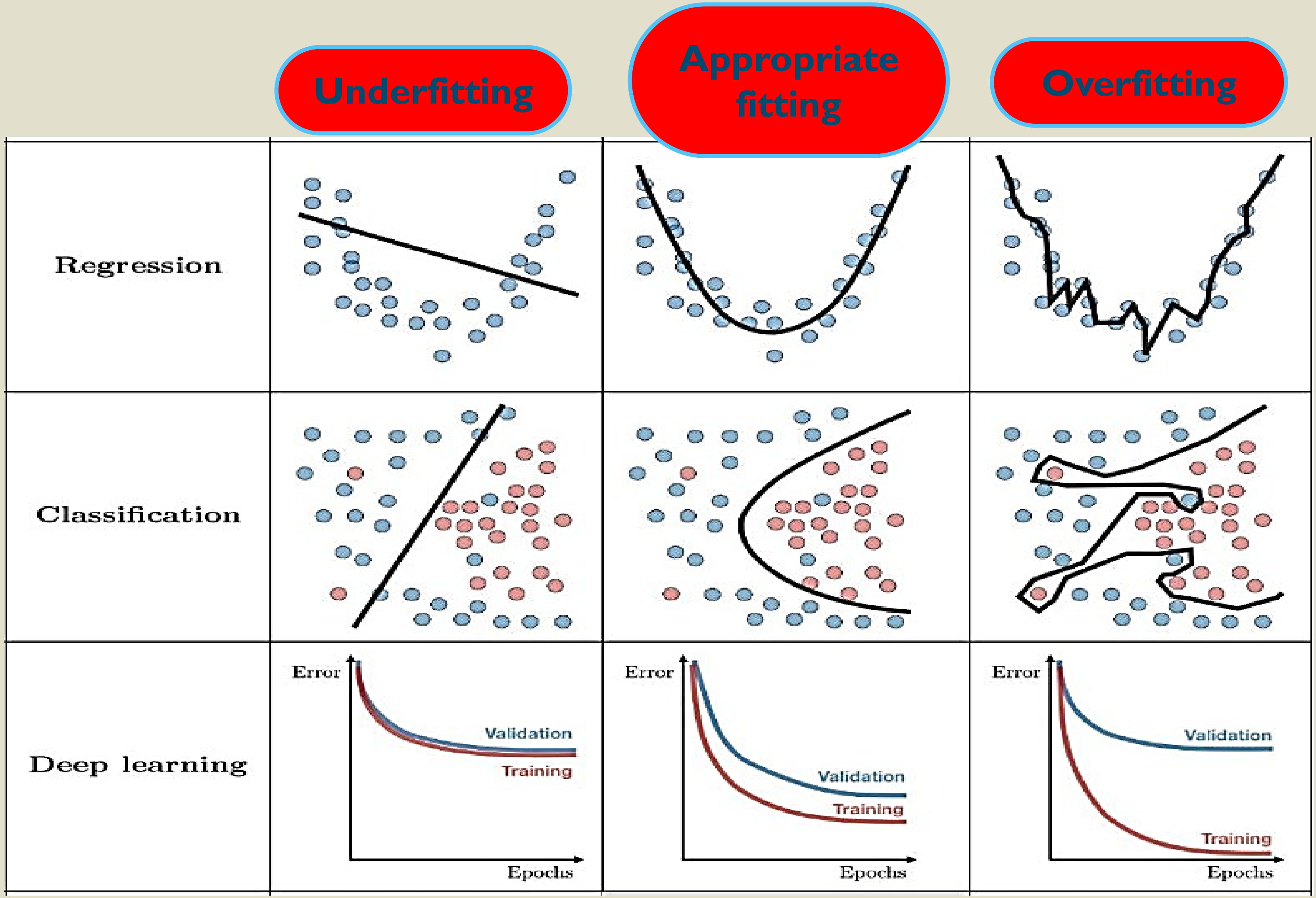


OVERFITTING ? UNDERFITTING ?

- **Underfitting**
  - is the case where the model has" not learned enough" from the training data, resulting in low generalization and unreliable predictions.
- **Overfitting**
  - is the case where the overall cost is really small, but the generalization of the model is unreliable. This is due to the model learning "too much" from the training data set.

	Underfitting	Just right	Overfitting
Symptoms	- High training error - Training error close to test error - High bias	- Training error slightly lower than test error	- Low training error - Training error much lower than test error - High variance
Remedies	- Complexify model - Add more features - Train longer		- Regularize - Get more data

