

Regression

2020/10/28

Regression

- Stock Market Forecast

$$f(\text{台股加權股價指數}) = \text{台股加權股價指數}$$



- Self-driving Car

$$f(\text{方向盤角度}) = \text{方向盤角度}$$

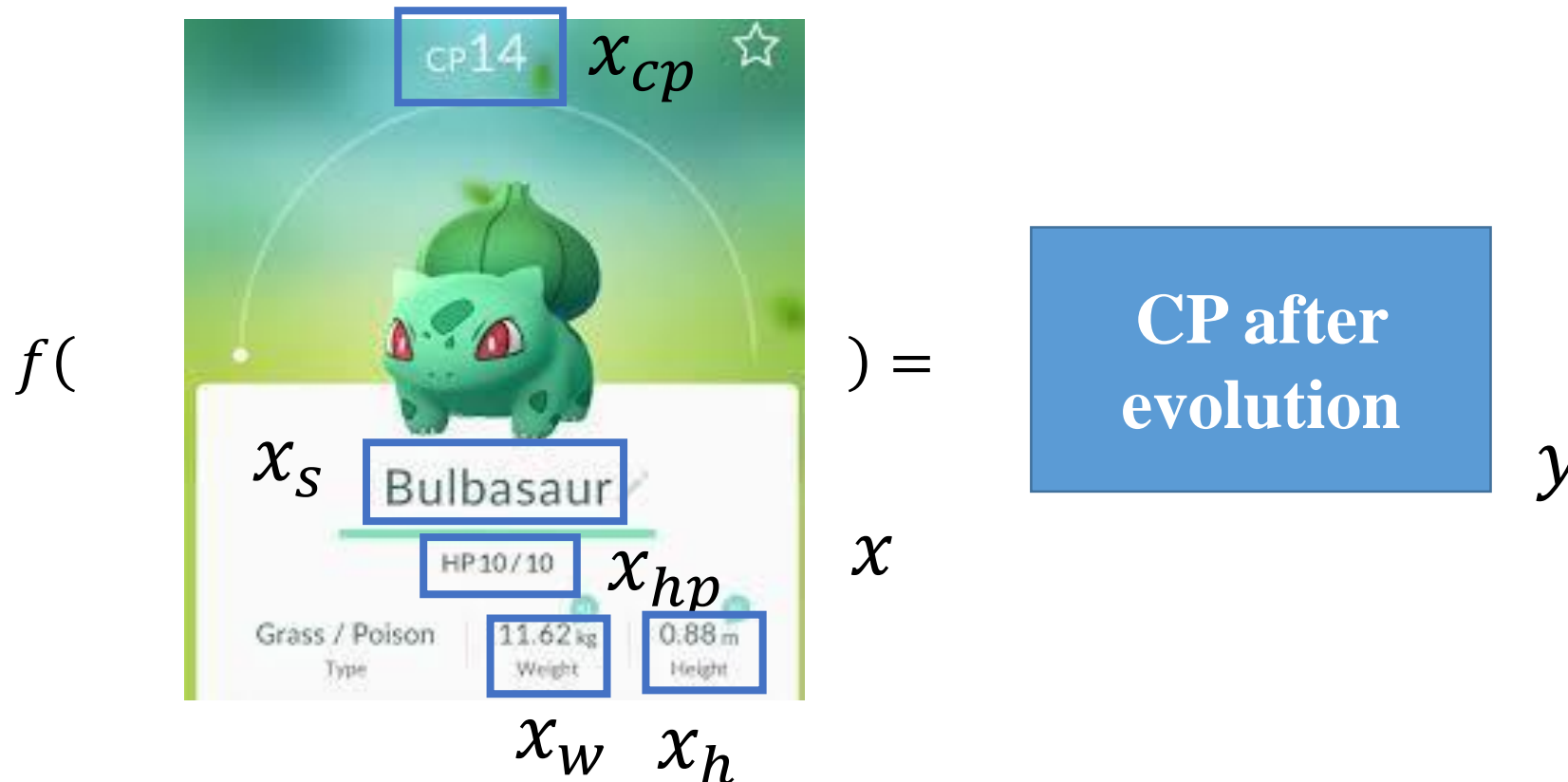


- Recommendation

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

Example Application

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



Step 1 : Model

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

Model

A set of
function

$f_1, f_2 \dots$

$f($



x

) =

CP after
evolution

y

.....infinite

Linear model:

