

ML Lecture 1: Regression - Case Study

臺灣大學人工智慧中心 科技部人工智慧技術暨全幅健康照護聯合研究中心 <http://ai.ntu.edu.tw>

Regression 的應用

- 股票預測
 - | Input : 過去各股票起伏的資料及各公司的資料 \Rightarrow Output : 預測明天的道瓊工業指數
- 自駕車
 - | Input : 車子各個 sensor 感應到的資料 \Rightarrow Output : 方向盤的角度
- 推薦系統：
 - | Input : 一位使用者 A 和商品 B \Rightarrow Output : 使用者 A 購買 B 的可能性

Regression: Output a scalar

- Stock Market Forecast

$$f(\text{圖像}) = \text{Dow Jones Industrial Average at tomorrow}$$

- Self-driving Car

$$f(\text{圖像}) = \text{方向盤角度}$$

- Recommendation

$$f(\text{使用者 A}, \text{商品 B}) = \text{購買可能性}$$

預測寶可夢的 cp 值

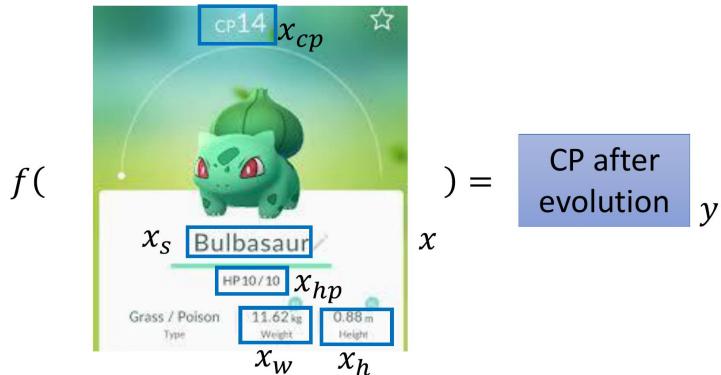
- 原因：如果可以預測寶可夢進化後的 cp 值，就可以知道抓到之後，要進化他還是犧牲他，做最有效的利用
- 期待：Input 一隻寶可夢 \Rightarrow Output 進化後的 cp 值
- 定義：寶可夢的各數值

x 代表「寶可夢」、 x_s 代表屬於哪個「物種」、 x_{hp} 代表「生命值」

x_w 代表「重量」、 x_h 代表「高度」、 x_{cp} 表示「CP值」、 y 表示「進化後的 CP 值」

Example Application

- Estimating the Combat Power (CP) of a pokemon after evolution



Step 1. 找合適的 model

- 假設 $y = b + w * x_{cp}$

 | y 為進化後的 cp 值、 b 為一常數、 w 為一數值、 x_{cp} 為進化前的 cp 值

 | w, b 可填入不同的數字，得到無窮多不同的 function，例

 | $f_1 : y = 10.0 + 9.0 * x_{cp}$ $f_2 : y = 9.8 + 9.2 * x_{cp}$ $f_3 : y = -0.8 - 1.2 * x_{cp} \dots$

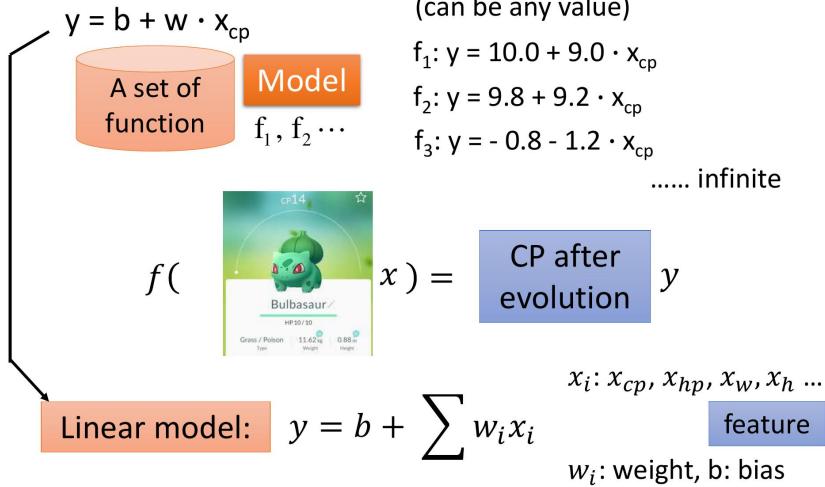
再由 training data 判斷哪個是合理的 function

- Linear model : $y = b + \sum w_i x_i$

 | x_i 為 input 寶可夢的各種不同屬性，是從 input object 裡面抽出的數值，稱之為 **feature**

 | w_i 和 b 分別為 weight 和 bias

Step 1: Model



Step 2. 判斷 function 的好壞

1. 收集 training data :

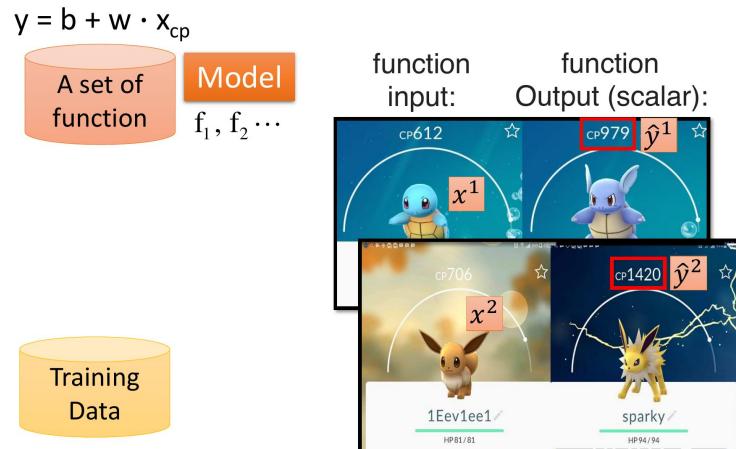
- 此為 supervised learning 的 task : 收集 function 的 input 和 output
- 此為 regression 的 task : function 的 output 是一個數值