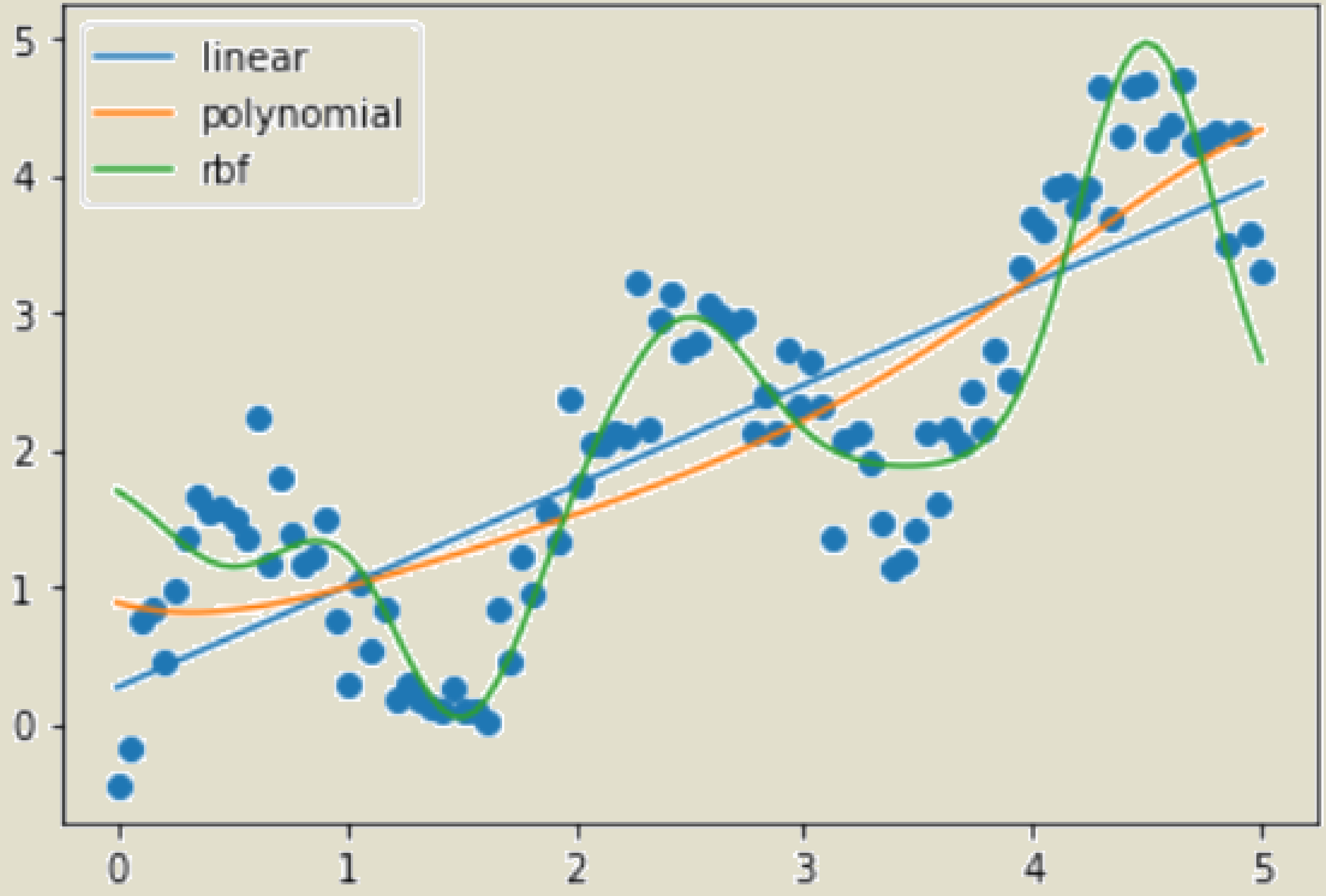
OVERFITTING ? UNDERFITTING ?



