

Regression

Hung-yi Lee

李宏毅

Regression: Output a scalar

- Stock Market Forecast

$$f(\text{Image of stock market charts}) = \text{Dow Jones Industrial Average at tomorrow}$$

- Self-driving Car

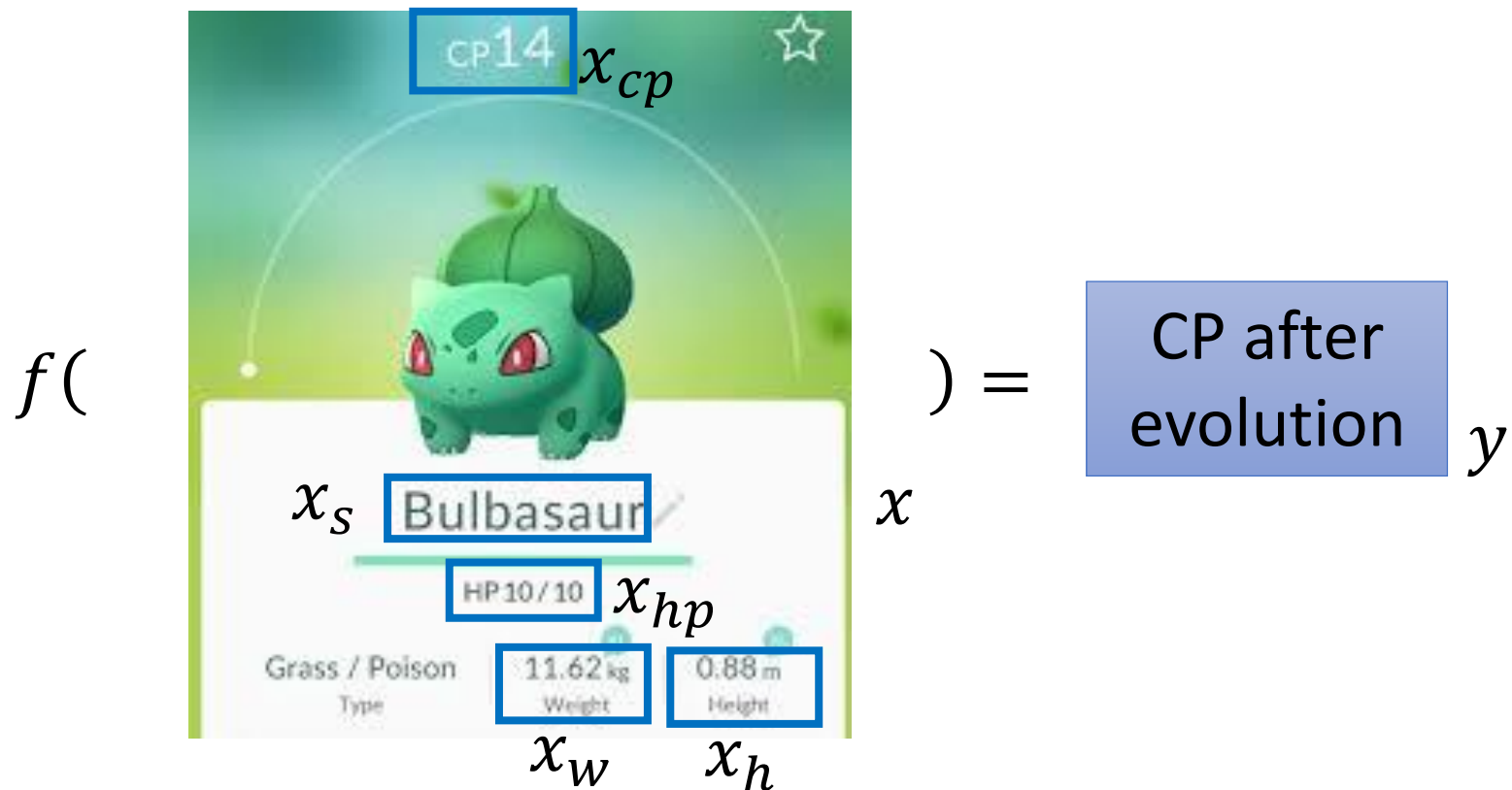
$$f(\text{Image of a self-driving car on a road}) = \text{方向盤角度}$$

- Recommendation

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

Example Application

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



Step 1: Model

一個model中，不同的參數組合成一個function set

w and b are parameters
(can be any value)

$$f_1: y = 10.0 + 9.0 \cdot x_{cp}$$

$$f_2: y = 9.8 + 9.2 \cdot x_{cp}$$

$$f_3: y = -0.8 - 1.2 \cdot x_{cp}$$

..... infinite

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

A set of
function

Model

$f_1, f_2 \dots$

$f($



$x) =$

CP after
evolution

y

Linear model:

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

$x_i: x_{cp}, x_{hp}, x_w, x_h \dots$

feature

w_i : weight, b: bias

Step 2: Goodness of Function

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



^:表示實際觀察到的結果

function
input:

function
Output (scalar):



Step 2: Goodness of Function

Training Data:
10 pokemons

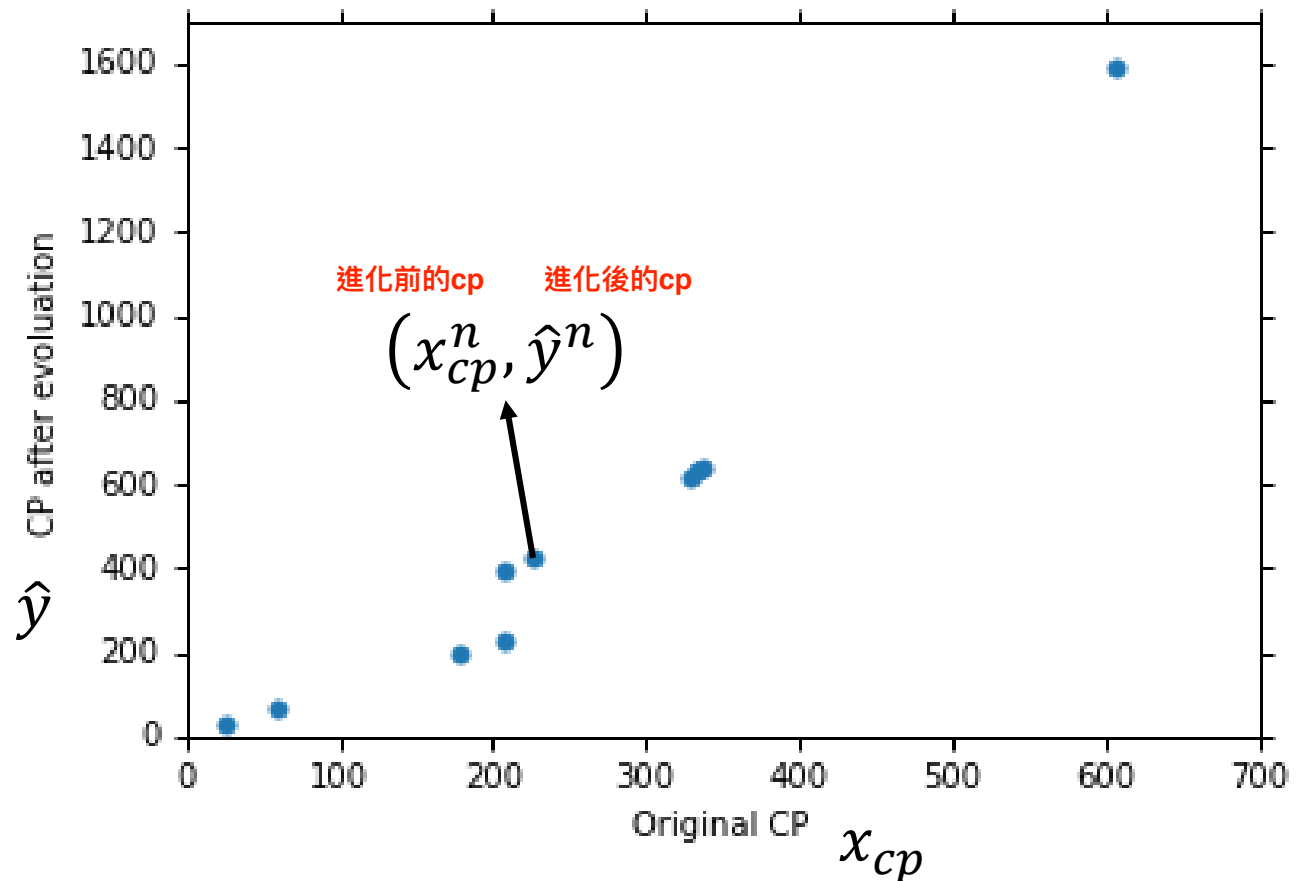
$$(x^1, \hat{y}^1)$$

$$(x^2, \hat{y}^2)$$

⋮

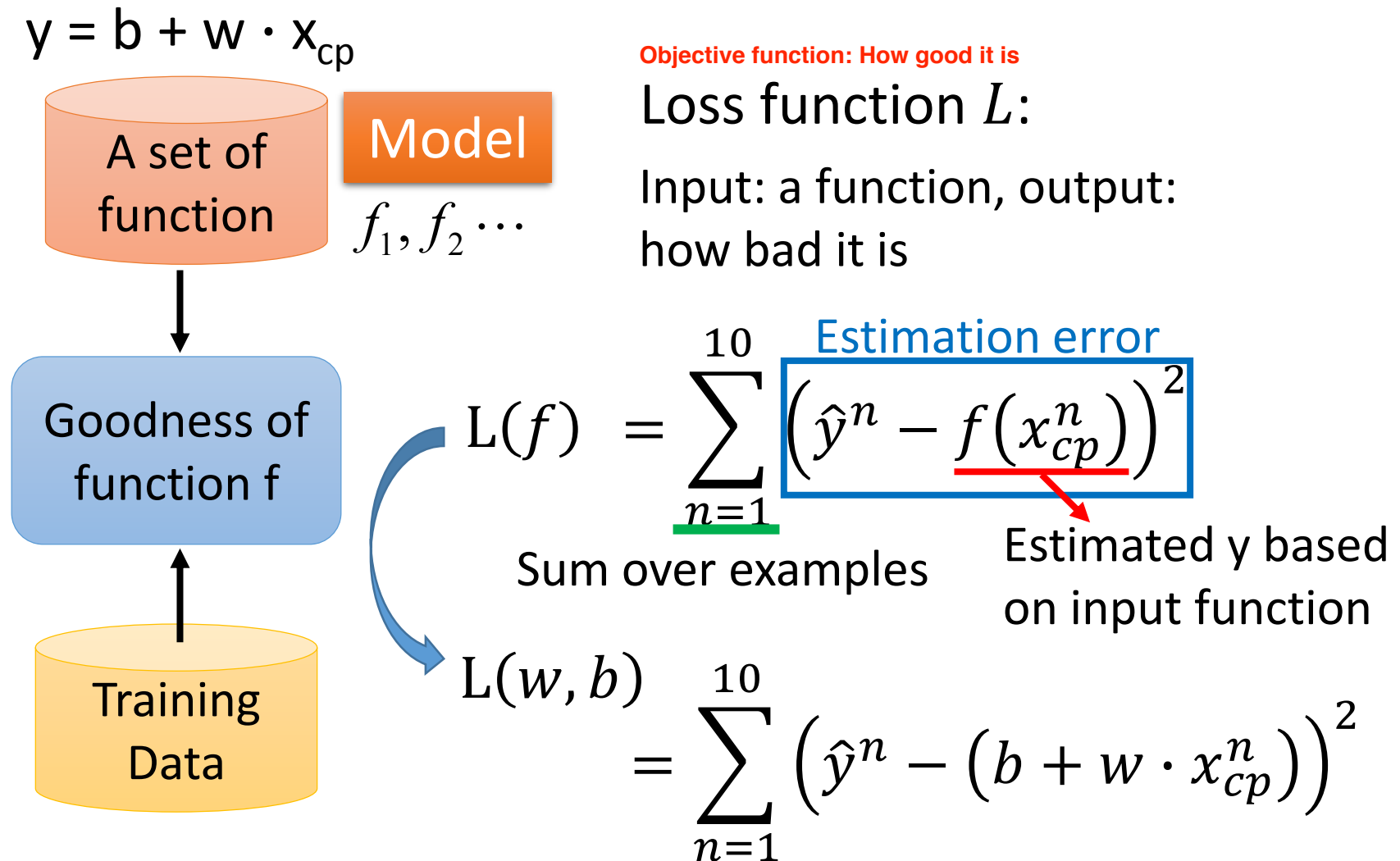
$$(x^{10}, \hat{y}^{10})$$

This is real data.