例 :

傑尼龜以 x^1 表示，進化後卡咪龜的 cp 值 **979**，為 x^1 的 output，以 \hat{y}^1 表示 ($\hat{y}^1 = 979$)

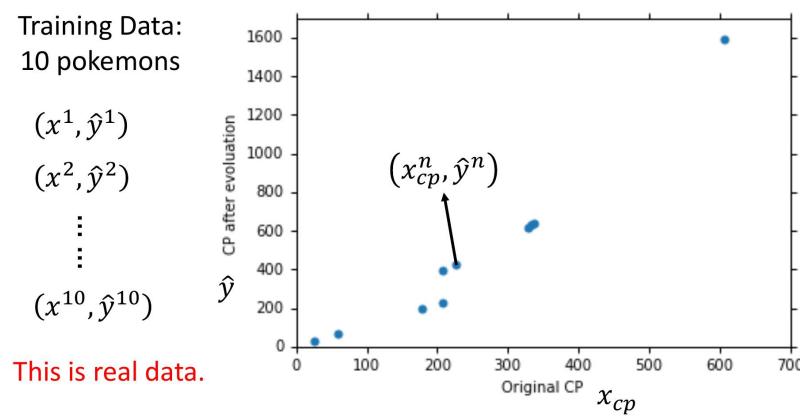
伊布以 x^2 表示，進化後雷精靈的 cp 值 **1420**，就是 \hat{y}^2

Step 2: Goodness of Function



- 收集 10 隻神奇寶貝： $(x^1, \hat{y}^1), (x^2, \hat{y}^2), \dots, (x^{10}, \hat{y}^{10})$ 以「進化前的 cp 值」為橫軸，「進化後的 cp 值」為縱軸繪圖，如下圖所示

Step 2: Goodness of Function



Source: <https://www.openintro.org/stat/data/?data=pokemon>

2. 判斷 function 好壞：

- Loss function (L)** 定義：Input 一個 function, f ，Output 這個 function 的好壞，以數值表示

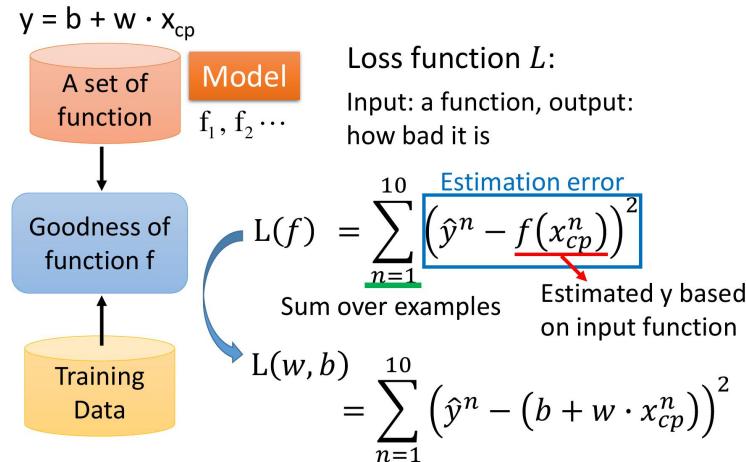
$L(f) = L(w, b)$: 衡量 function 的好壞亦即衡量一組參數的好壞，因為一個 function 是由裡面的兩個參數 w, b 所決定

- 定義 $L : L(f) = \sum_{n=1}^{10} (\hat{y}^n - f(x_{cp}^n))^2$

將各組 w, b 實際代入 $y = b + w * x_{cp}$ 中，得預測的 y 值, $f(x_{cp}^n)$

$\hat{y}^n - f(x_{cp}^n)$ 即為估測誤差，最後 sum 起來，就是 loss function

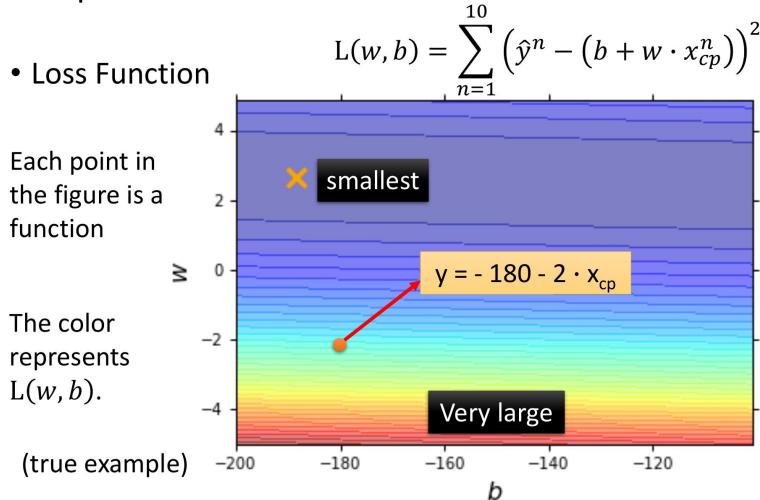
Step 2: Goodness of Function



- Loss function 作圖：

- 每一個點代表一組 (w, b)
- 顏色代表根據 loss function 判斷的這個 function 的好壞；紅色代表誤差越大，藍色代表誤差越小、function 越好