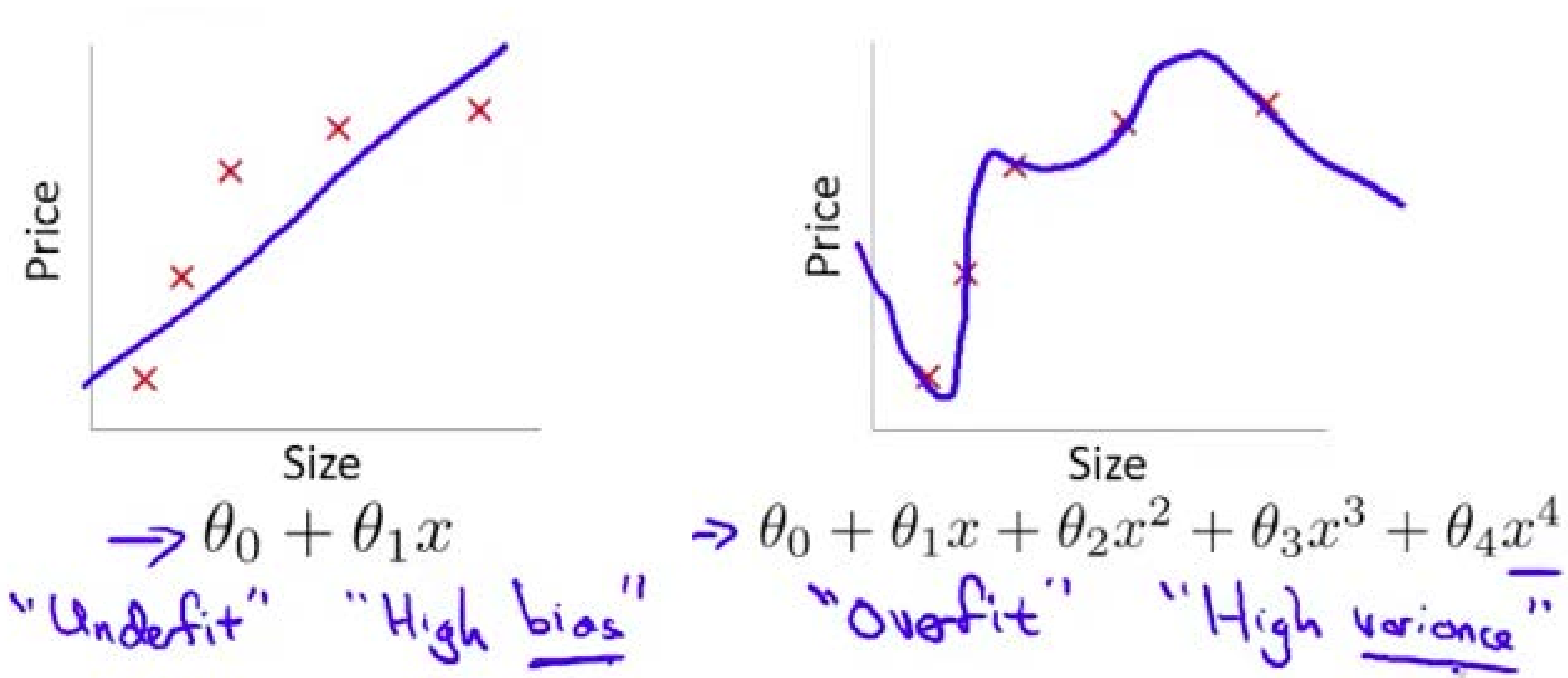
實作

- Loss Function.ipynb
- Linear Regression_1.ipynb
- Linear Regression_2.ipynb
- Linear Regression_3.ipynb

PROBLEM



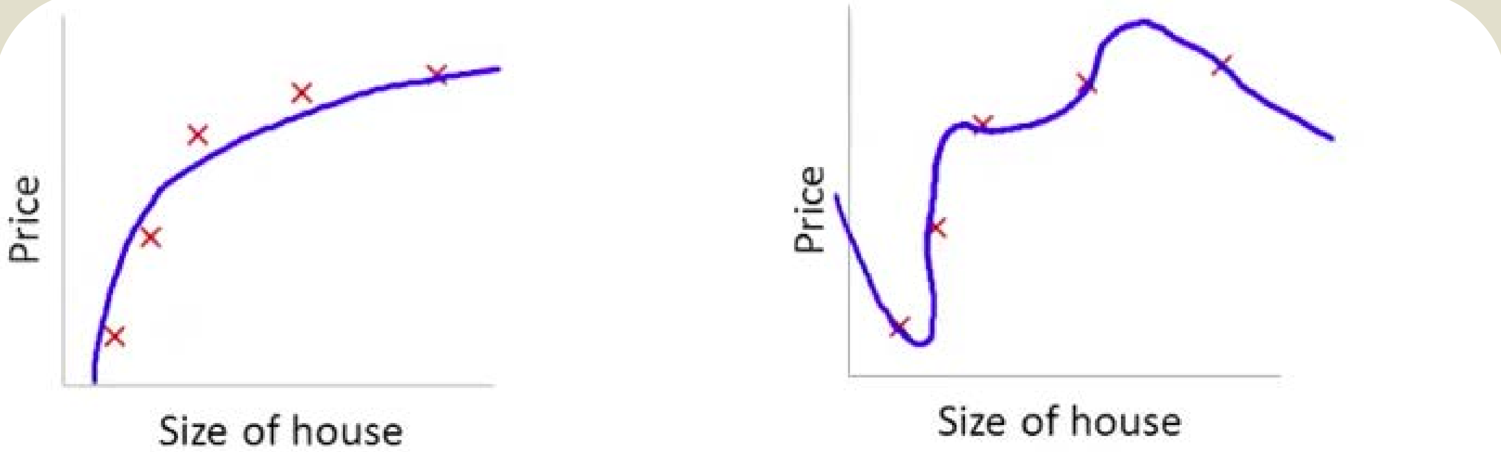
PROBLEM



Price
Size
$\rightarrow \theta_0 + \theta_1 x$
"Underfit" "High bias"
Price
Size
$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$
"Overfit" "High variance"

如何解決OVERFITTING

- **If we have too many features, the learned hypothesis may fit the training set very well (J(θ)=0), but fail to generalize to new examples(Predictions on new examples)**