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

Step 2: Goodness of Function



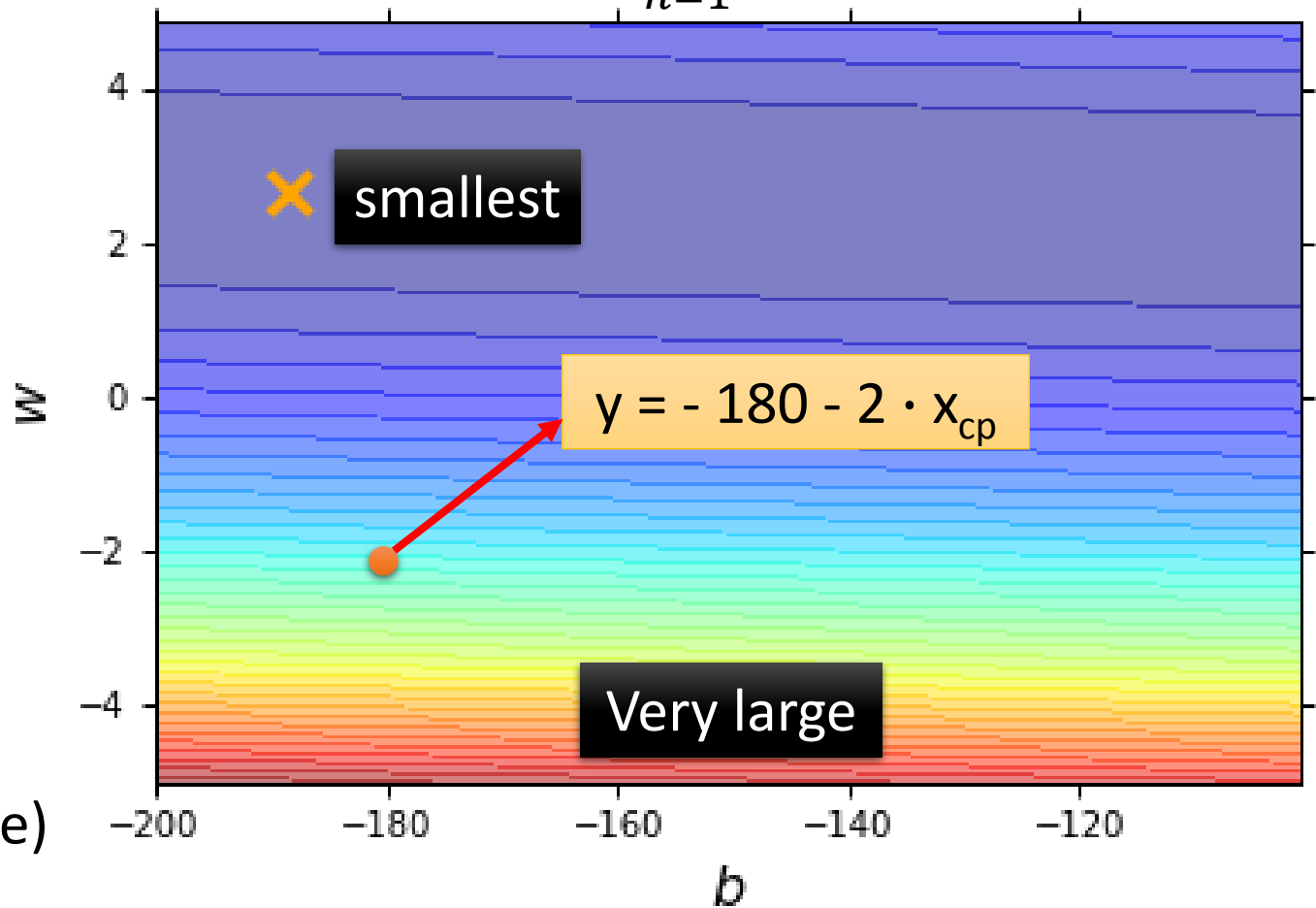
Step 2: Goodness of Function

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

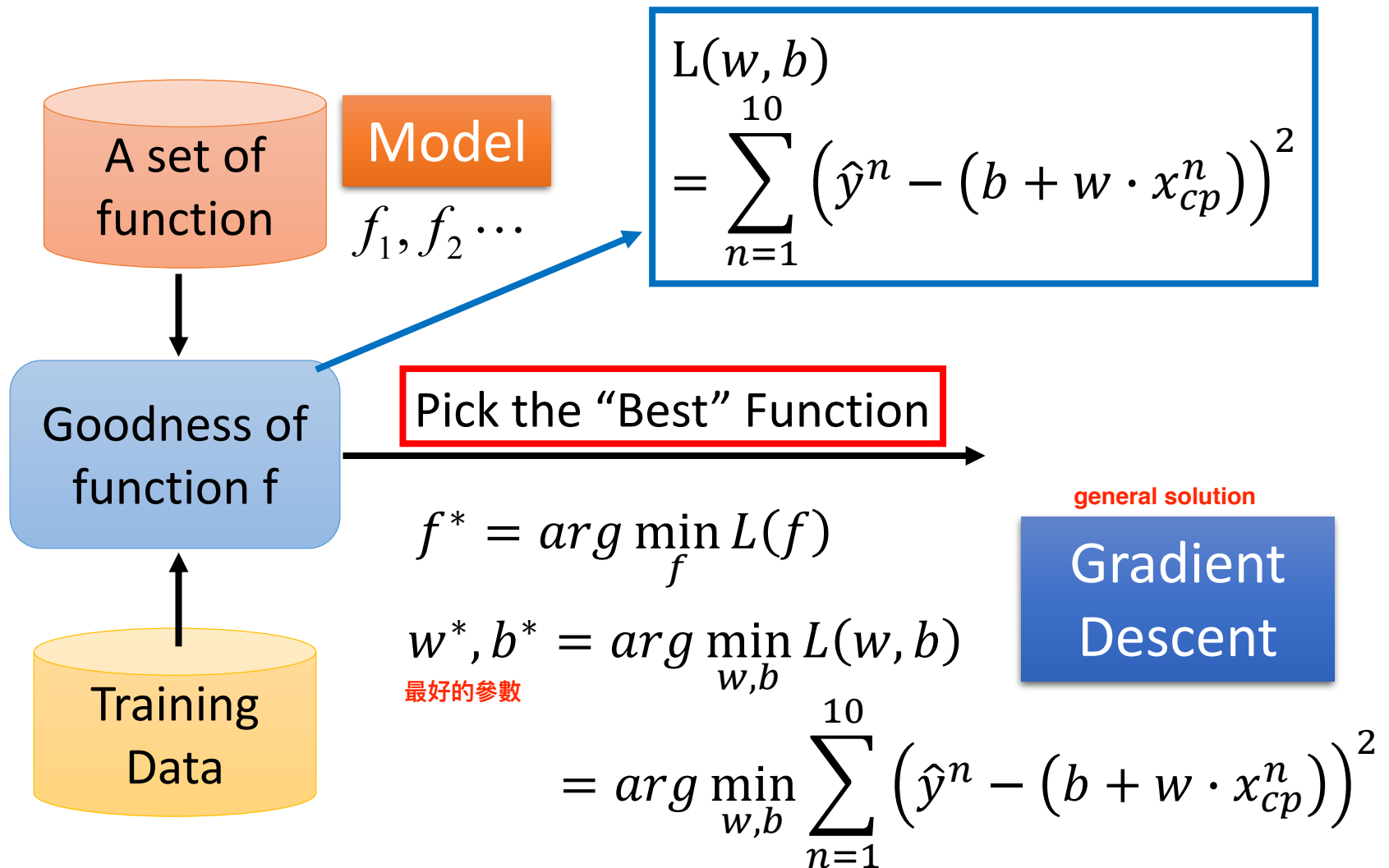
- Loss Function

Each point in the figure is a function

The color represents $L(w, b)$.