Step 2: Goodness of Function

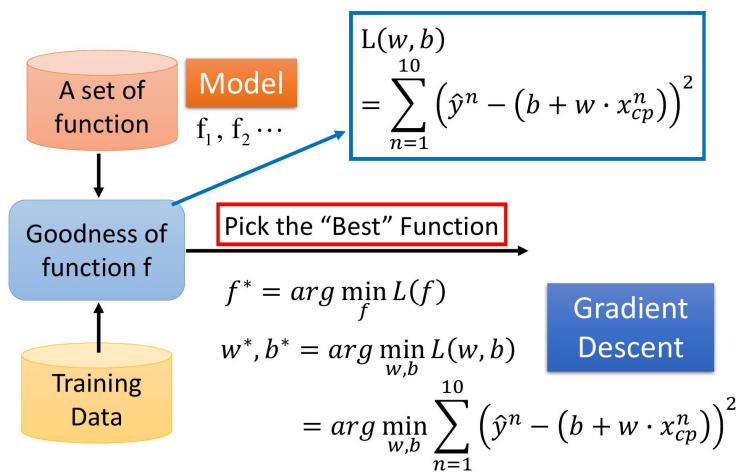


Step 3. 選出 Best Function

- 列式：

- 找最佳 function : $f^* = \arg \min_f L(f)$
- 找最佳參數 : $w^*, b^* = \arg \min_{w,b} L(w, b) = \arg \min_{w,b} \sum_{n=1}^{10} (\hat{y}^n - (b + w \cdot x_{cp}^n))^2$

Step 3: Best Function



- **Gradient Descent :**

單一參數

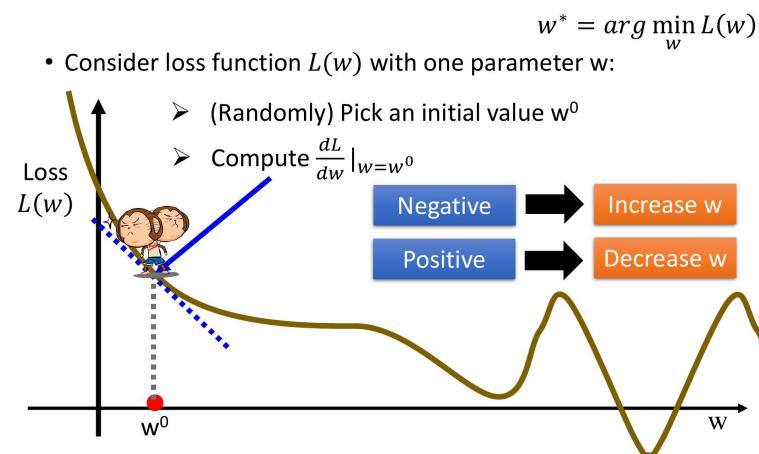
1. 隨機選取一個初始點 w^0
2. 對 w^0 作微分, 亦即計算 $\frac{dL}{dw}|_{w=w^0}$
3. 根據計算出的值

負 \Rightarrow 增加 w 的值

正 \Rightarrow 減少 w 的值

<http://chico386.pixnet.net/album/photo/171572850>

Step 3: Gradient Descent



4. 更新 w^0 的值 : $w^1 \leftarrow w^0 - \eta \frac{dL}{dw}|_{w=w^0}$

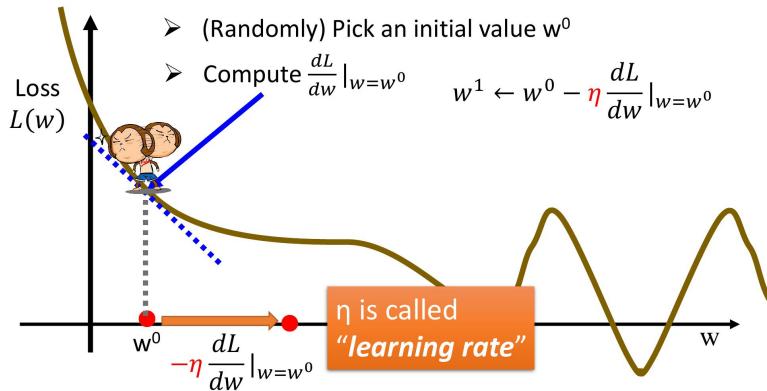
(1) 取決於 $\frac{dL}{dw}$ 的值 : $\frac{dL}{dw}$ 越大, 更新的距離就越大; 反之, 則越小

(2) 取決於 learning rate (η) : η 是一個事先決定好的值, η 越大則參數每次更新的幅度就越大; 反之, 則越小

Step 3: Gradient Descent

$$w^* = \arg \min_w L(w)$$

- Consider loss function $L(w)$ with one parameter w :

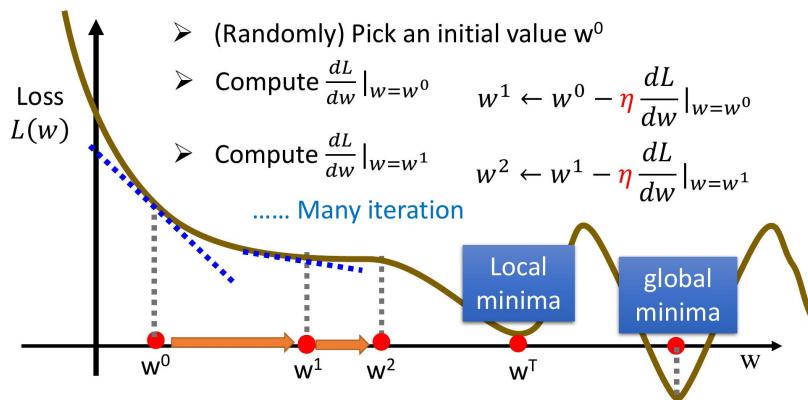


- 不斷重複步驟 4.，會走到 **local minimum** (微分是 0 的地方)，就卡住了

Step 3: Gradient Descent

$$w^* = \arg \min_w L(w)$$

- Consider loss function $L(w)$ with one parameter w :