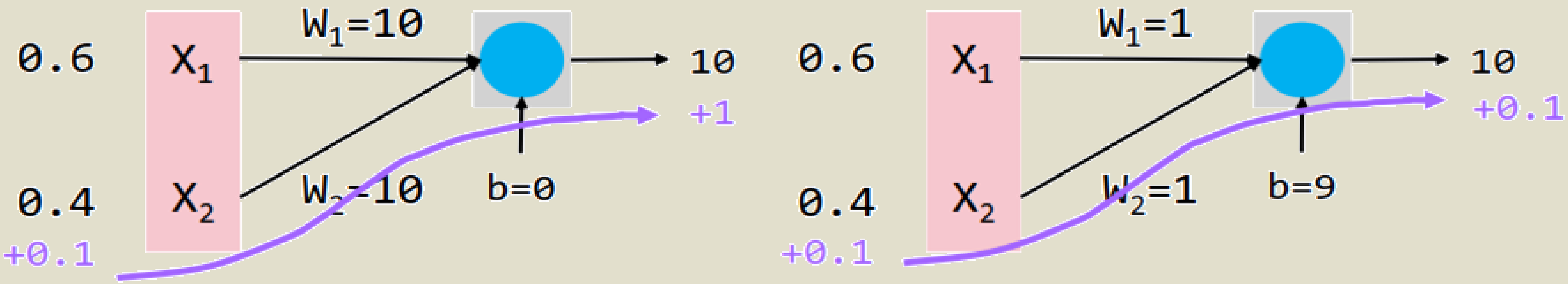
$\theta_0 + \theta_1 x + \theta_2 x^2$ $\qquad$ $\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

1. 降低**features**的數量：人工選擇、**model selection algorithm**

2. **Regularization**：維持現有的**features**，但是降低部分不重要**feature**的影響力。

REGULARIZATION

- 限制 weights 的大小讓 output 曲線比較平滑
- 為什麼要限制呢？



w_i 較小 ➔ Δx_i 對 $\hat{y}$ 造成的影響($\Delta\hat{y}$)較小
➔ 對 input 變化比較不敏感 ➔ generalization 好

REGULARIZATION

$$J(\theta) = 1/2m \sum_{i=1}^{m} (h(\theta)^{(i)} - y^{(i)})^2$$

$$\rightarrow \underbrace{\min_{\theta} \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + 1000\,\theta_3^2 + 1000\,\theta_4^2}$$

$$\theta_3 \approx 0 \qquad \theta_4 \approx 0$$

$$\theta_0 + \theta_1 x + \theta_2 x^2 + \cancel{\theta_3 x^3} + \cancel{\theta_4 x^4}$$

REGULARIZATION

$$J(\theta) = 1/2m \sum_{i=1}^{m} (h(\theta)^{(i)} - y^{(i)})^2$$

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2\right]$$

REGULARIZATION

- 為了**限制weights 的大小**，以避免落入**Overfitting**狀態，我們將**J(f)**加入損失函數中, 我們叫做**Regularization**

$$\min L(y_i, f(x_i)) + \boxed{\lambda J(f)}$$

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2\right]$$

λ是用來調整regularization 的比重
小心顧此失彼(降低weights 的大小而犧牲模型準確性)

L1 AND L2 REGULARIZERS

- **Regularization**又分兩種：
  - 第一種是**L1**正則化 **(Lasso)**，第二種是**L2**正則化 **(Ridge)**