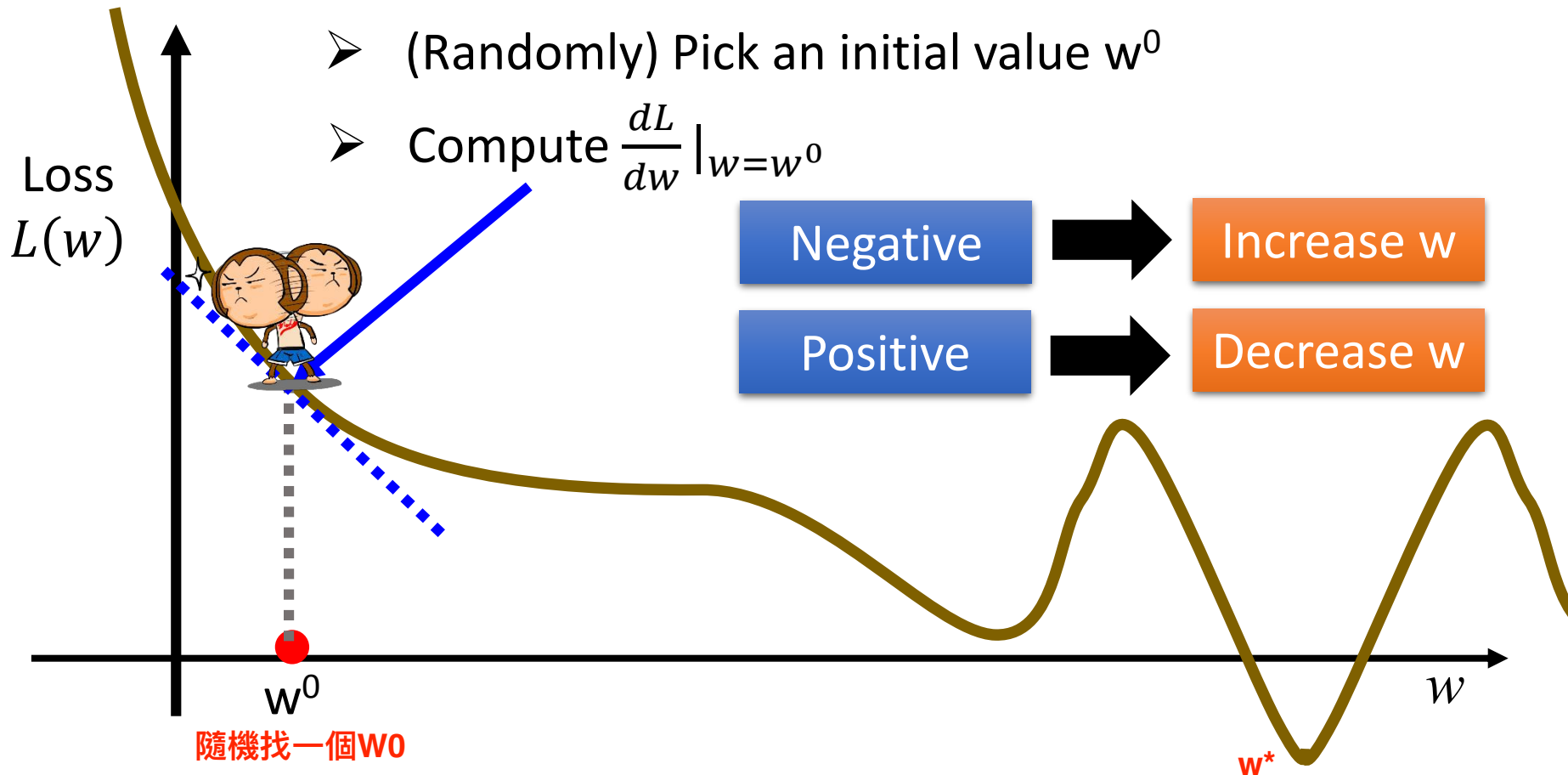
Step 3: Best Function



Step 3: Gradient Descent

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

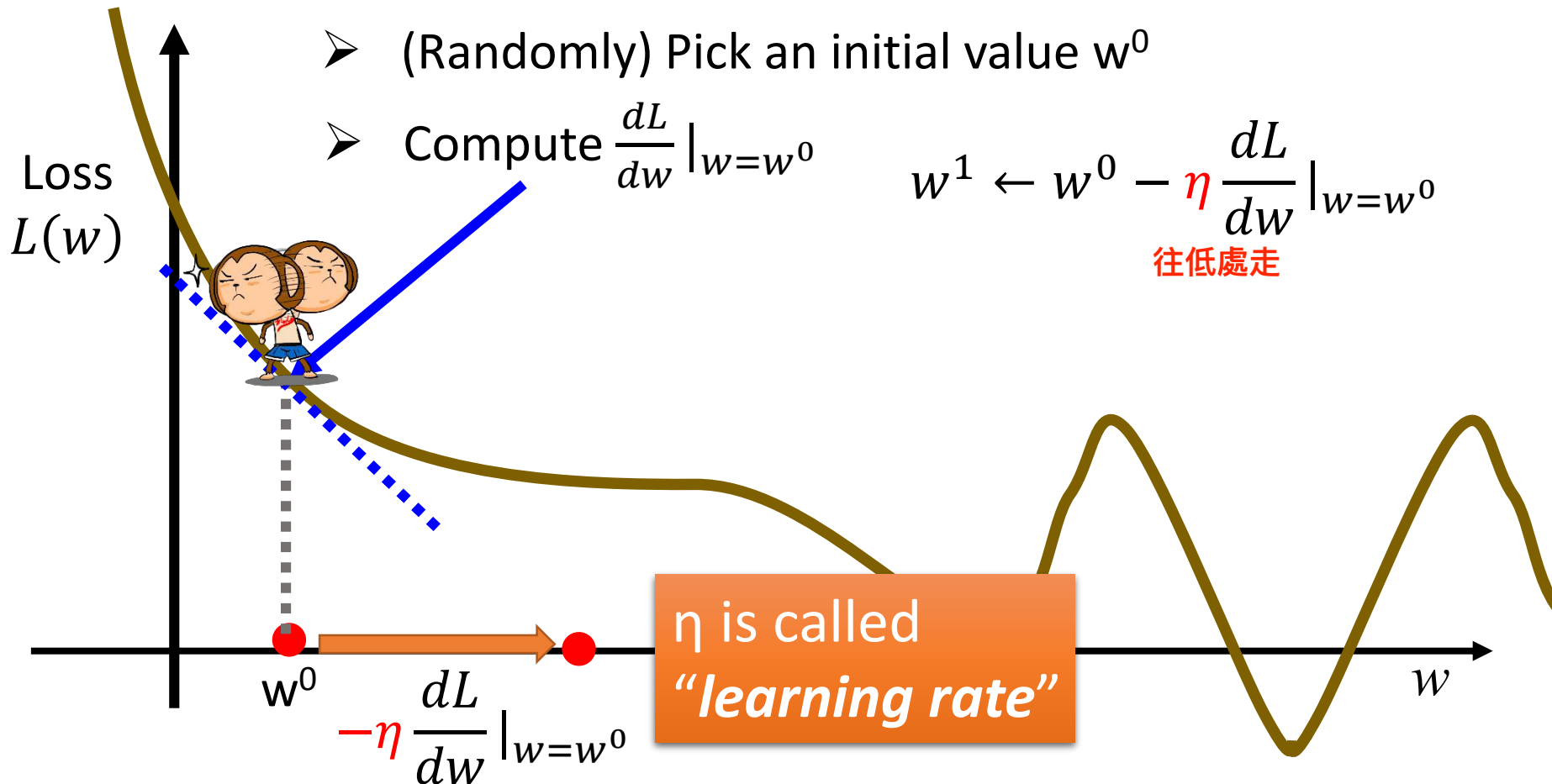
- Consider loss function $L(w)$ with one parameter w :



Step 3: Gradient Descent

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

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



Step 3: Gradient Descent

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

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

➤ (Randomly) Pick an initial value w^0

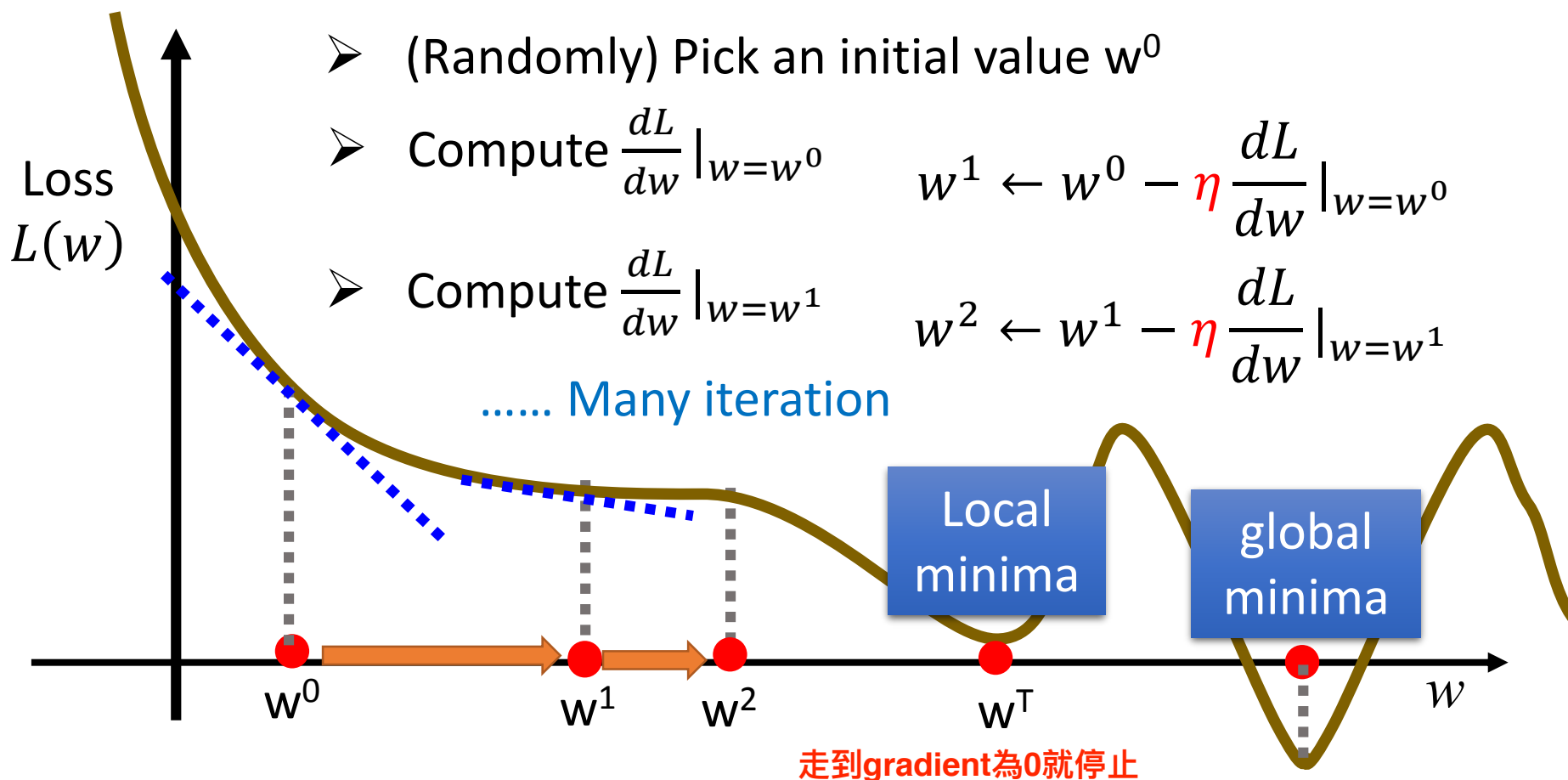
➤ Compute $\frac{dL}{dw} \big|_{w=w^0}$

$$w^1 \leftarrow w^0 - \eta \frac{dL}{dw} \big|_{w=w^0}$$

➤ Compute $\frac{dL}{dw} \big|_{w=w^1}$

$$w^2 \leftarrow w^1 - \eta \frac{dL}{dw} \big|_{w=w^1}$$

..... Many iteration



排成vector，叫做gradient，
用倒三角形L表示

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

Step 3: Gradient Descent

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

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

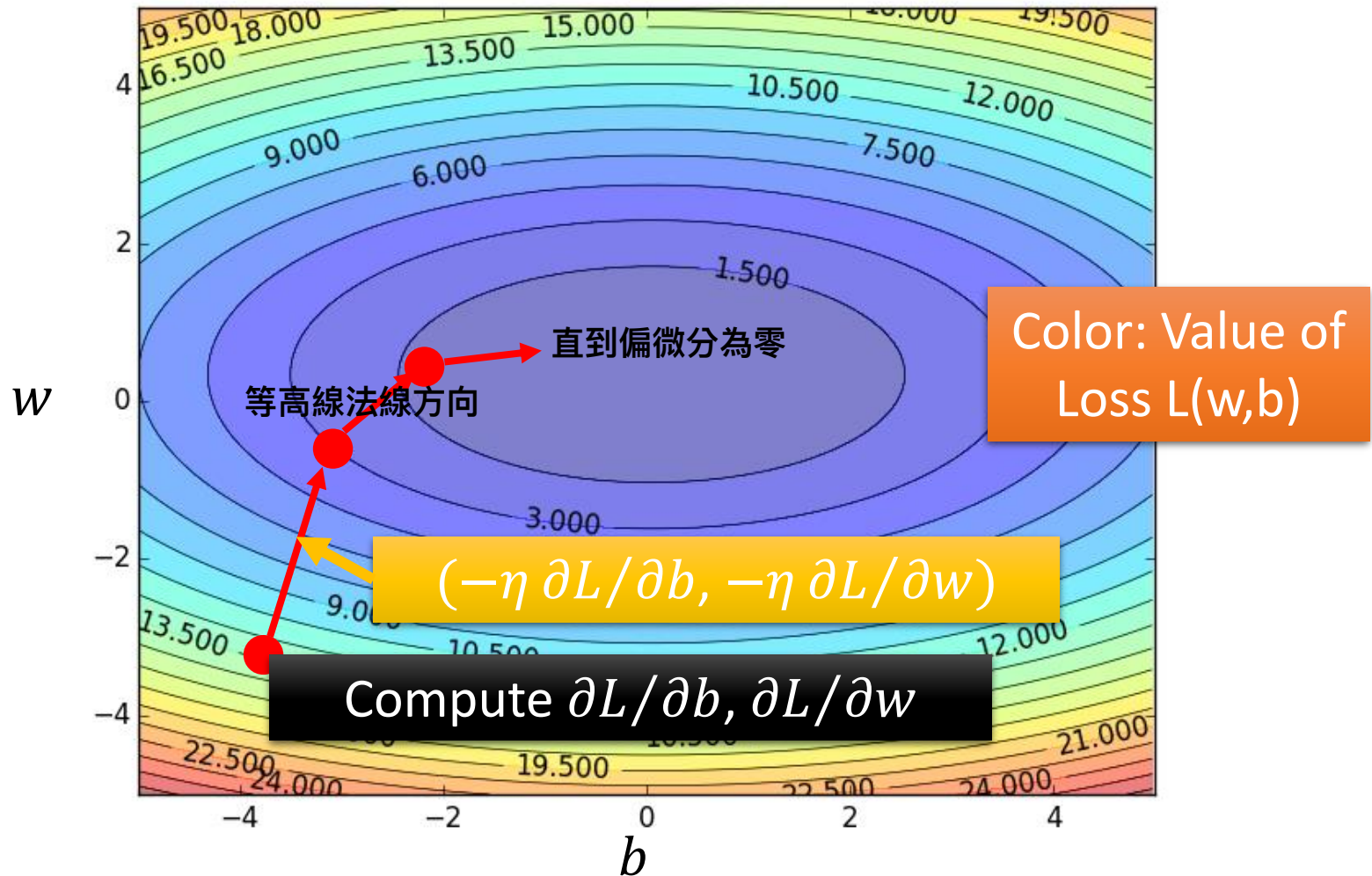
➤ Compute $\frac{\partial L}{\partial w} \Big|_{w=w^0, b=b^0}, \frac{\partial L}{\partial b} \Big|_{w=w^0, b=b^0}$ 兩個參數做偏微分

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

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

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

Step 3: Gradient Descent



Step 3: Gradient Descent

- When solving:

θ : 一個參數集合

$$\theta^* = \arg \max_{\theta} L(\theta) \quad \text{by gradient descent}$$

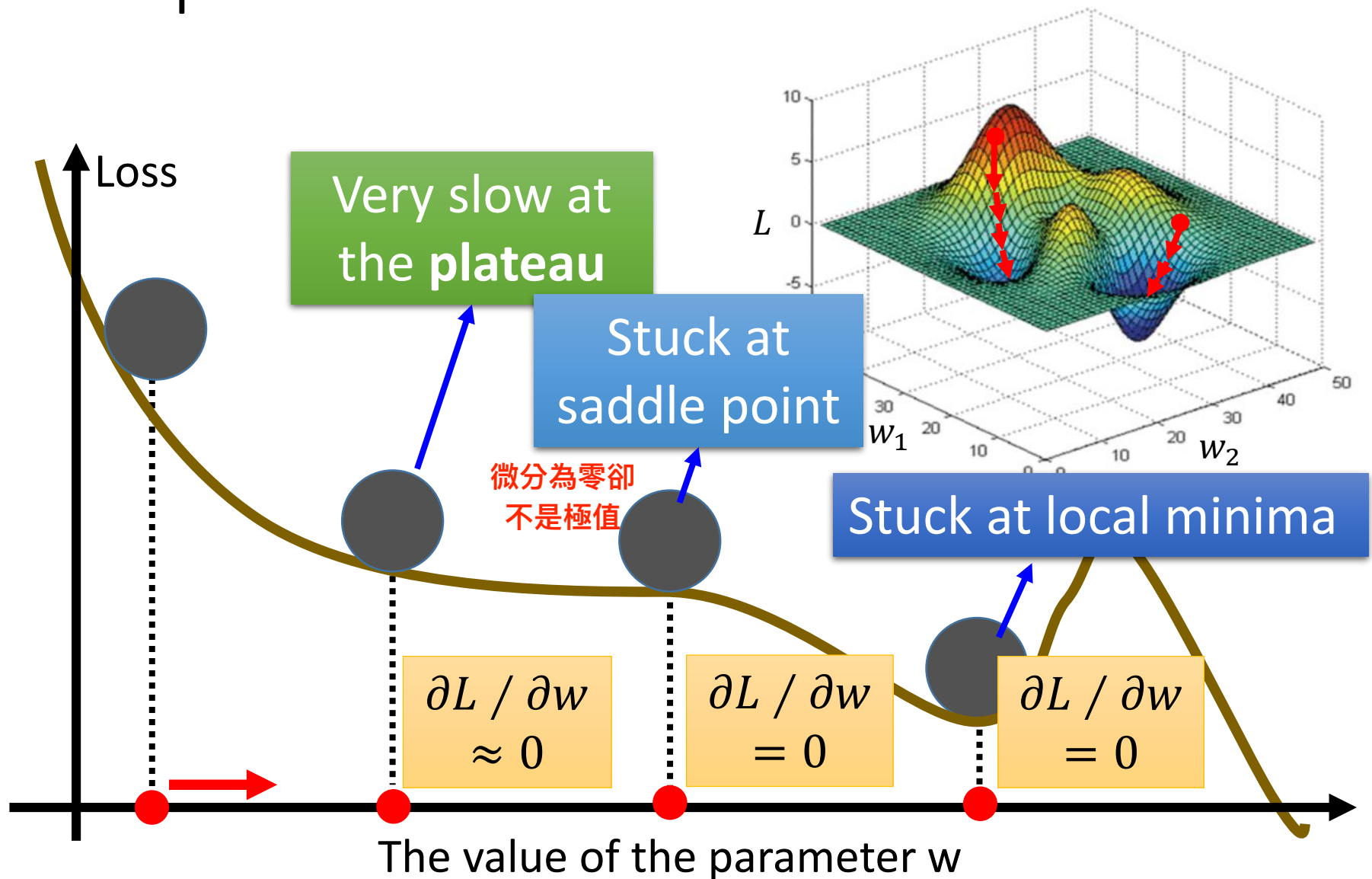
- Each time we update the parameters, we obtain θ that makes $L(\theta)$ smaller.

$$L(\theta^0) > L(\theta^1) > L(\theta^2) > \dots$$

Is this statement correct?

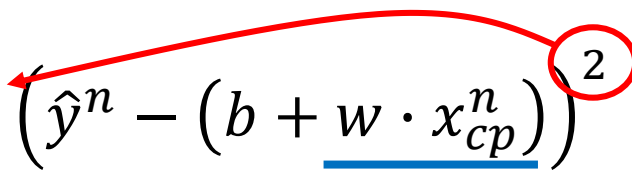
No!!

Step 3: Gradient Descent



Step 3: Gradient Descent

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


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


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

$$\frac{\partial L}{\partial b} = ?$$

Step 3: Gradient Descent

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

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


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

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

Step 3: Gradient Descent

How's the results?

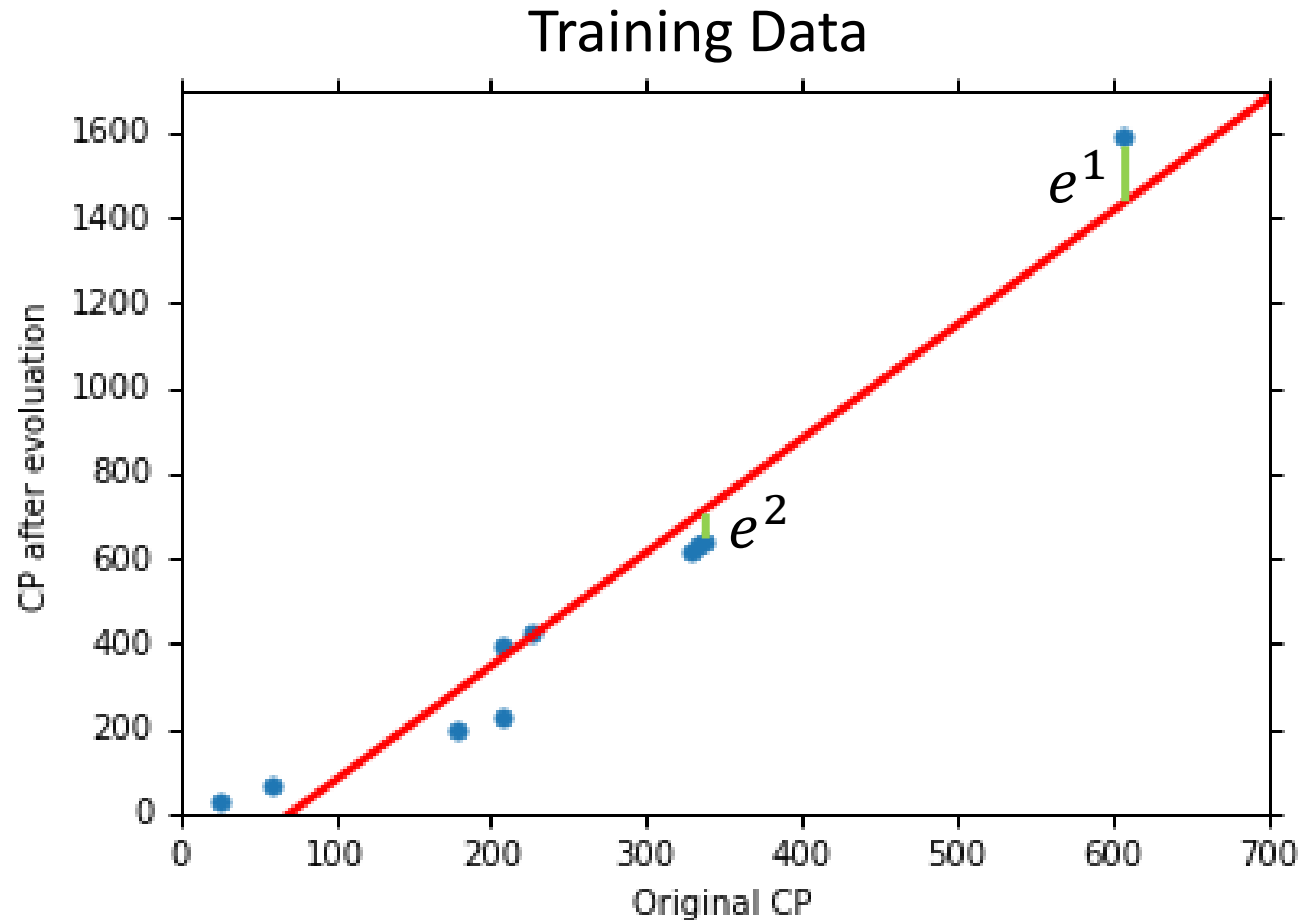
$$y = b + w \cdot x_{cp}$$

$$b = -188.4$$

$$w = 2.7$$

Average Error on
Training Data

$$= \sum_{n=1}^{10} e^n = 31.9$$



How's the results?

- Generalization

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

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

$$b = -188.4$$

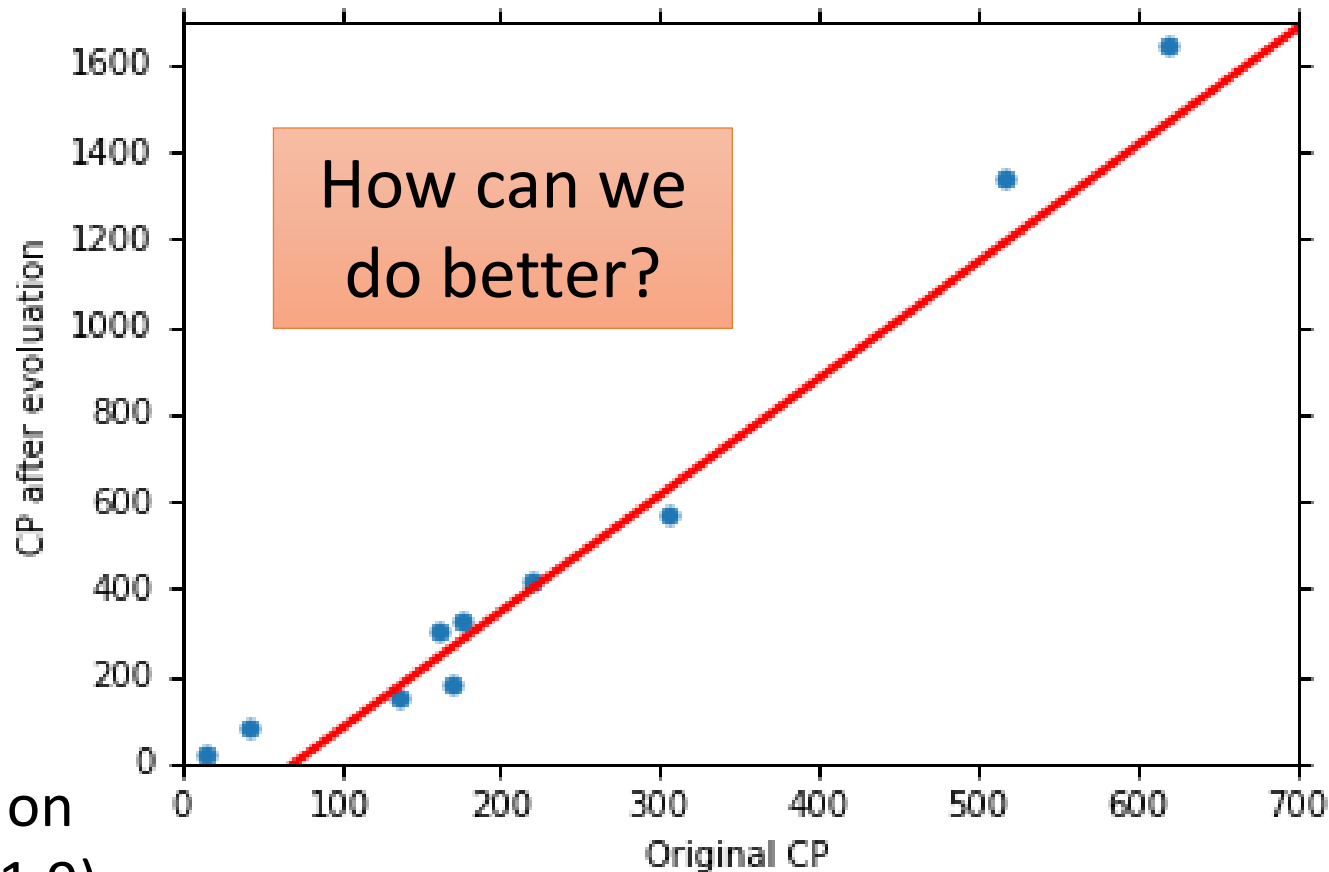
$$w = 2.7$$

Average Error on
Testing Data

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

> Average Error on
Training Data (31.9)