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

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

$w_i: weight, b: bias$

feature

Step 2 : Goodness of Function

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

Model

A set of
function

$f_1, f_2 \dots$

Training
Data

function
input:

function
output (scalar):



Step 2 : Goodness of Function

Training data:

10 pokemons

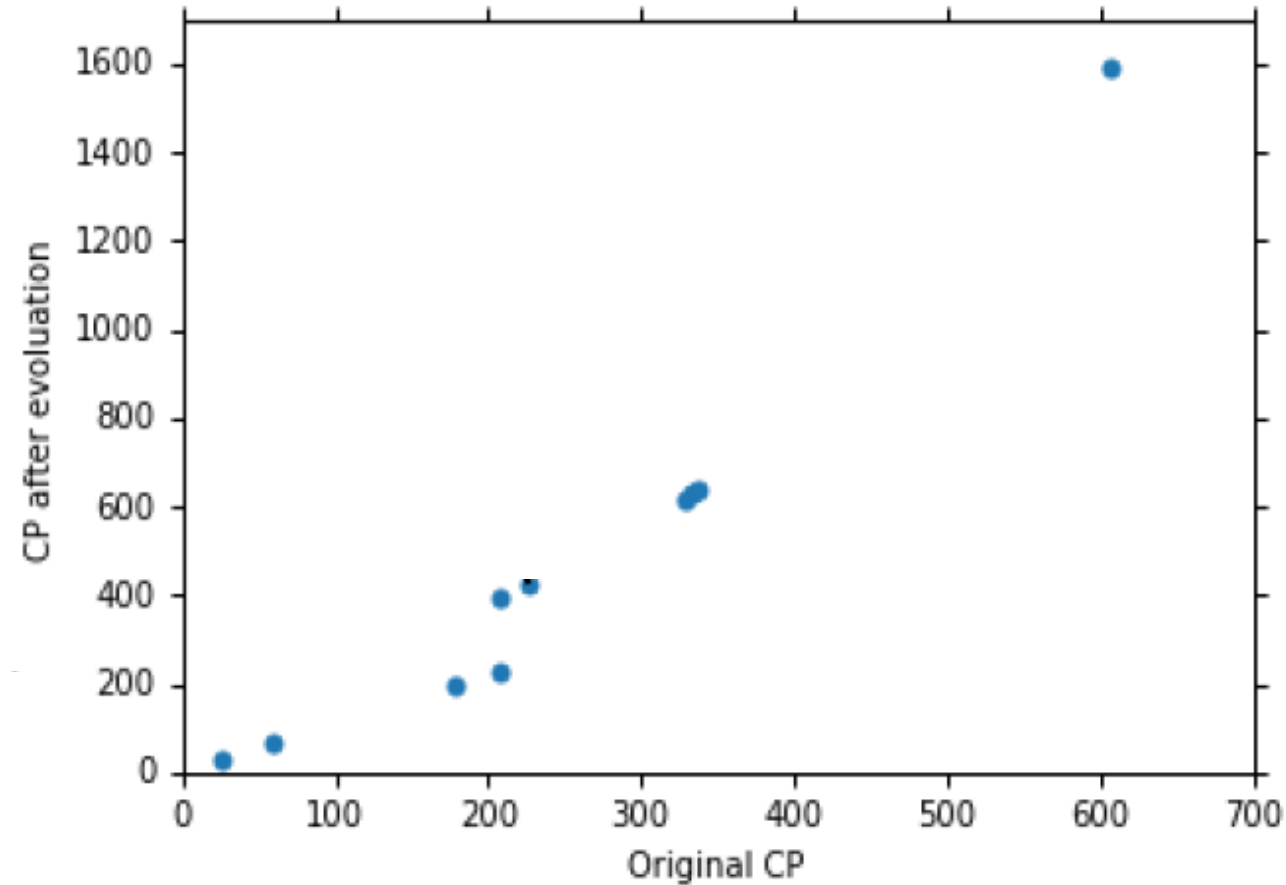
(x^1, \hat{y}^1)