兩個參數, Gradient, $\nabla L = \begin{bmatrix} \frac{\partial L}{\partial w} \\ \frac{\partial L}{\partial b} \end{bmatrix}$

- 隨機選取兩個初始值 w^0, b^0
- 計算 $\frac{dL}{dw}|_{w=w^0, b=b^0}, \frac{dL}{db}|_{w=w^0, b=b^0}$
- 不斷更新 w, b
例： $w^1 \leftarrow w^0 - \eta \frac{dL}{dw}|_{w=w^0, b=b^0}, b^1 \leftarrow b^0 - \eta \frac{dL}{db}|_{w=w^0, b=b^0}$
- 最後，找到一組 loss 相對較小的 w 跟 b 值，就結束了

Step 3: Gradient Descent

$$\boxed{\begin{bmatrix} \frac{\partial L}{\partial w} \\ \frac{\partial L}{\partial b} \end{bmatrix} \text{gradient}}$$

- How about two parameters? $w^*, b^* = \arg \min_{w,b} L(w,b)$

➤ (Randomly) Pick an initial value w^0, b^0

➤ Compute $\frac{\partial L}{\partial w}|_{w=w^0, b=b^0}, \frac{\partial L}{\partial b}|_{w=w^0, b=b^0}$

$$w^1 \leftarrow w^0 - \eta \frac{\partial L}{\partial w}|_{w=w^0, b=b^0} \quad b^1 \leftarrow b^0 - \eta \frac{\partial L}{\partial b}|_{w=w^0, b=b^0}$$

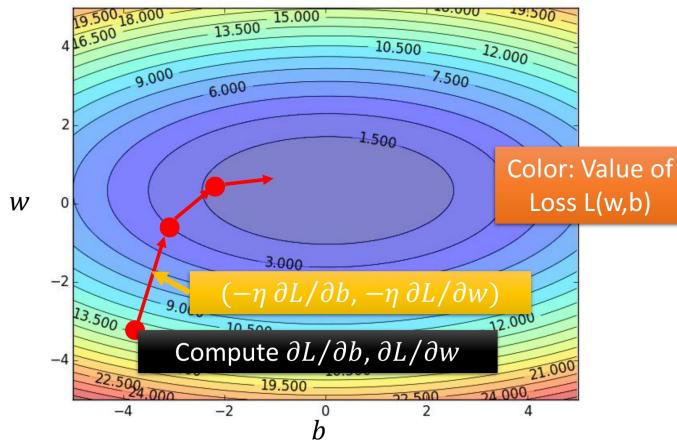
➤ Compute $\frac{\partial L}{\partial w}|_{w=w^1, b=b^1}, \frac{\partial L}{\partial b}|_{w=w^1, b=b^1}$

$$w^2 \leftarrow w^1 - \eta \frac{\partial L}{\partial w}|_{w=w^1, b=b^1} \quad b^2 \leftarrow b^1 - \eta \frac{\partial L}{\partial b}|_{w=w^1, b=b^1}$$

Visualization

1. 最左下角的紅點代表一開始隨機選的初始值
2. 計算偏微分乘上 η ，偏微分的方向即為等高線的法線方向，不斷更新參數
3. 圖上的顏色代表 loss function 的數值，越偏藍色代表 loss 越小，最後走到圖中間的地方

Step 3: Gradient Descent



實際計算 $\frac{\partial L}{\partial w}, \frac{\partial L}{\partial b}$

$$L(w, b) = \sum_{n=1}^{10} (\hat{y}^n - (b + w * x_{cp}^n))^2$$

$$\frac{\partial L}{\partial w} = \sum_{n=1}^{10} 2(\hat{y}^n - (b + w * x_{cp}^n))(-x_{cp}^n)$$

$$\frac{\partial L}{\partial b} = \sum_{n=1}^{10} 2(\hat{y}^n - (b + w * x_{cp}^n))$$

Step 3: Gradient Descent

- Formulation of $\partial L / \partial w$ and $\partial L / \partial b$

$$L(w, b) = \sum_{n=1}^{10} (\hat{y}^n - (b + w \cdot x_{cp}^n))^2$$

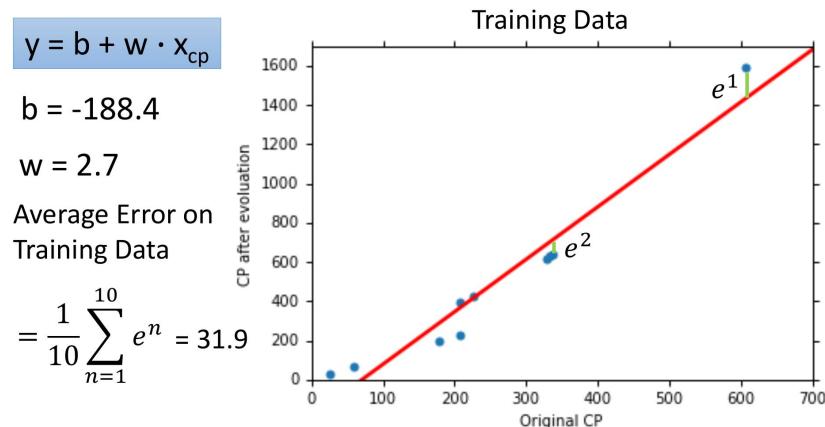
$$\frac{\partial L}{\partial w} = ? \sum_{n=1}^{10} 2(\hat{y}^n - (b + w \cdot x_{cp}^n))(-x_{cp}^n)$$

$$\frac{\partial L}{\partial b} = ? \sum_{n=1}^{10} 2(\hat{y}^n - (b + w \cdot x_{cp}^n))$$

How's the Result ?