Another 10 pokemons as testing data



Selecting another Model

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

linear model

Best Function

$$b = -10.3$$

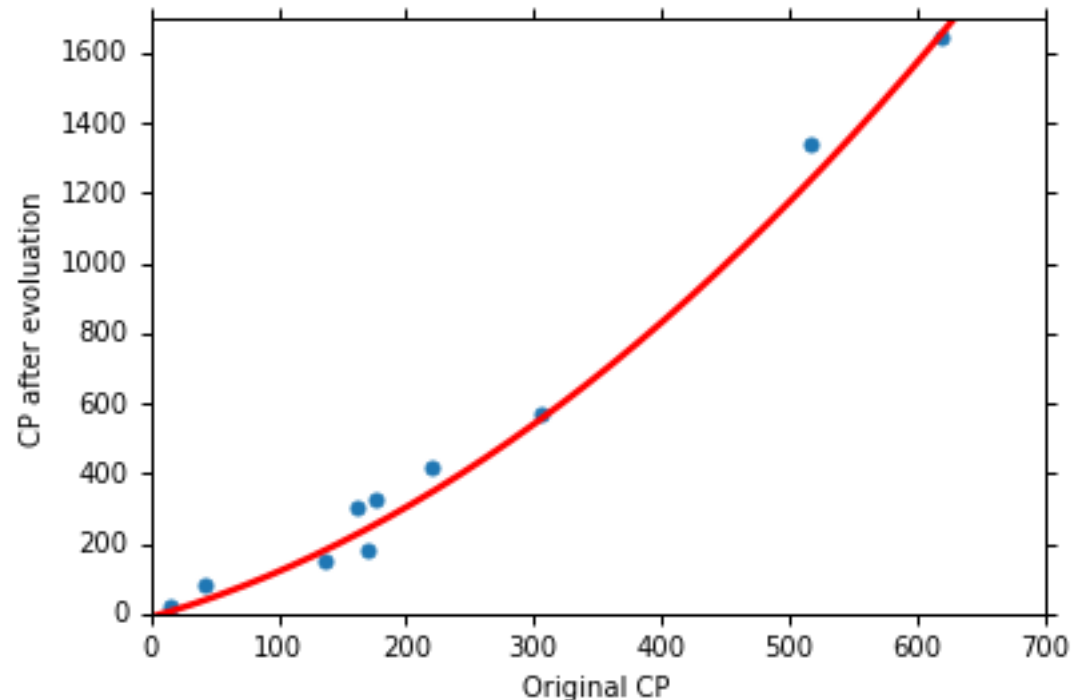
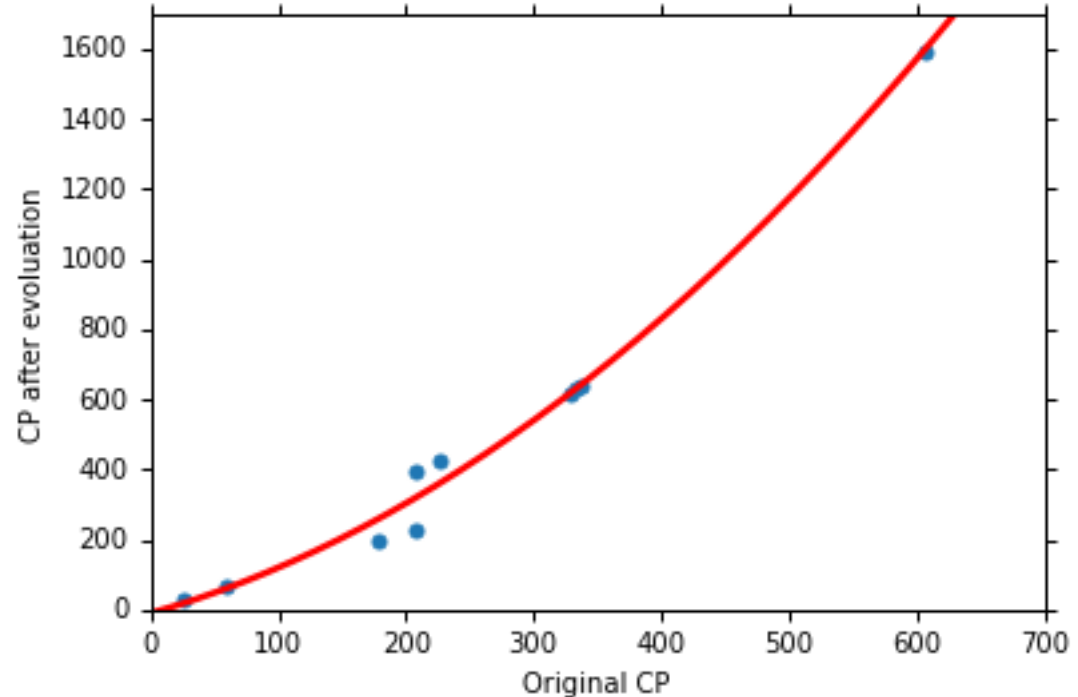
$$w_1 = 1.0, w_2 = 2.7 \times 10^{-3}$$

Average Error = 15.4

Testing:

Average Error = 18.4

Better! Could it be even better?



Selecting another Model

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

Best Function

$$b = 6.4, w_1 = 0.66$$

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

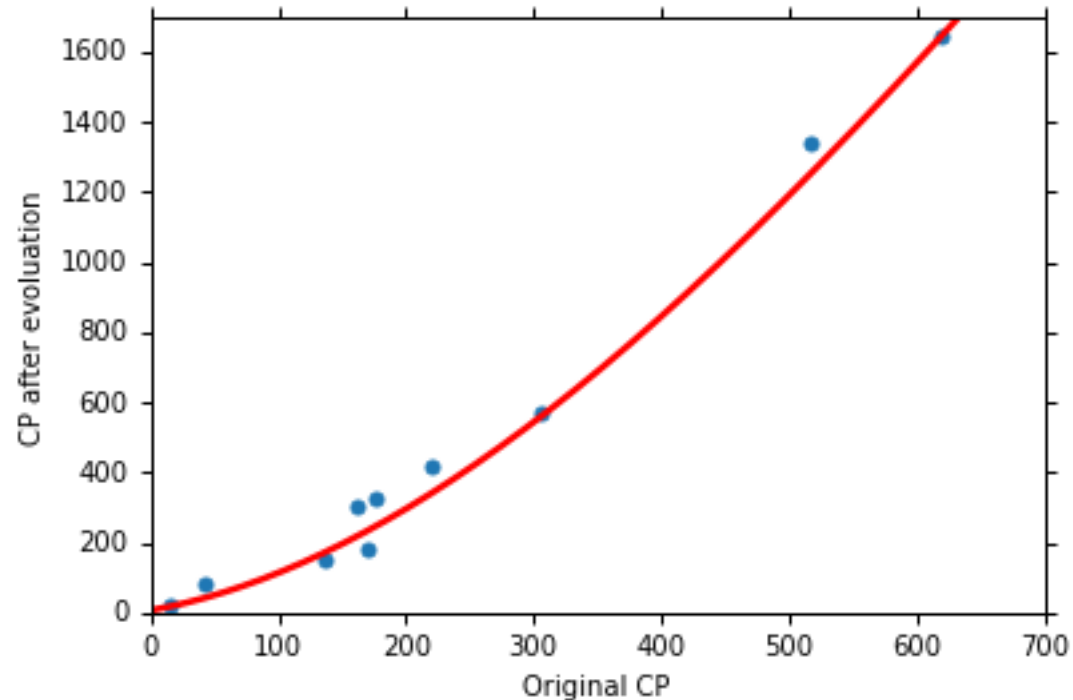
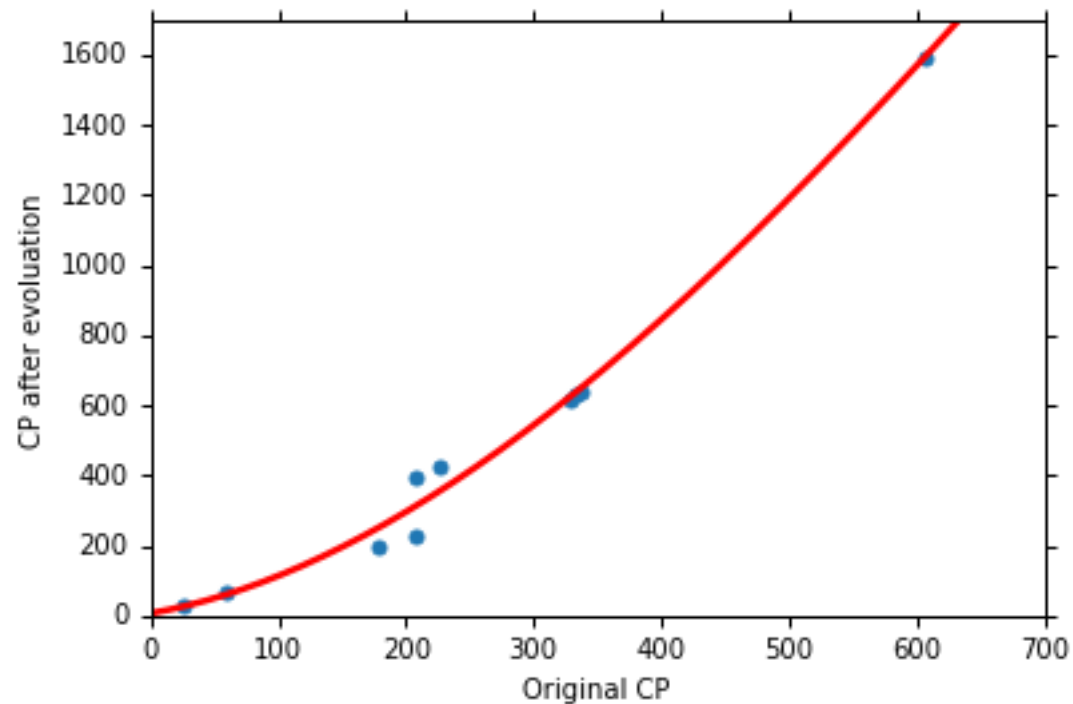
$$w_3 = -1.8 \times 10^{-6}$$

Average Error = 15.3

Testing:

Average Error = 18.1

Slightly better.
How about more
complex model?



Selecting another Model

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

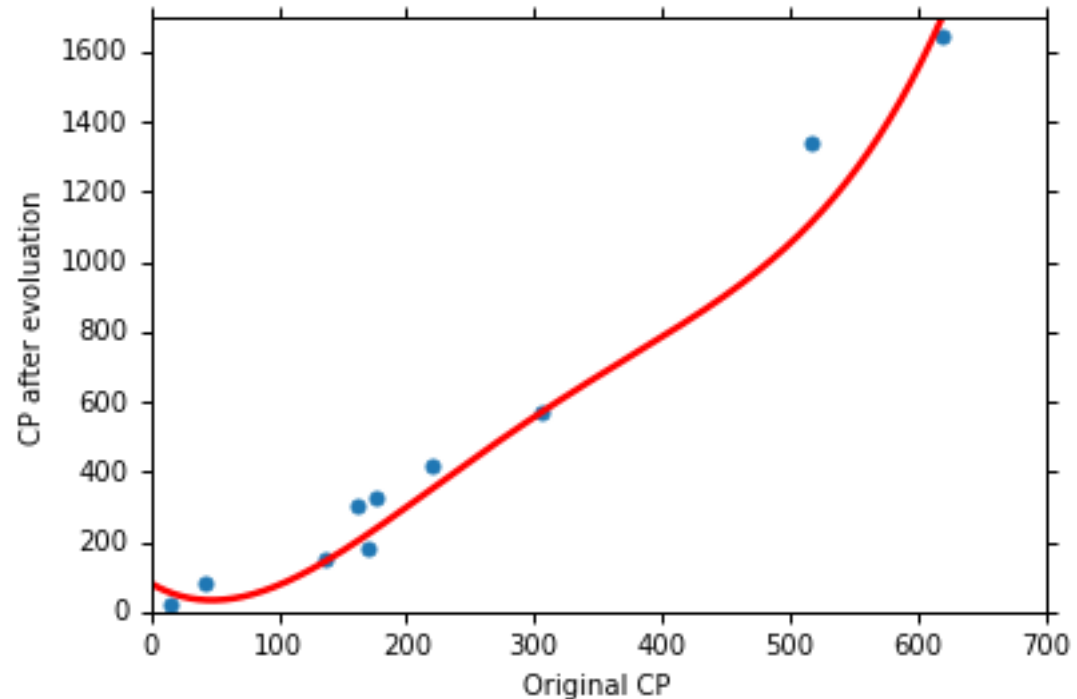
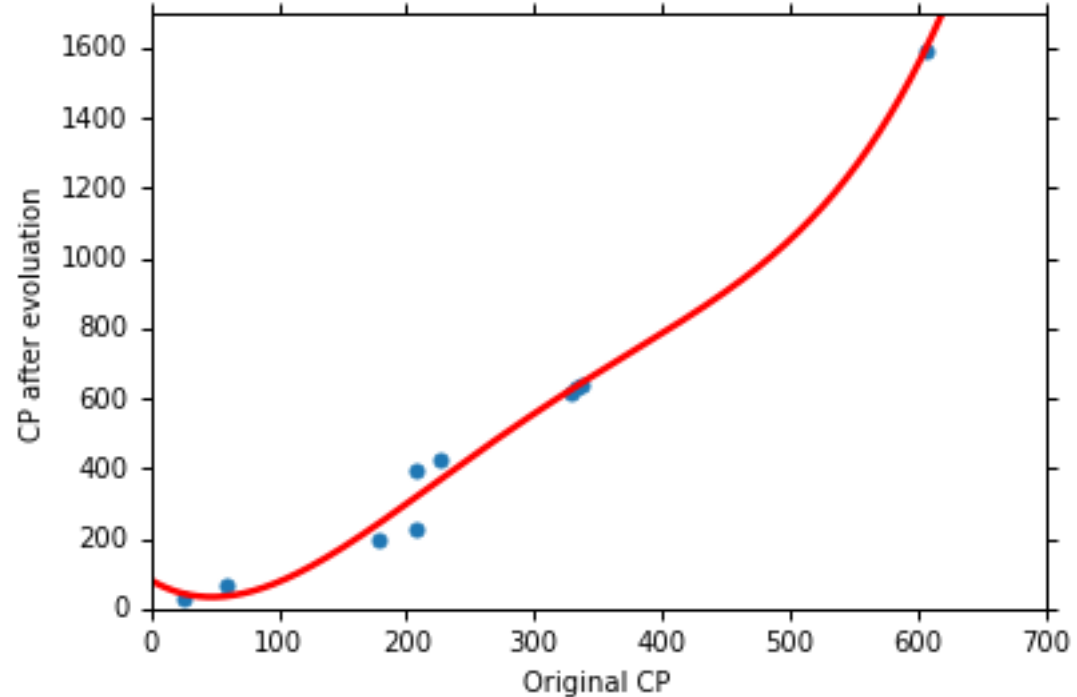
Best Function