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

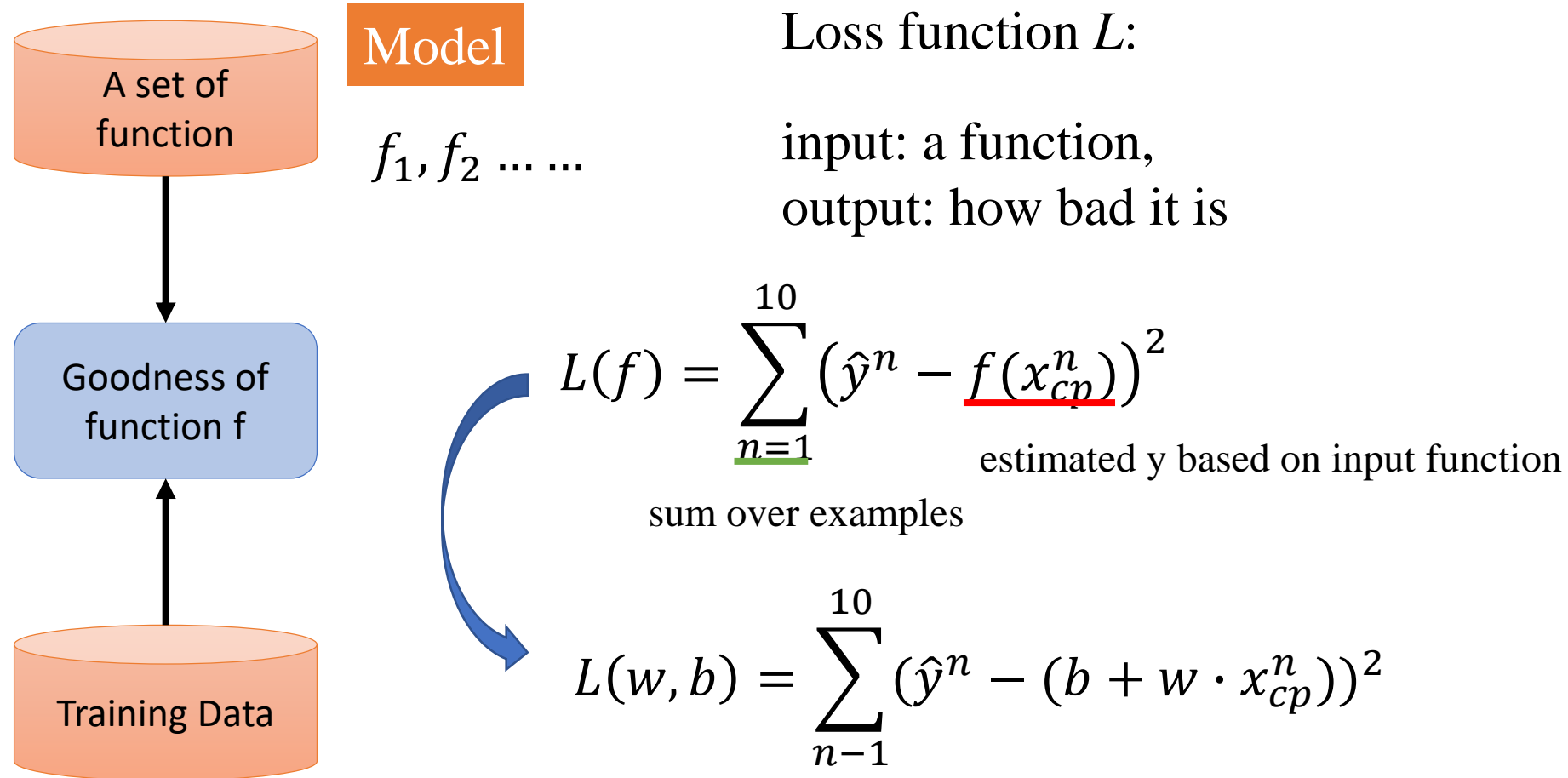
...