- 圖中紅色的線是 function, $y = b + w * x_{cp}$ ，每一個藍色的點是 training data error 就是兩者之間的「距離」，此時，平均誤差為 31.9 (如下圖)

How's the results?



- 我們關心的是 **testing data** 上的誤差 (Generalization 的 case)
計算 testing data 和紅線的距離，得 35.0，比 training data 上的誤差大一點

How's the results? - Generalization

What we really care about is the error on new data (testing data)

$$y = b + w \cdot x_{cp}$$

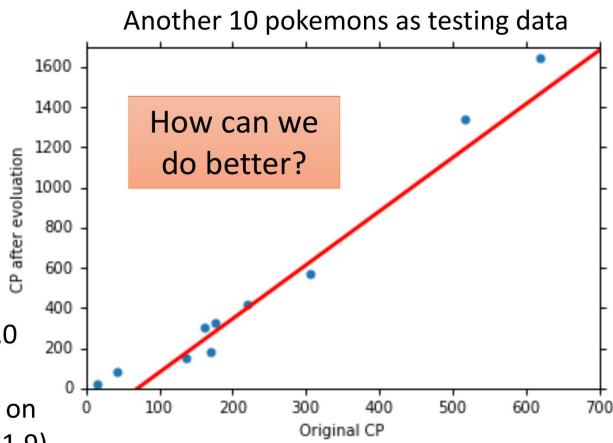
$$b = -188.4$$

$$w = 2.7$$

Average Error on Testing Data

$$= \frac{1}{10} \sum_{n=1}^{10} e^n = 35.0$$

> Average Error on Training Data (31.9)



- testing data 的誤差有沒有可能可以更小呢？

- 觀察：發現 cp 值特別小及 cp 值特別大的地方，誤差比較大
- 推想：最好的 function 可能不是一直線，可能稍微複雜一點，可能需要引入二次式
- 設計：將 model 重新設計為 $y = b + w_1 * x_{cp} + w_2 * (x_{cp})^2$ ，引入 $(x_{cp})^2$ 這一項
- 重複上面的 Step 2. 及 Step 3.，判斷 function 的好壞 以及找出最佳的 function

$b = -10.3, w_1 = 1.0, w_2 = 2.7 * 10^{-3}$ ，training data 的誤差為 15.4，testing data 的誤差為 18.4（更小了）

Selecting another Model

$$y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2$$

Best Function

$$b = -10.3$$

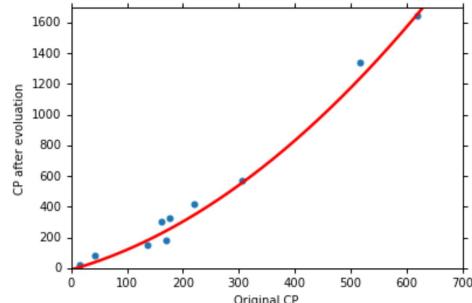
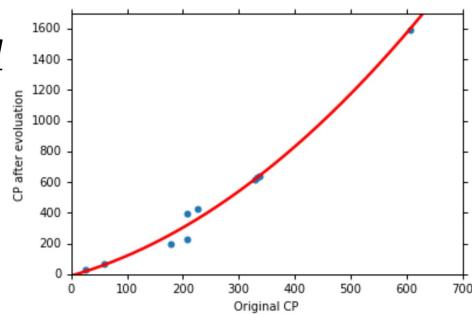
$$w_1 = 1.0, w_2 = 2.7 \times 10^{-3}$$

Average Error = 15.4

Testing:

Average Error = 18.4

Better! Could it be even better?



- 同樣的問題：有沒有可能讓誤差更小呢？

- 重複上述步驟，引入 $(x_{cp})^3$ ，得到下圖的結果，誤差為 18.1，誤差又更小了，那更複雜的 model 呢？

Selecting another Model

$$y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2 + w_3 \cdot (x_{cp})^3$$

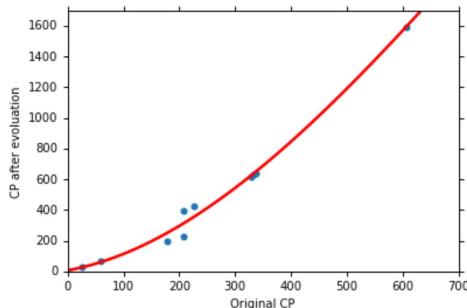
Best Function

$$b = 6.4, w_1 = 0.66$$

$$w_2 = 4.3 \times 10^{-3}$$

$$w_3 = -1.8 \times 10^{-6}$$

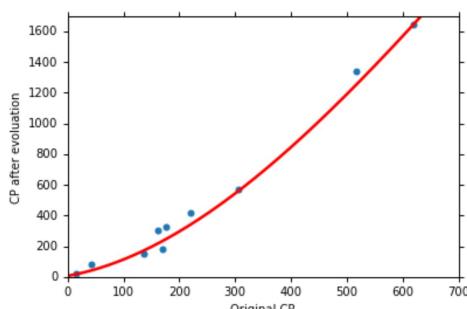
Average Error = 15.3



Testing:

Average Error = 18.1

Slightly better.
How about more
complex model?