Average Error = 14.9

Testing:

Average Error = 28.8

The results become worse ...



Selecting another Model

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

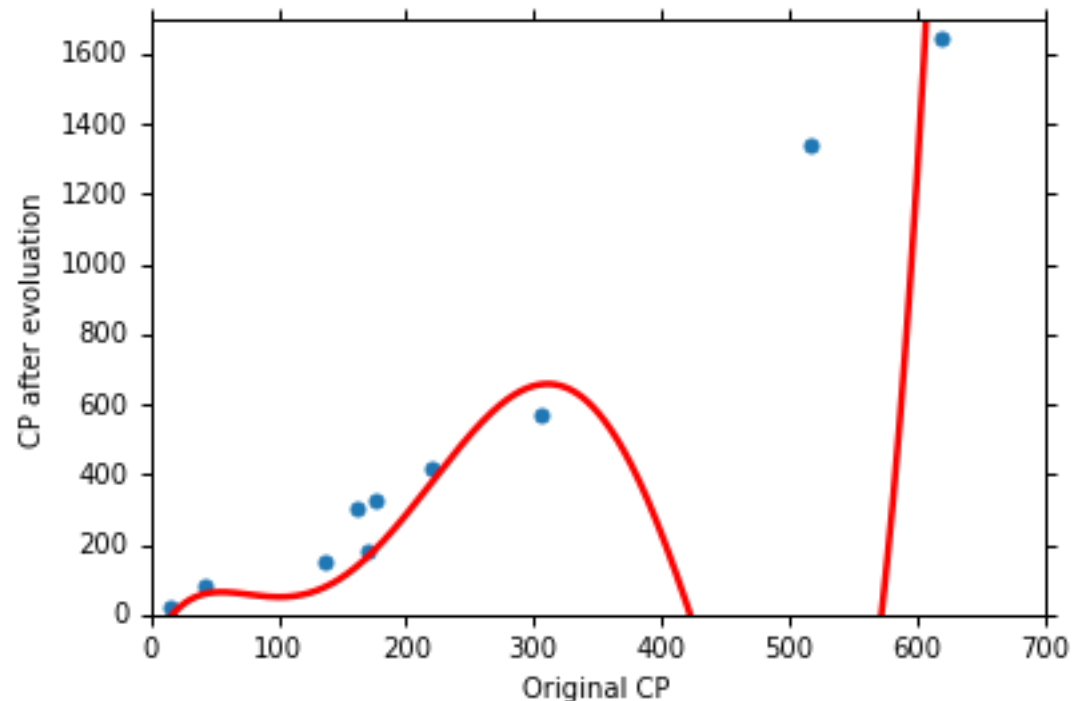
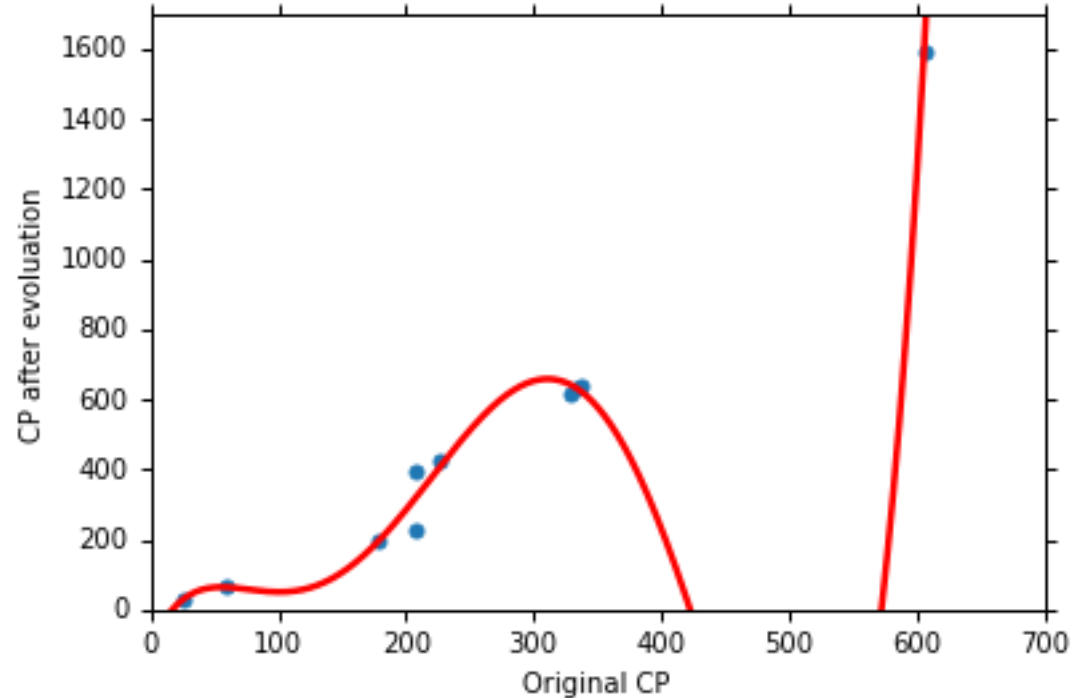
Best Function

Average Error = 12.8

Testing:

Average Error = 232.1

The results are so bad.



Model Selection

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

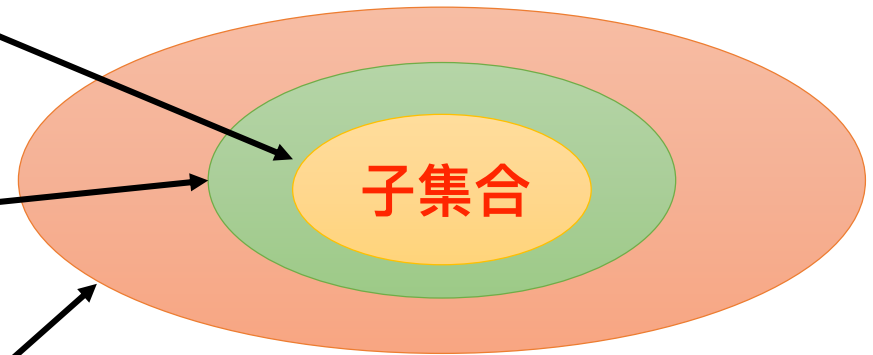
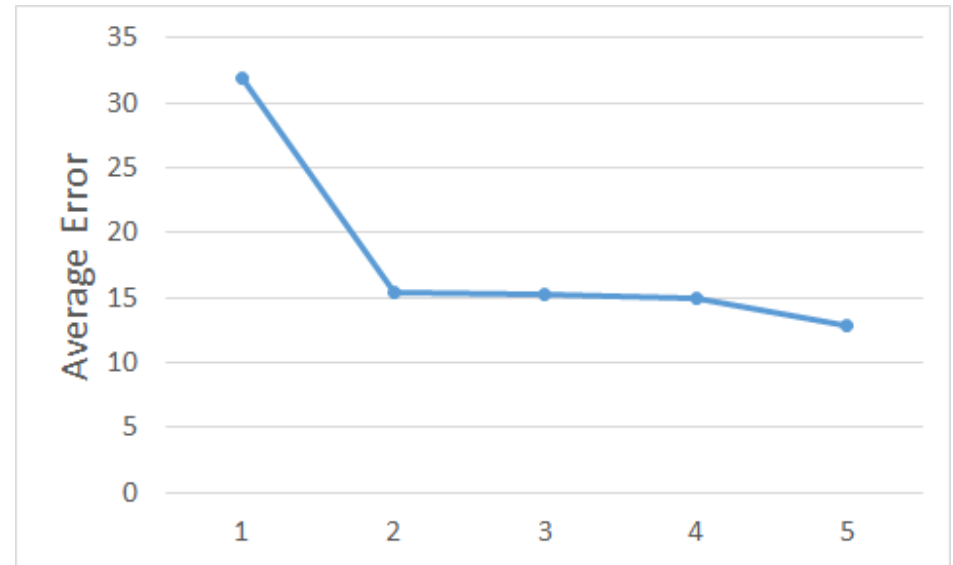
2. $y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2$

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

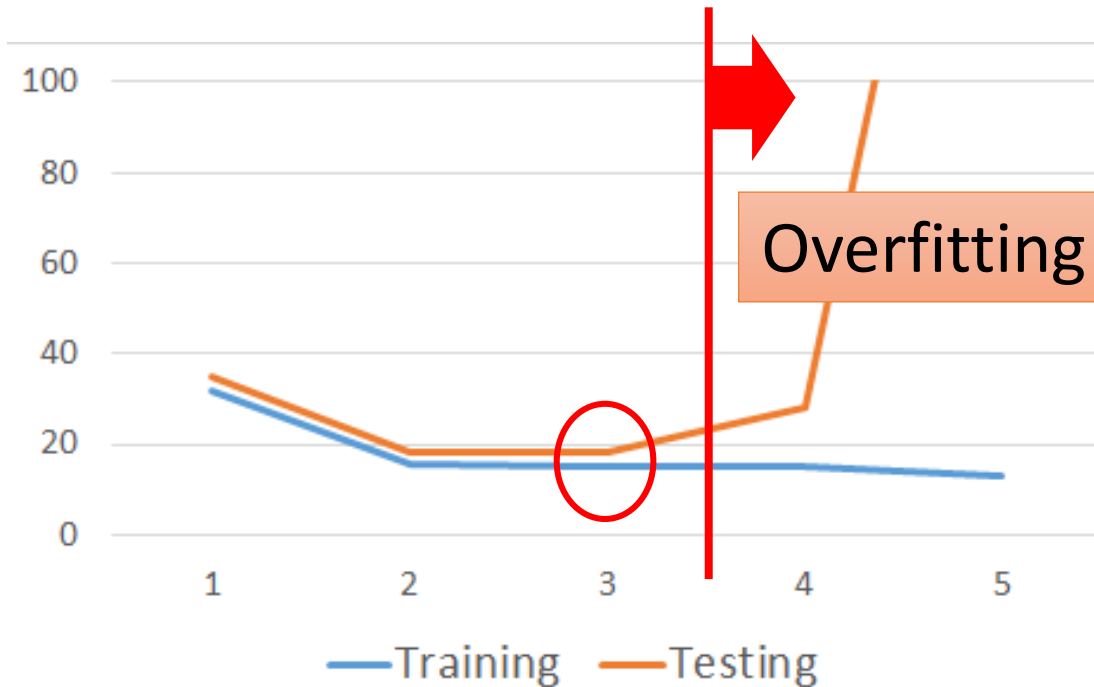
Training Data



A more complex model yields lower error on training data.