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

This is real data.



Step 2 : Goodness of Function



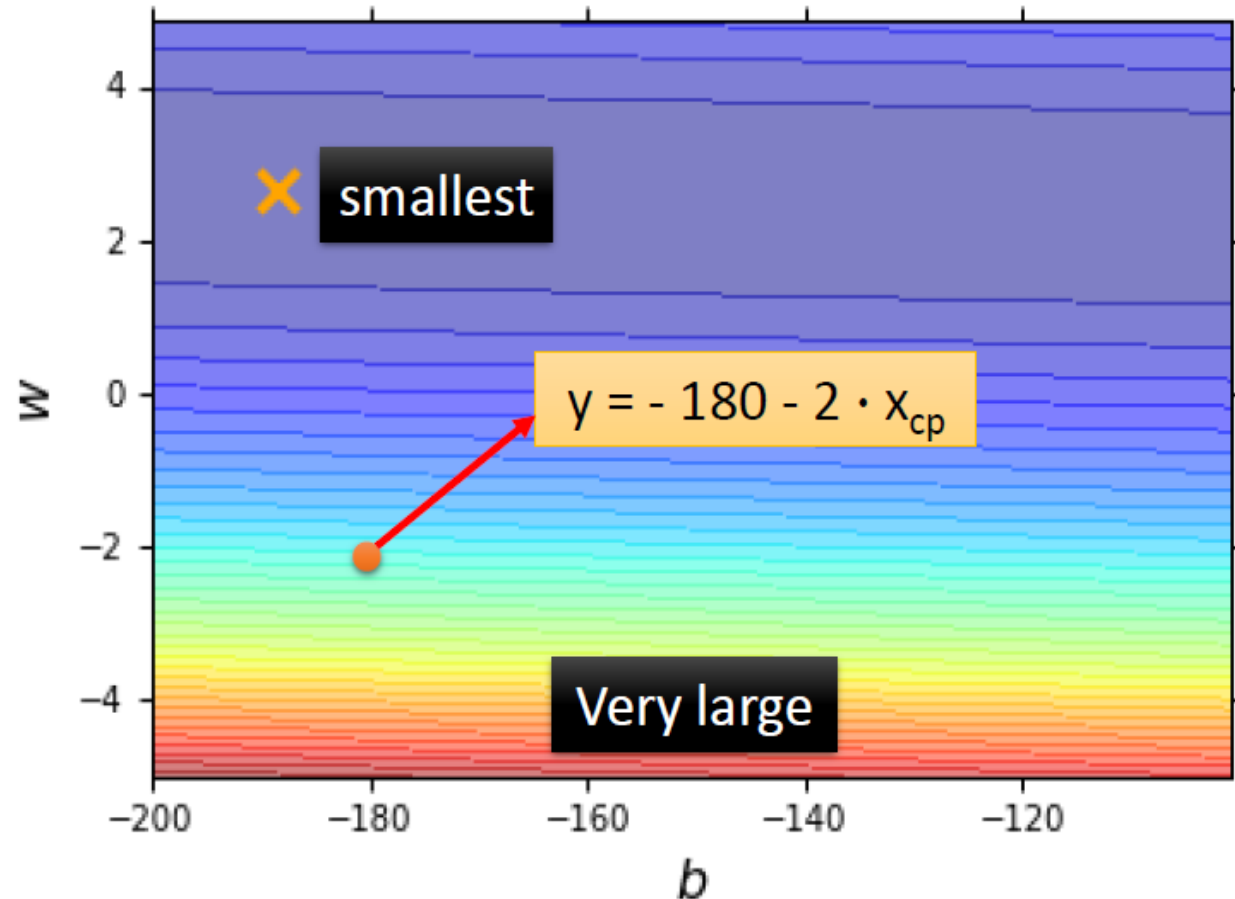