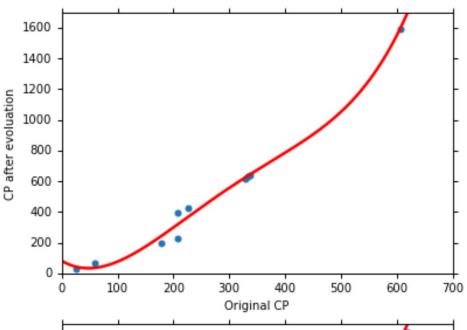
- 重複上述步驟，引入 $(x_{cp})^4$ ，得到下圖的結果，誤差為 28.8，反而變大了！？

Selecting another Model

$$y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2 + w_3 \cdot (x_{cp})^3 + w_4 \cdot (x_{cp})^4$$

Best Function

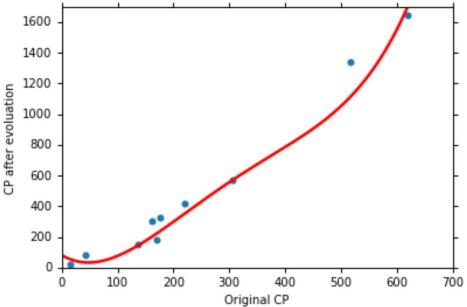
Average Error = 14.9



Testing:

Average Error = 28.8

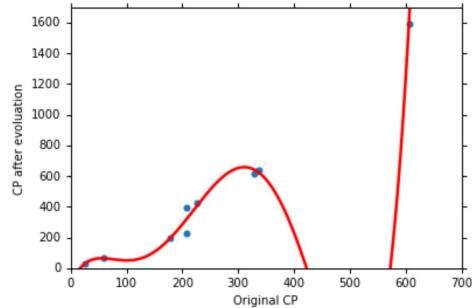
The results become
worse ...



- 重複上述步驟，引入 $(x_{cp})^5$ ，得到下圖結果，誤差為 232.1，結果又更糟，完全爛掉了。

Selecting another Model

$$y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2 + w_3 \cdot (x_{cp})^3 + w_4 \cdot (x_{cp})^4 + w_5 \cdot (x_{cp})^5$$



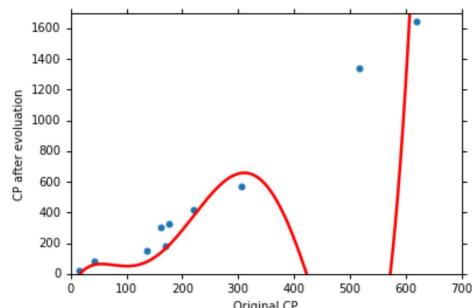
Best Function

Average Error = 12.8

Testing:

Average Error = 232.1

The results are so bad.



- 分析上述的五個 function

- 將五個 function 在 **training data** 的 average error 畫在圖上，如下圖所示

則「越複雜的 model，training data 上的 error 越小」

因為越複雜的 funciton，會包含上一層較簡單的式子，所以理論上，function 越複雜，error 越低

- 但，要在 **gradient descent** 能夠幫我們找出 **best function** 的前提下

Model Selection

1. $y = b + w \cdot x_{cp}$

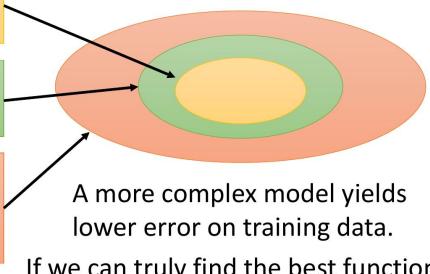
2. $y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2$

3. $y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2 + w_3 \cdot (x_{cp})^3$

4. $y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2 + w_3 \cdot (x_{cp})^3 + w_4 \cdot (x_{cp})^4$

5. $y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2 + w_3 \cdot (x_{cp})^3 + w_4 \cdot (x_{cp})^4 + w_5 \cdot (x_{cp})^5$

Training Data

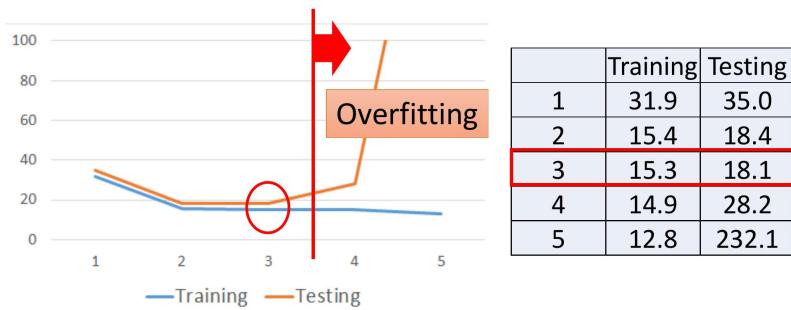


- 在 testing data 上，model 越複雜，error 越低在 第三個 function 之前是對的
但是，第四個 function 開始，error 暴增
- 所以，我們得到一個觀察

越複雜的model 可以在training data 上面給我們越好的結果；但並不一定能在testing data 上給我們越好的結果。

這件事就稱為 "overfitting"

Model Selection



A more complex model does not always lead to better performance on testing data.

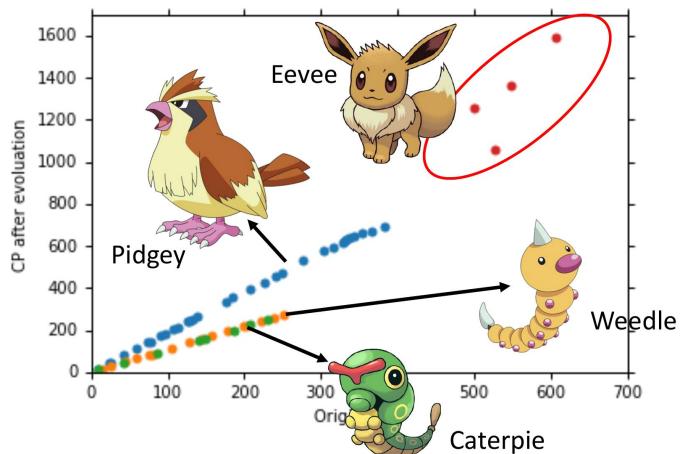
This is **Overfitting**. → Select suitable model

Collecting more data

- 當我們收集到 60 隻寶可夢的時候，會發現前面白忙一場，中間有一些 hidden factor 在影響進化後的 cp 值

其實就是寶可夢的「物種」，進化後的 cp 值受物種的影響其實很大，不應只考慮進化前的 cp 值

What are the hidden factors?