If we can truly find the best function

Model Selection

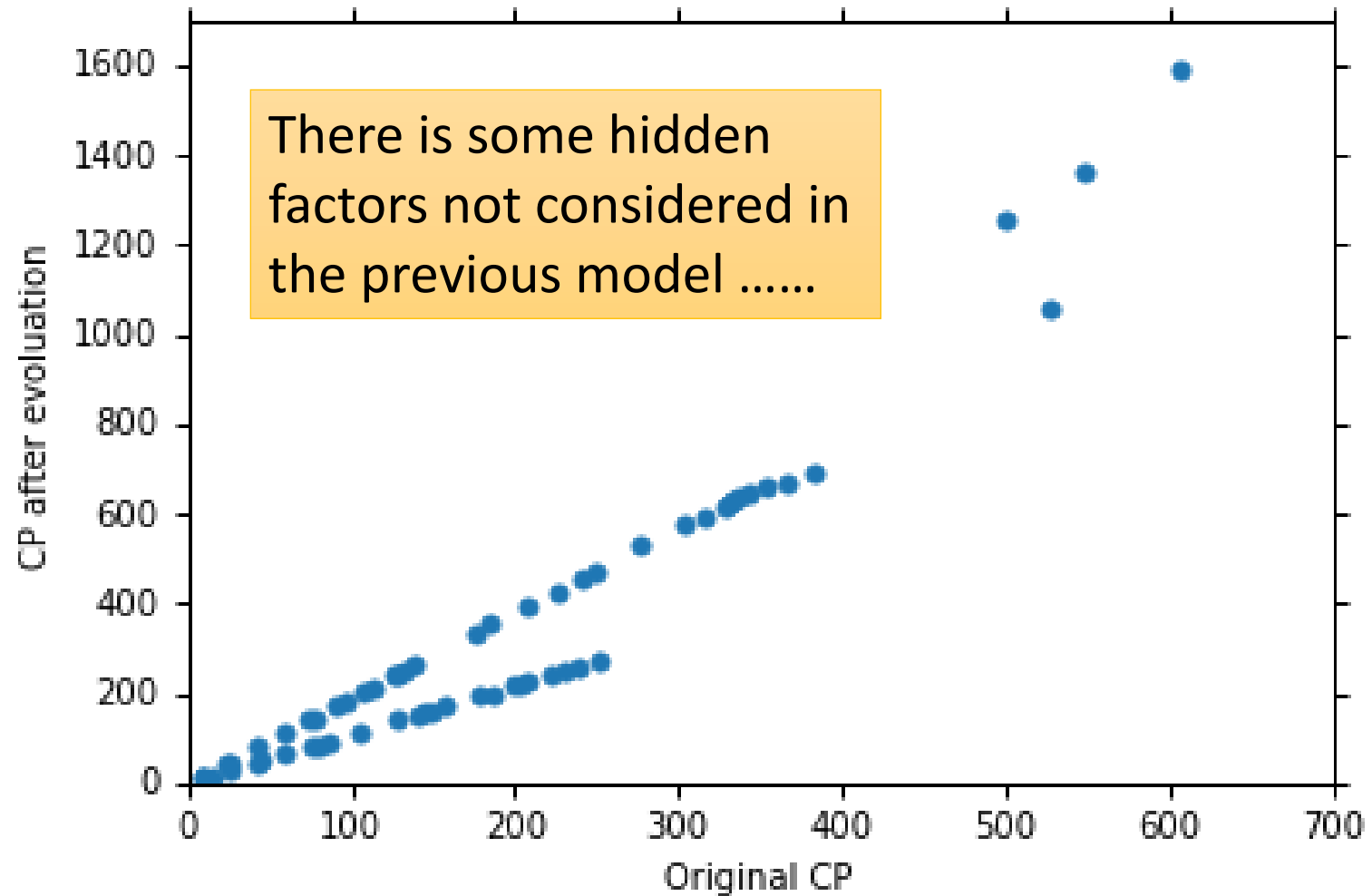


| | Training | Testing |
|---|----------|---------|
| 1 | 31.9 | 35.0 |
| 2 | 15.4 | 18.4 |
| 3 | 15.3 | 18.1 |
| 4 | 14.9 | 28.2 |
| 5 | 12.8 | 232.1 |

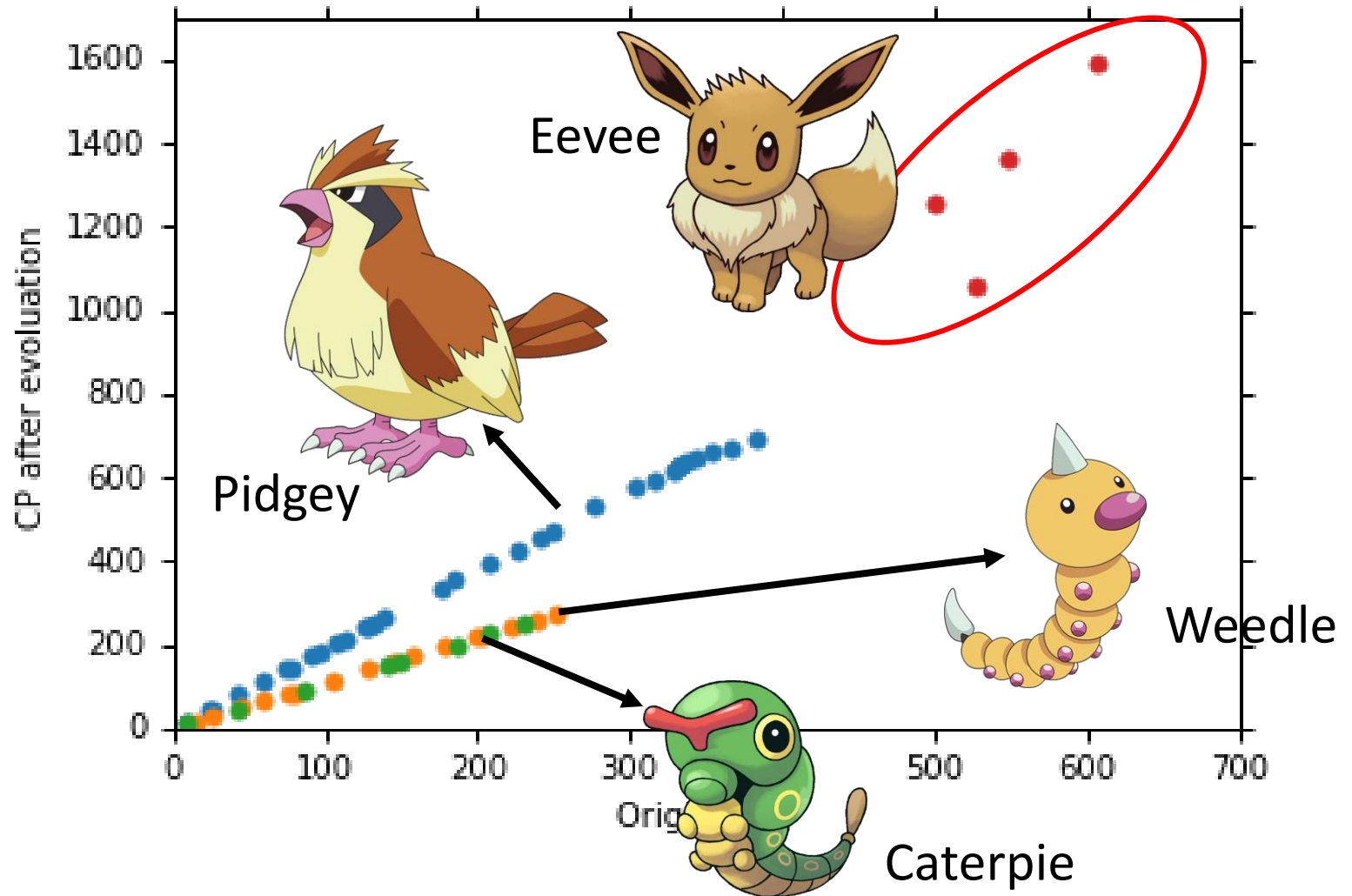
A more complex model does not always lead to better performance on testing data.

This is Overfitting.  Select suitable model

Let's collect more data



What are the hidden factors?



Back to step 1: Redesign the Model

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

Linear model?

yes

x_s = species of x

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

Back to step 1: Redesign the Model

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

Linear model?