Step 2 : Goodness of Function

- Loss function

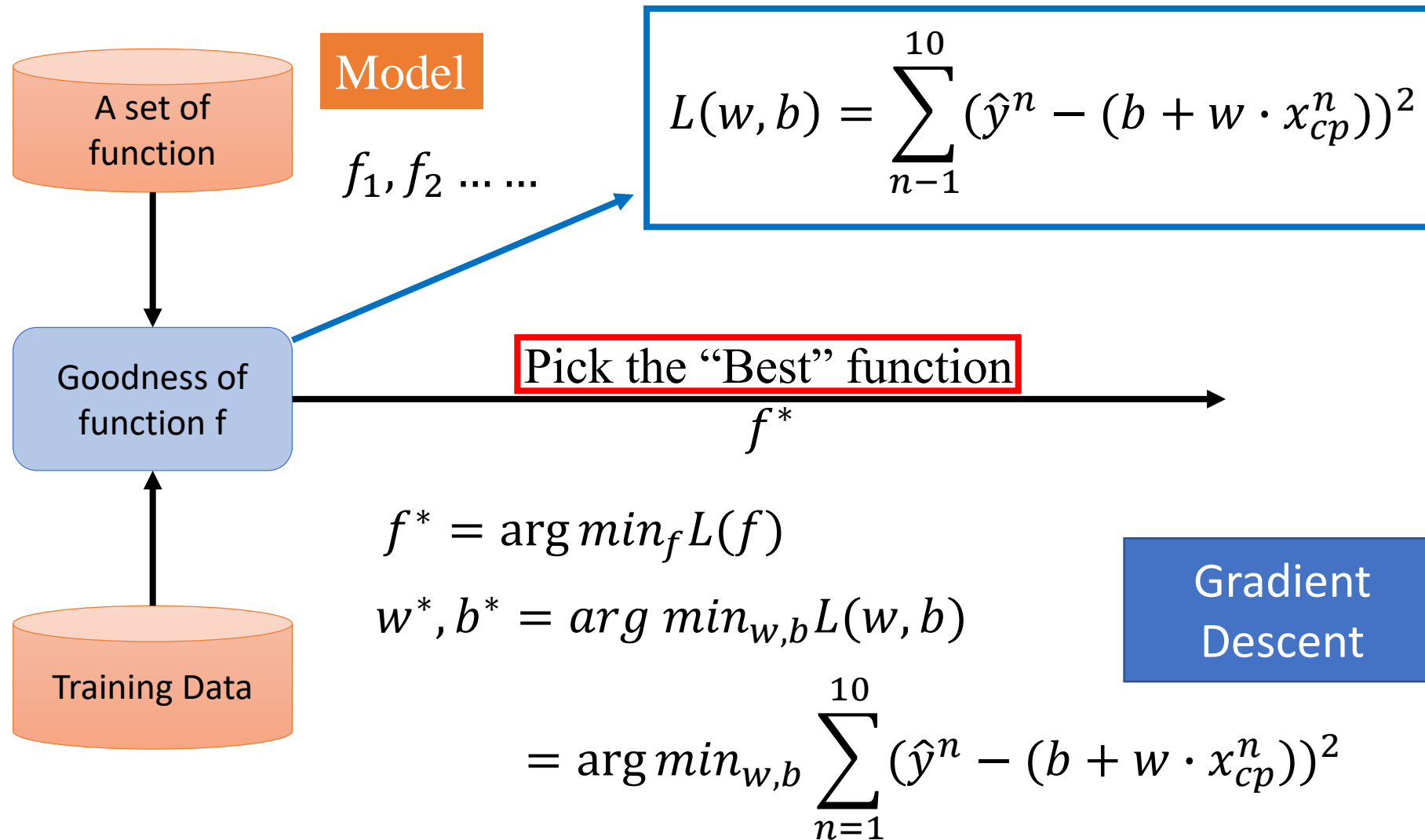
Each point in the figure is a function

The color represents $L(w, b)$