Back to Step1. 重新設計 function set

- 設計 x_S 代表 input 寶可夢的物種，不同的物種用不同的式子代表，如下圖所示

Back to step 1: Redesign the Model

$$y = b + \sum w_i x_i$$

Linear model?

x_s = species of x



If x_s = Pidgey:	$y = b_1 + w_1 \cdot x_{cp}$
If x_s = Weedle:	$y = b_2 + w_2 \cdot x_{cp}$
If x_s = Caterpie:	$y = b_3 + w_3 \cdot x_{cp}$
If x_s = Eevee:	$y = b_4 + w_4 \cdot x_{cp}$



y

- 再將式子改寫成一個 **linear function**， $y = b + \sum w_i x_i$ 藍色框框內代表：當 $x_s = \text{物種 } i$ 時，其對應到的 b_i, w_i 值不為 0；其他的參數值為 0 例： $x_s = \text{Pidgey}$ 時， $y = b_1 + w_1 * x_{cp}$

Back to step 1: Redesign the Model

$$y = b + \sum w_i x_i$$

Linear model?

$$\begin{aligned}
 y &= b_1 \cdot 1 \\
 &\quad + w_1 \cdot 1 \cdot x_{cp} \\
 &\quad + b_2 \cdot 0 \\
 &\quad + w_2 \cdot 0 \\
 &\quad + b_3 \cdot 0 \\
 &\quad + w_3 \cdot 0 \\
 &\quad + b_4 \cdot 0 \\
 &\quad + w_4 \cdot 0
 \end{aligned}$$

$\delta(x_s = \text{Pidgey})$

$$\begin{cases} =1 & \text{If } x_s = \text{Pidgey} \\ =0 & \text{otherwise} \end{cases}$$

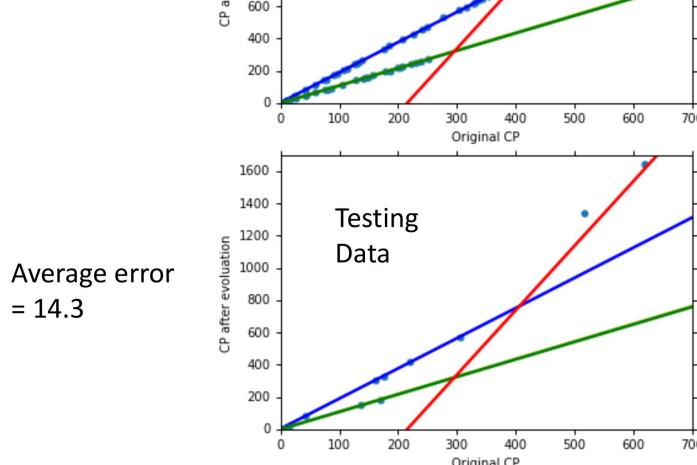
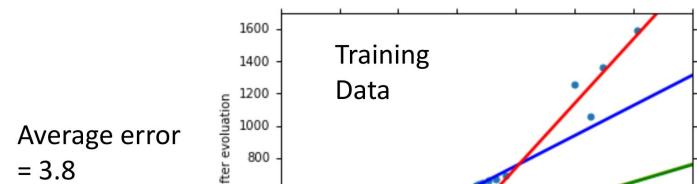
If $x_s = \text{Pidgey}$

$$y = b_1 + w_1 \cdot x_{cp}$$

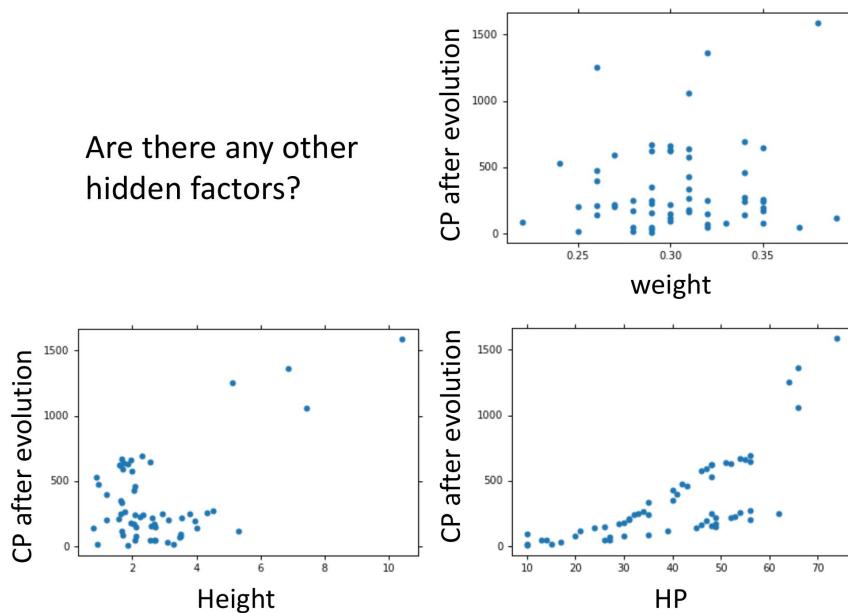
How's the result?