$$\lambda J(f)$$

L1 norm (Lasso Regression) ➔ can also do feature selection

$$L_1 = \sum_{i=1}^{N} |W_i|$$

Sum of absolute values

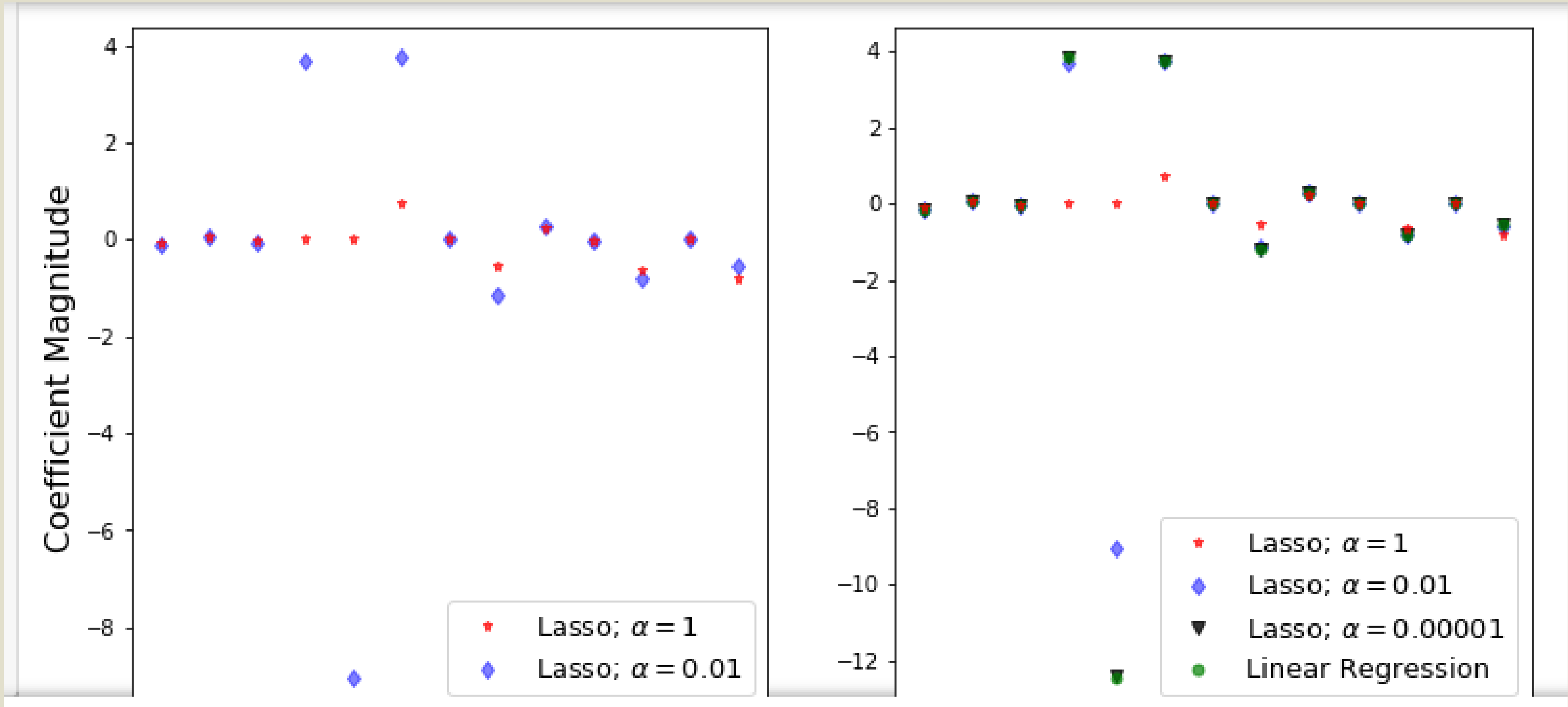
L2 norm (Ridge Regression) ➔ reduces the coefficients close to zero