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

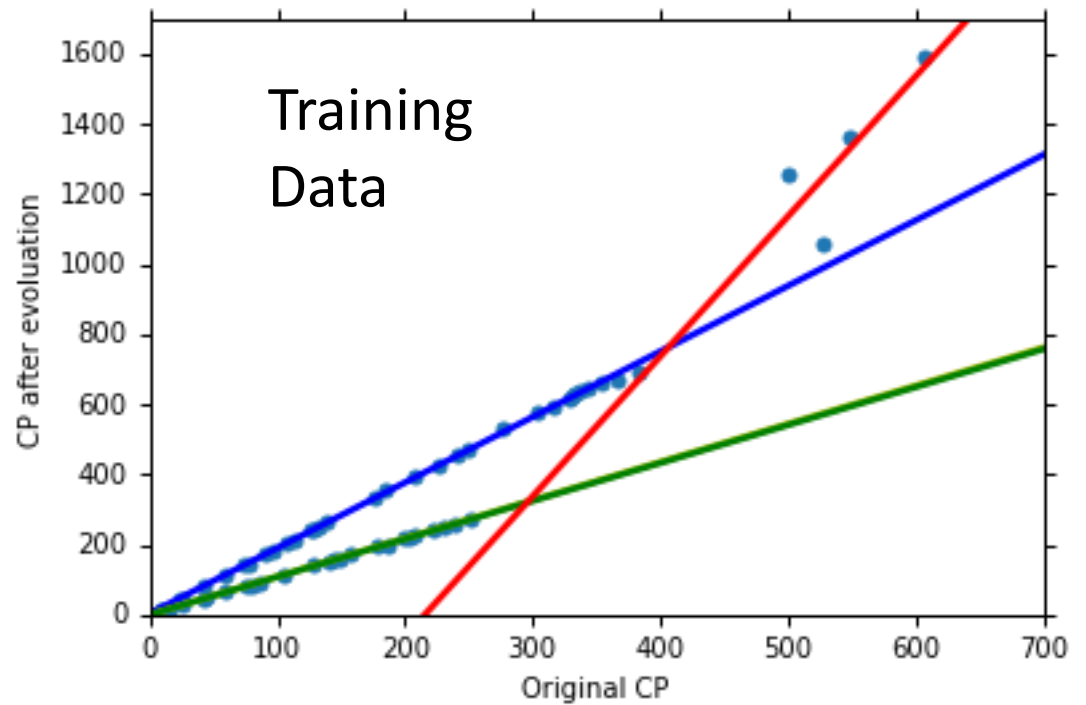
$$\delta(x_s = \text{Pidgely})$$

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

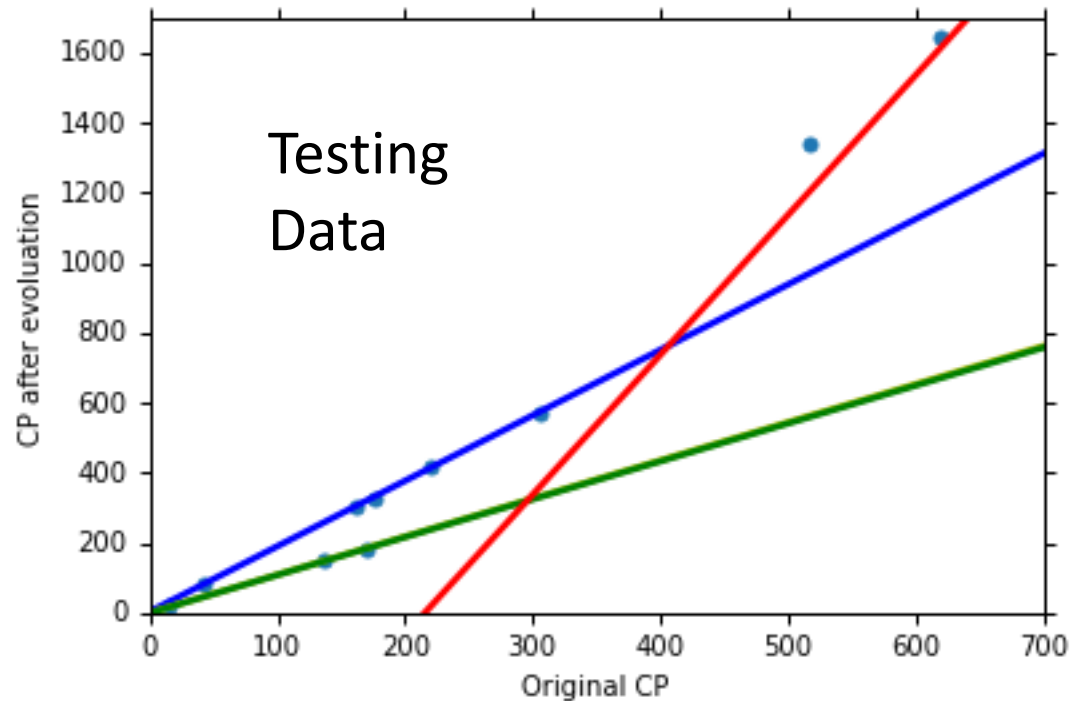
If $x_s = \text{Pidgely}$

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

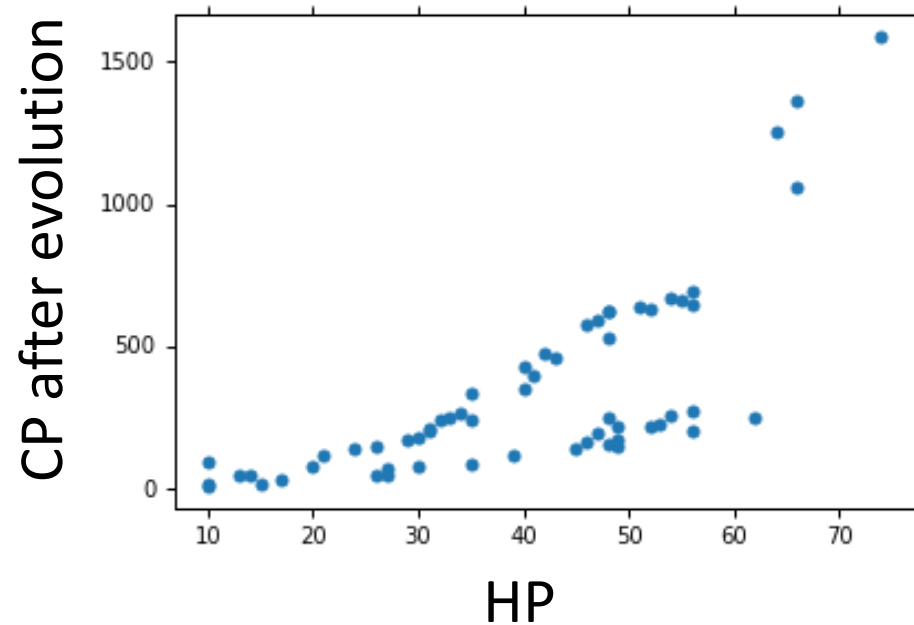
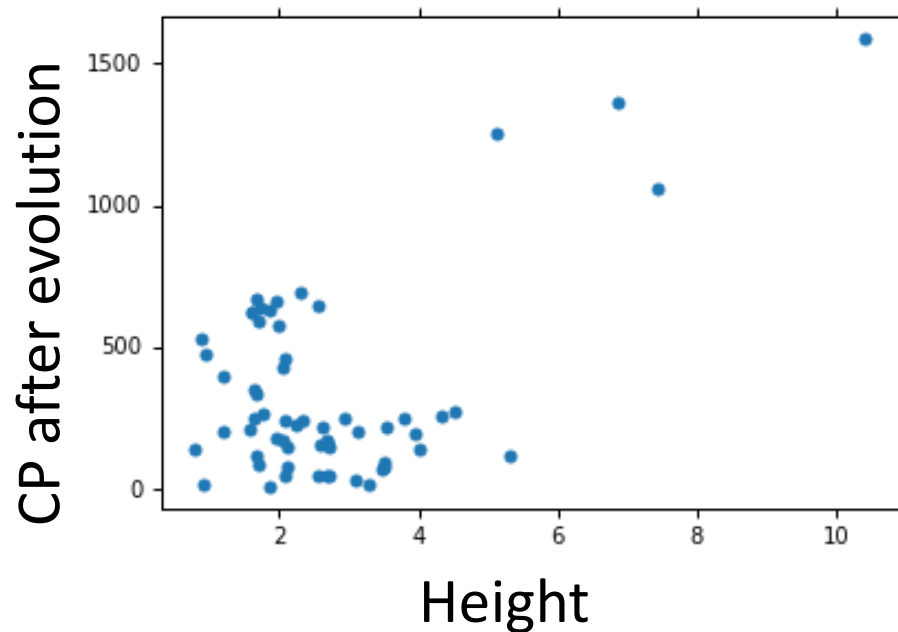
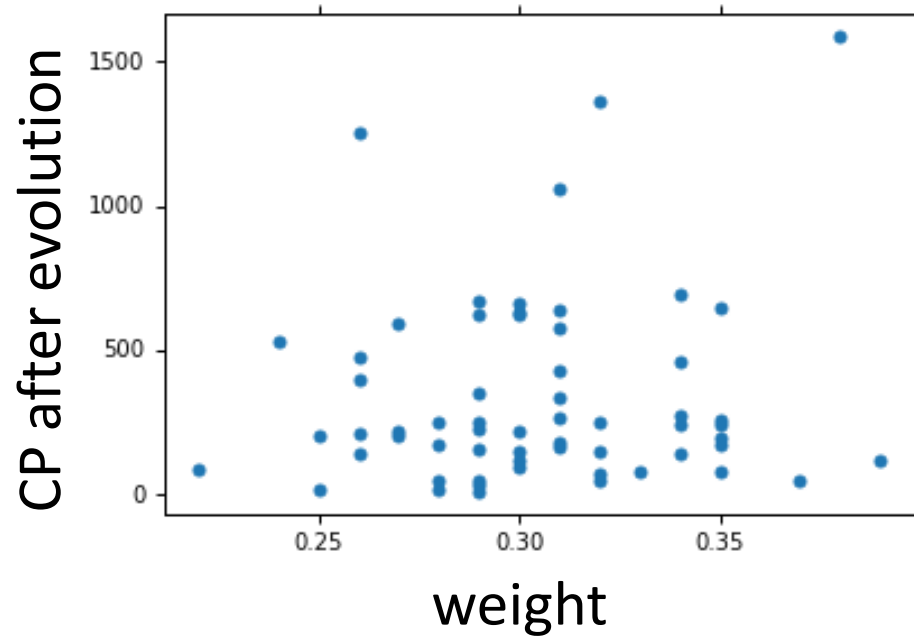
Average error
= 3.8



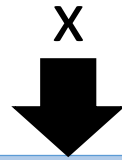
Average error
= 14.3



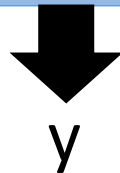
Are there any other hidden factors?



Back to step 1: Redesign the Model Again



$$\begin{aligned} \text{If } x_s = \text{Pidgey:} \quad & y' = b_1 + w_1 \cdot x_{cp} + w_5 \cdot (x_{cp})^2 \\ \text{If } x_s = \text{Weedle:} \quad & y' = b_2 + w_2 \cdot x_{cp} + w_6 \cdot (x_{cp})^2 \\ \text{If } x_s = \text{Caterpie:} \quad & y' = b_3 + w_4 \cdot x_{cp} + w_7 \cdot (x_{cp})^2 \\ \text{If } x_s = \text{Eevee:} \quad & y' = b_4 + w_4 \cdot x_{cp} + w_8 \cdot (x_{cp})^2 \\ & y = y' + w_9 \cdot x_{hp} + w_{10} \cdot (x_{hp})^2 \\ & + w_{11} \cdot x_h + w_{12} \cdot (x_h)^2 + w_{13} \cdot x_w + w_{14} \cdot (x_w)^2 \end{aligned}$$



Training Error
= 1.9

Testing Error
= 102.3

Overfitting!

Back to step 2: Regularization

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

w很小意味著平滑的function

The functions with
smaller w_i are better

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

$$+ \lambda \sum (w_i)^2$$

regularization term

➤ Smaller w_i means ...

smoother

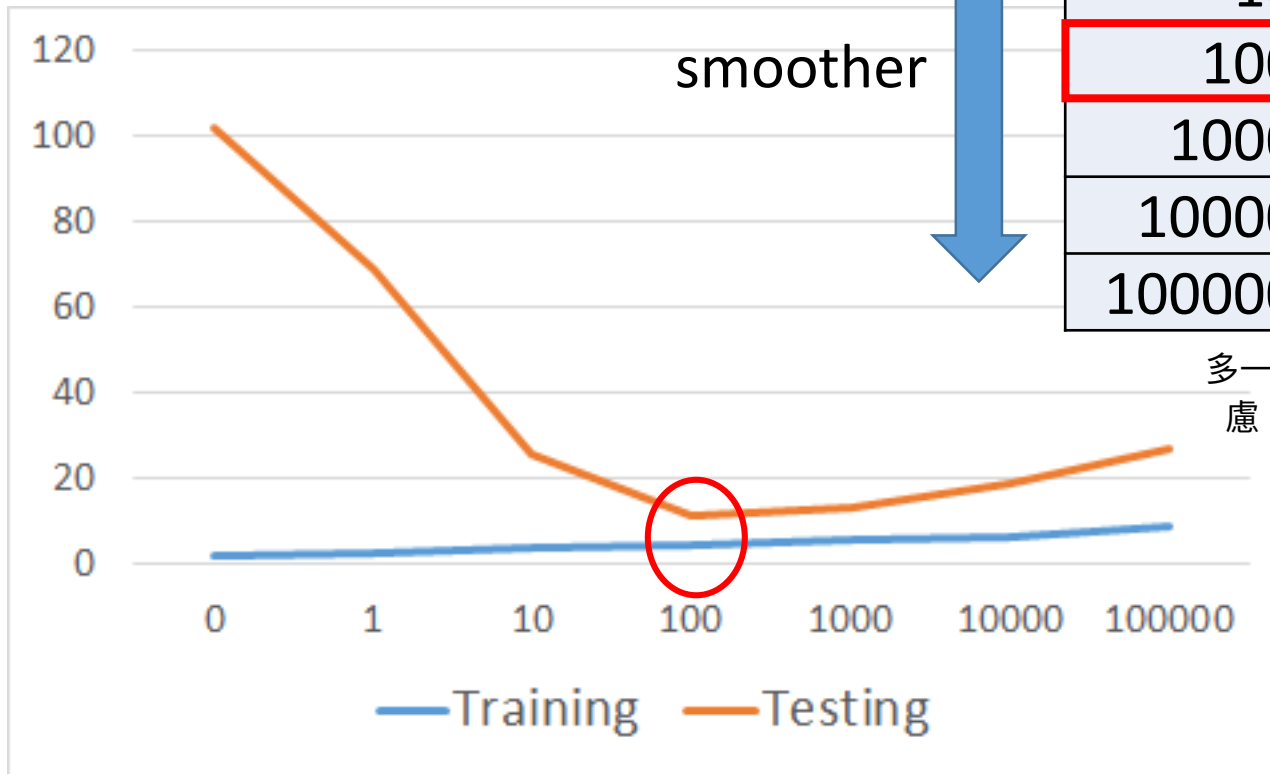
$$y = b + \sum w_i x_i$$

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

➤ We believe smoother function is more likely to be correct

Do you have to apply regularization on bias?

Regularization



| λ | Training | Testing |
|-----------|----------|---------|
| 0 | 1.9 | 102.3 |
| 1 | 2.3 | 68.7 |
| 10 | 3.5 | 25.7 |
| 100 | 4.1 | 11.1 |
| 1000 | 5.6 | 12.8 |
| 10000 | 6.3 | 18.7 |
| 100000 | 8.5 | 26.8 |

多一個smooth因素要考慮，因此影響到error

How smooth?

Select λ obtaining the best model

- Training error: larger λ , considering the training error less
- We prefer smooth function, but don't be too smooth.

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

Reference

- Bishop: Chapter 1.1

Acknowledgment

- 感謝 鄭凱文 同學發現投影片上的符號錯誤