- 在 training data 上，不同種類的寶可夢 model 是不一樣的（如下圖，紅色是伊布，藍色是比比鳥）

所以，我們的 model 在 training data 上可以得到更低的 error



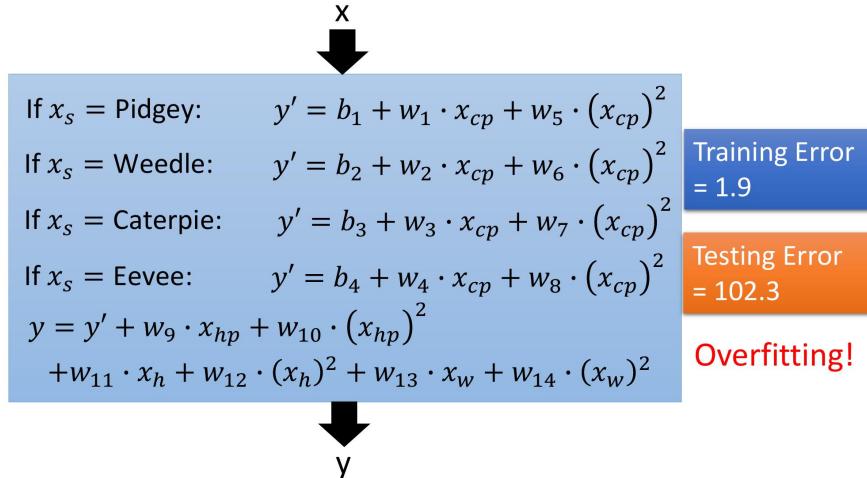
- 在 testing data 上，表現的比剛才 18 點多的 error 更好了，但再觀察一下，感覺有些東西可以做得更好
- 有一些略高略低於藍色綠色線的值，這些 **difference** 可能來自哪裡呢？
- 可能來自 weight, HP, Height.....



Back to Step1. again

- 重新設計 model，將所有有可能的參數都塞到 function 裡面
- $$y = y' + w_9 * x_{hp} + w_{10} * x_{hp}^2 + w_{11} * x_h + w_{12} * x_h^2 + w_{13} * x_w + w_{14} * x_w^2$$
- training set 上得到很好的結果：1.9，但是，testing set 上得到 102.3 (overfitting)，壞掉了

Back to step 1: Redesign the Model Again



Back to Step2. Regularization, 重新定義我們的 loss function

- 我們的 model : $y = b + \sum w_i x_i$
 原始的 loss function : $L = \sum_n (\hat{y}^n - (b + \sum w_i x_i))^2 + \lambda \sum (w_i)^2$
 只考慮了 prediction 的 error
- Regularization 再加上一項， $\lambda \sum (w_i)^2$
 這項代表，我們期待 w_i 的值越小越好。如此一來，function 會越平滑（亦即 output 對輸入的變化較不敏感）
 較不 sensitive 的 function，較不易受雜訊干擾，受到的影響較小，結果較好

Back to step 2: Regularization

$$y = b + \sum w_i x_i$$

$$L = \sum_n \left(\hat{y}^n - \left(b + \sum w_i x_i \right) \right)^2 + \lambda \sum (w_i)^2$$

The functions with smaller w_i are better

➤ Smaller w_i means ... smoother

$$y = b + \sum w_i x_i$$

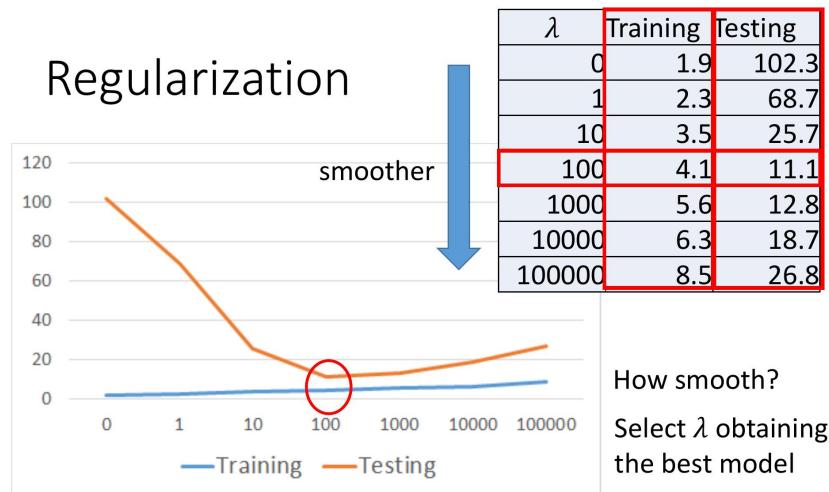
$$y + \sum w_i \Delta x_i = b + \sum w_i (x_i + \Delta x_i)$$

➤ We believe smoother function is more likely to be correct
 Do you have to apply regularization on bias?

- 做實驗，將 λ 的值由 0, 1, 10, 100...增加到 100000
 - λ 值越大，代表考慮 smooth 這項的影響力越大，得到的 function 就越平滑
 - λ 值越大，training data 上得到的 error 越大（因為傾向考慮 w 的值而減少考慮 error）

- λ 值越大，testing data 上的 error 可能會變小 ($\lambda = 100$)，但是 λ 太大時，error 又會變大 ($\lambda = 1000$)

所以，我們必須調整 λ 來決定 function 的平滑程度，找到最小的 testing error



➤ Training error: larger λ , considering the training error less

➤ We prefer smooth function, but don't be too smooth.

Conclusion

- 「寶可夢進化後的cp值」跟「進化前的cp值」以及是哪個「物種」，是非常有關係的，後兩項幾乎能決定進化後的cp值
但可能還有一些 hidden factor
- 我們有提到 Gradient Descent 及 Regularization，後面的課程會再詳細的說明

Conclusion

- Pokémon: Original CP and species almost decide the CP after evolution
 - There are probably other hidden factors
- Gradient descent
 - More theory and tips in the following lectures
- We finally get average error = 11.1 on the testing data
 - How about new data? Larger error? Lower error?
- Next lecture: Where does the error come from?
 - More theory about overfitting and regularization
 - The concept of validation