$$L_2 = \sqrt{\sum_{i=1}^{N} |W_i|^2}$$

Root mean square of absolute values

實作

- **lasso_regression.ipynb**



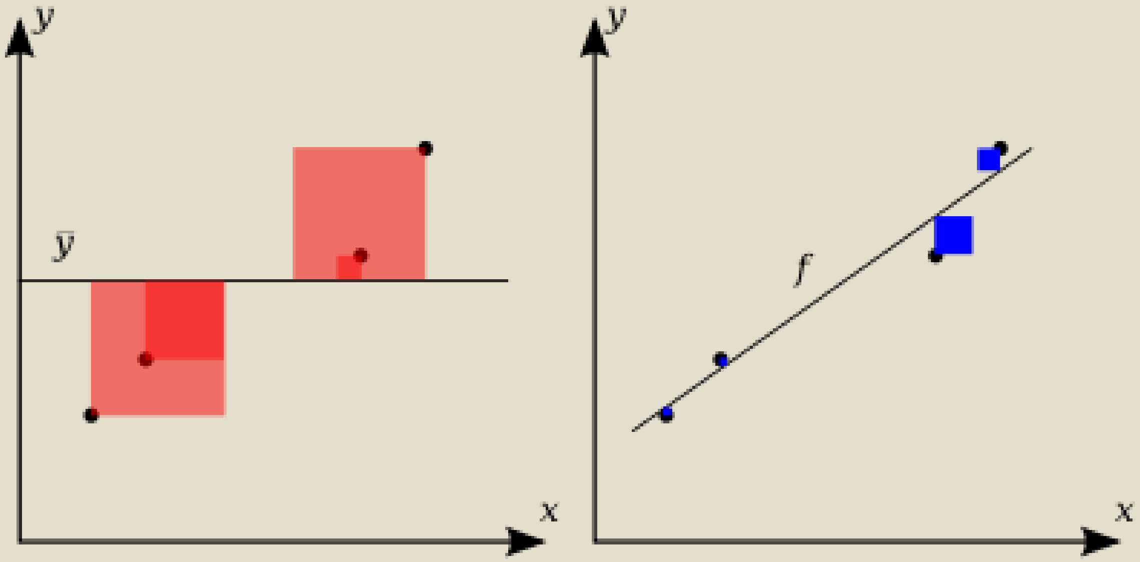
R^2

$$R^2 \equiv 1 - \frac{SS_{\rm res}}{SS_{\rm tot}}$$

$$SS_{\rm res} = \sum_i (y_i - f_i)^2 = \sum_i e_i^2$$

$$SS_{\rm tot} = \sum_i (y_i - \bar{y})^2$$

$$R^2 = 1 - \frac{MSE(\hat{y}, y)}{Var(y)}$$



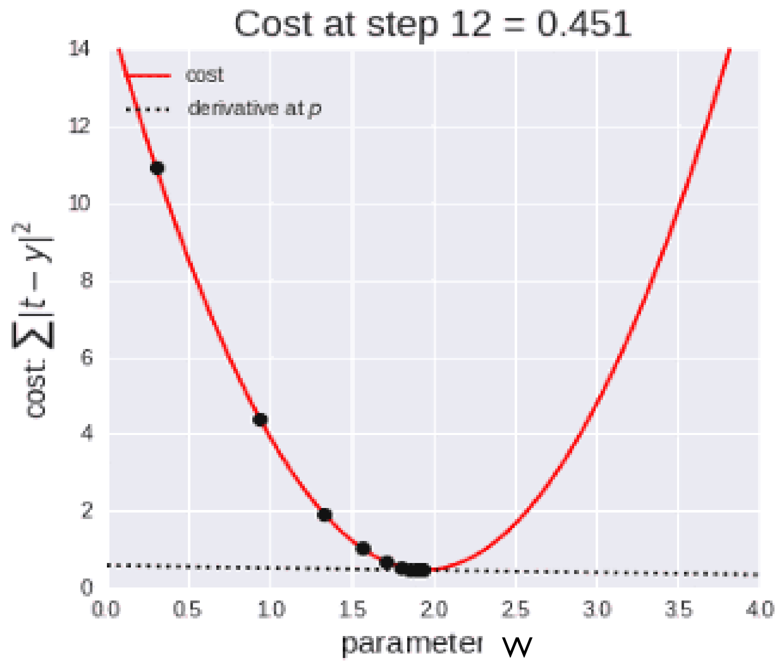
結果是1 ➔ 模型無錯誤
結果是0 ➔模型跟瞎猜差不多
結果是0~1之間的數➔ 即模型的好壞程度
結果是負數➔我們的模型還不如瞎猜

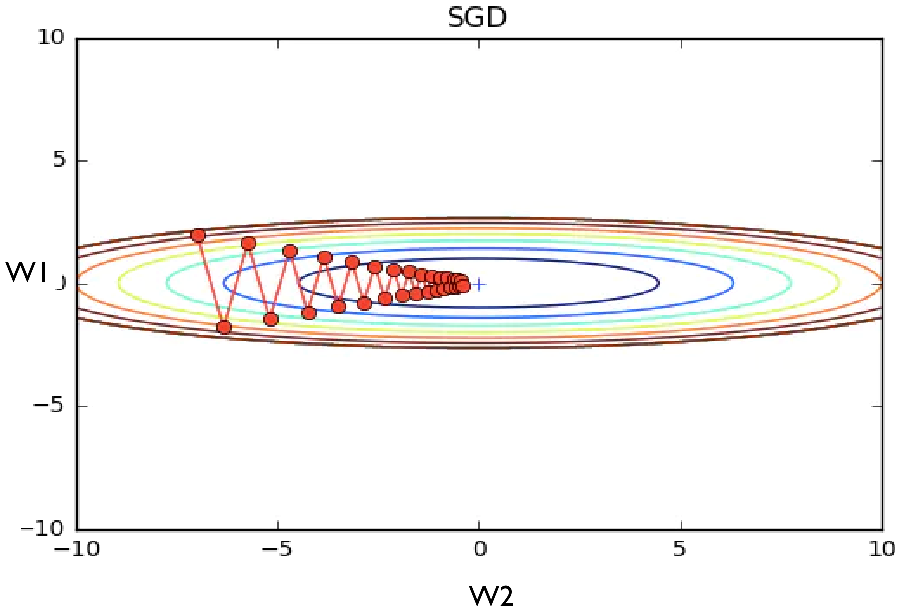
https://en.wikipedia.org/wiki/Coefficient_of_determination

APPENDIX
GRADIENT DESCENT ALGORITHM

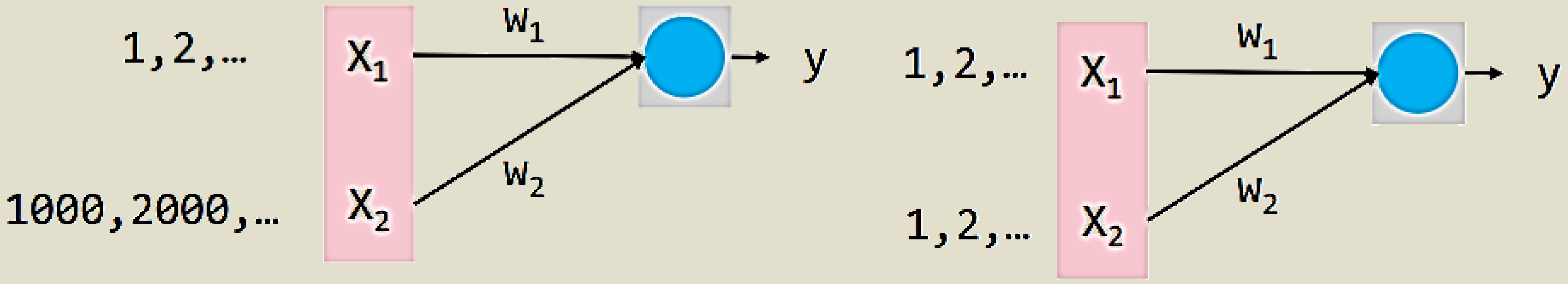
SGD(STOCHASTIC GRADIENT DESCENT)