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



Step 3 : Best Function

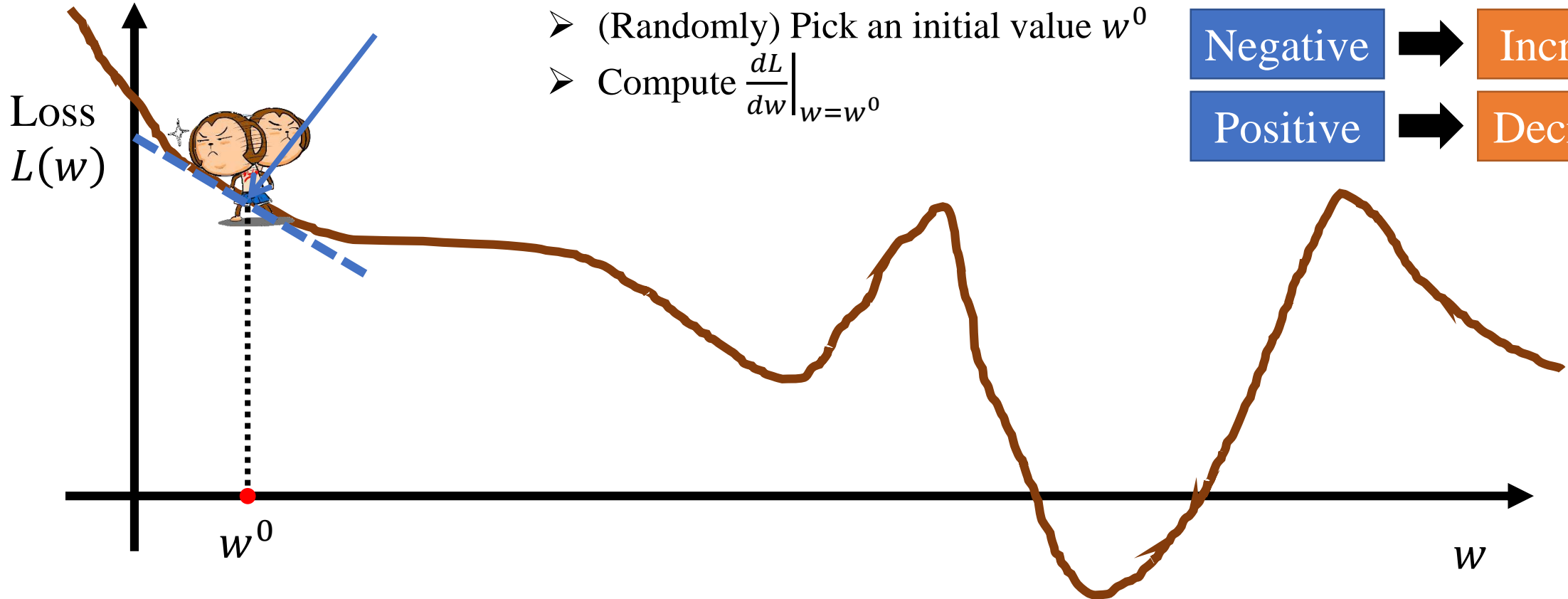
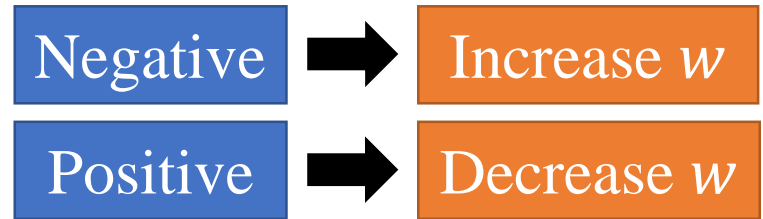