- Choose an initial vector of parameters $\mathbf{W}$ and learning rate η
- Repeat until an approximate minimum is obtained:
  - Randomly shuffle samples in the training set.
  - For each sample do $\mathrm{W} \leftarrow \mathrm{W} - \eta \frac{\partial L}{\partial W}$

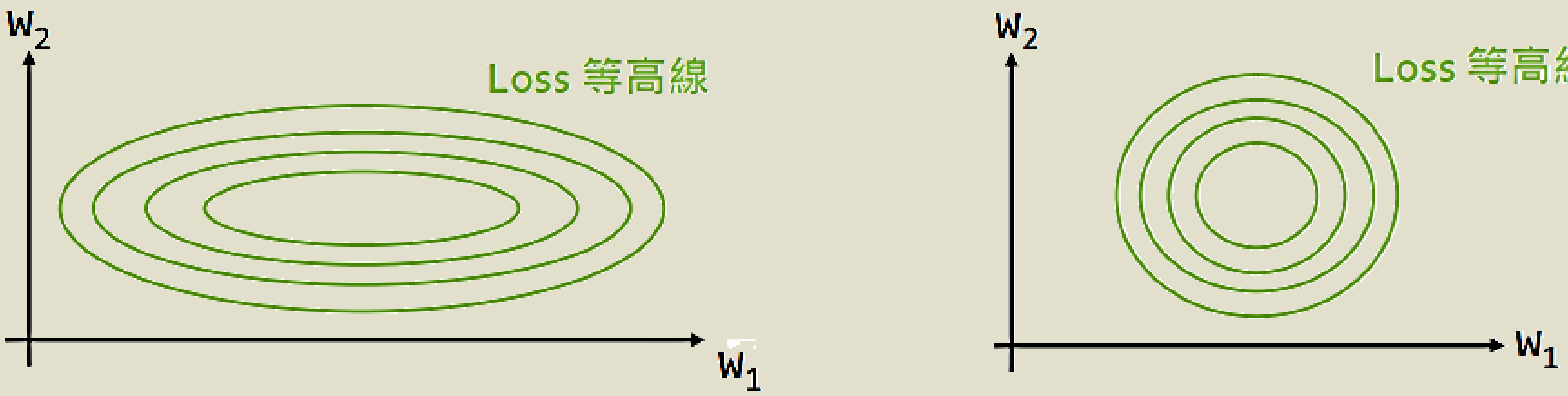




資料 RE-SCALE 對 WEIGHT 影響



w_2 的修正(ΔW)對於 loss 的影響比較大



如果不**re-scale ,** 值域特大的特徵會對**Loss**貢獻比較多
修正**W2**對**LOSS**影響比較大

有做**re-scale** 收斂速度也比較快

GRADIENTS

$Y=XW^T+b$

$$X = \begin{bmatrix} x_1^1 & x_2^1 \\ x_1^2 & x_2^2 \end{bmatrix} \quad w = \begin{bmatrix} w_1 & w_2 \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

$$\hat{Y} = XW^T + b$$

$$\hat{Y} = \begin{bmatrix} x_1^1 w_1 + x_2^1 w_2 + b_1 \\ x_1^2 w_1 + x_2^2 w_2 + b_2 \end{bmatrix} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \end{bmatrix}$$

np.dot(X,w.T)+b

Bias 這一項其實可以直接在X加上一欄 “1”

$$X = \begin{bmatrix} 1 & x_1^1 & x_2^1 \\ 1 & x_1^2 & x_2^2 \end{bmatrix}$$

2x3

np.c_[np.ones((len(X),1)),X]

$$w = \begin{bmatrix} w_0 & w_1 & w_2 \end{bmatrix}$$

$$\begin{bmatrix} w_0 + x_1^i w_1 + x_2^i w_2 \\ w_0 + x_1^i w_1 + x_2^i w_2 \end{bmatrix}$$

np.dot(X,w.T)

GRADIENTS

Loss function: MSE

$$loss(w) = 1/m \sum_{i=1}^{m} (\hat{y}^i - y^i)^2$$

```
loss=np.dot(X,w.T)-y
 1/m*np.sum(np.square(loss))
```

gradient

$$\begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} - \eta \begin{bmatrix} \frac{\partial}{\partial w0} 1/m \sum_{i=1}^{m} (w_0 + x_1^i w_1 + x_2^i w_2 - y_i)^2 \\ \frac{\partial}{\partial w1} 1/m \sum_{i=1}^{m} (w_0 + x_1^i w_1 + x_2^i w_2 - y_i)^2 \\ \frac{\partial}{\partial w2} 1/m \sum_{i=1}^{m} (w_0 + x_1^i w_1 + x_2^i w_2 - y_i)^2 \end{bmatrix} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} - \eta \begin{bmatrix} 2/m \sum_{i=1}^{m} (w_0 + x_1^i w_1 + x_2^i w_2 - y_i)1 \\ 2/m \sum_{i=1}^{m} (w_0 + x_1^i w_1 + x_2^i w_2 - y_i)x_1^i \\ 2/m \sum_{i=1}^{m} (w_0 + x_1^i w_1 + x_2^i w_2 - y_i)x_2^i \end{bmatrix}$$

$$w_j := w_j - \eta(2/m \sum_{i=1}^{m} (\hat{y}^i - y^i).\, x_j^i)$$

$$\boxed{w = w - \eta(2/m)\frac{\partial Loss}{\partial w} = w - \eta(2/m)(X^T \cdot Loss)}$$

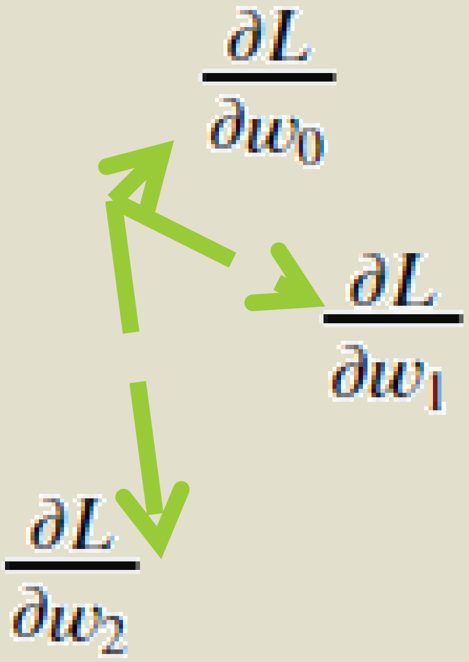
```
w=w- l*2/m*np.dot(X.T,loss)
```

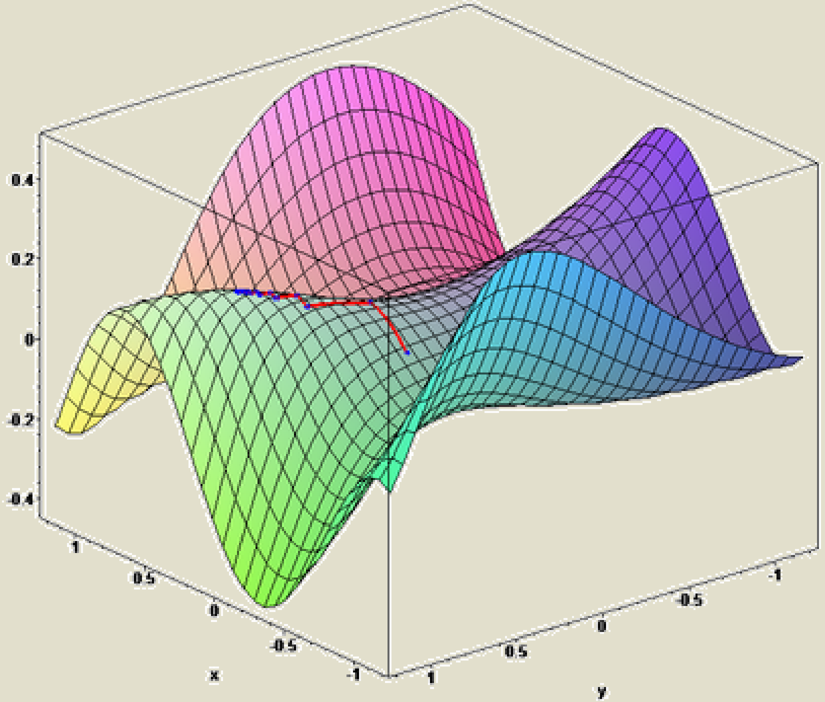
$$\underbrace{\begin{bmatrix} 1 & 1 \\ x_1^1 & x_1^2 \\ x_2^1 & x_2^2 \end{bmatrix}}_{3x2} \cdot \underbrace{\begin{bmatrix} Loss_1 \\ Loss_2 \end{bmatrix}}_{2x1} = \underbrace{\begin{bmatrix} \frac{\partial L}{\partial w_0} \\ \frac{\partial L}{\partial w_1} \\ \frac{\partial L}{\partial w_2} \end{bmatrix}}_{3x1}$$

X^T Loss Gradient

GRADIENT

- Gradient is a derivative of a function at a certain point.
- 每一變數的偏微分值，代表函數中該變數在某一點所看到坡度。不同變數Wi代表不同面向, 而偏微分值的大小是傾斜程度





Gradient Vector 描述了每一個人自己對於谷底位置的認知強度

GRADIENT實作

- 調整**Hyperparameters**，例如 **x_start**、**epochs**、**lr**，測試逼近的過程。