Step 3 : Gradient Descent

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

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

- (Randomly) Pick an initial value w^0
- Compute $\frac{dL}{dw} \Big|_{w=w^0}$

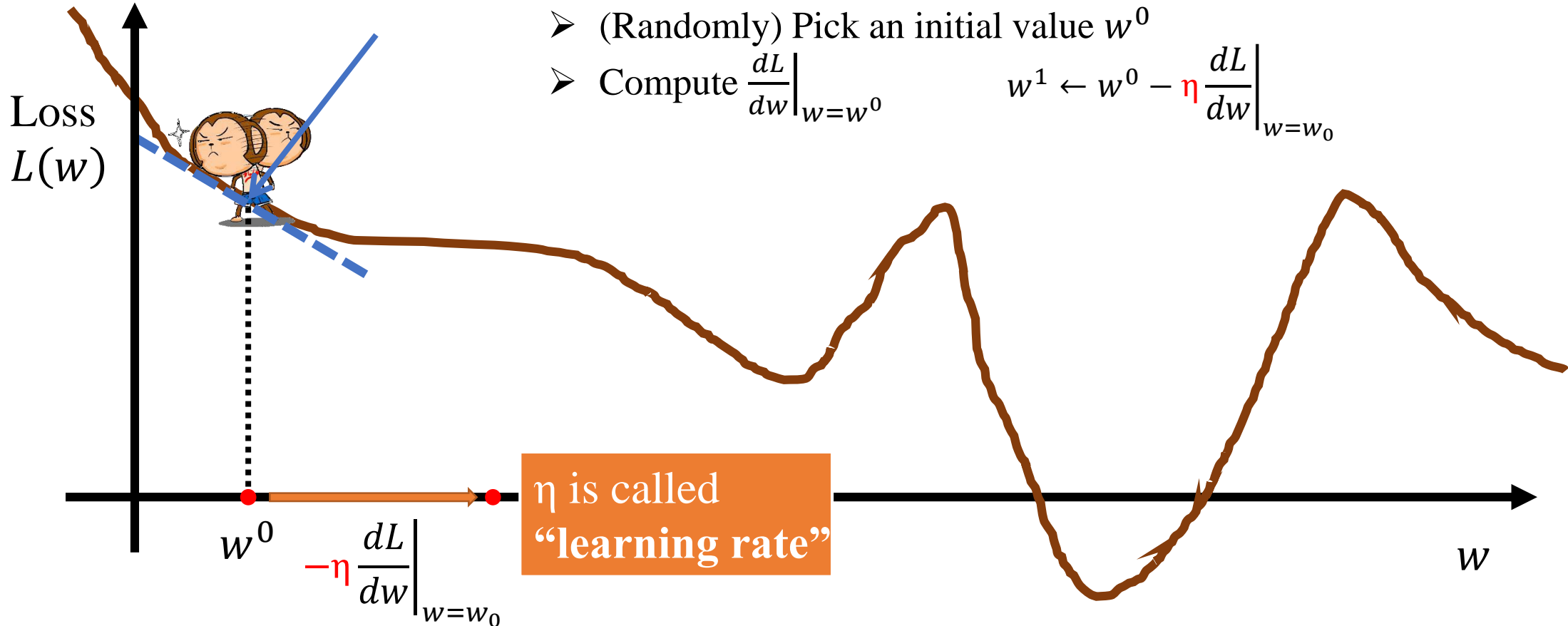


Step 3 : Gradient Descent

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

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

- (Randomly) Pick an initial value w^0
- Compute $\frac{dL}{dw}\bigg|_{w=w^0}$ $w^1 \leftarrow w^0 - \eta \frac{dL}{dw}\bigg|_{w=w^0}$

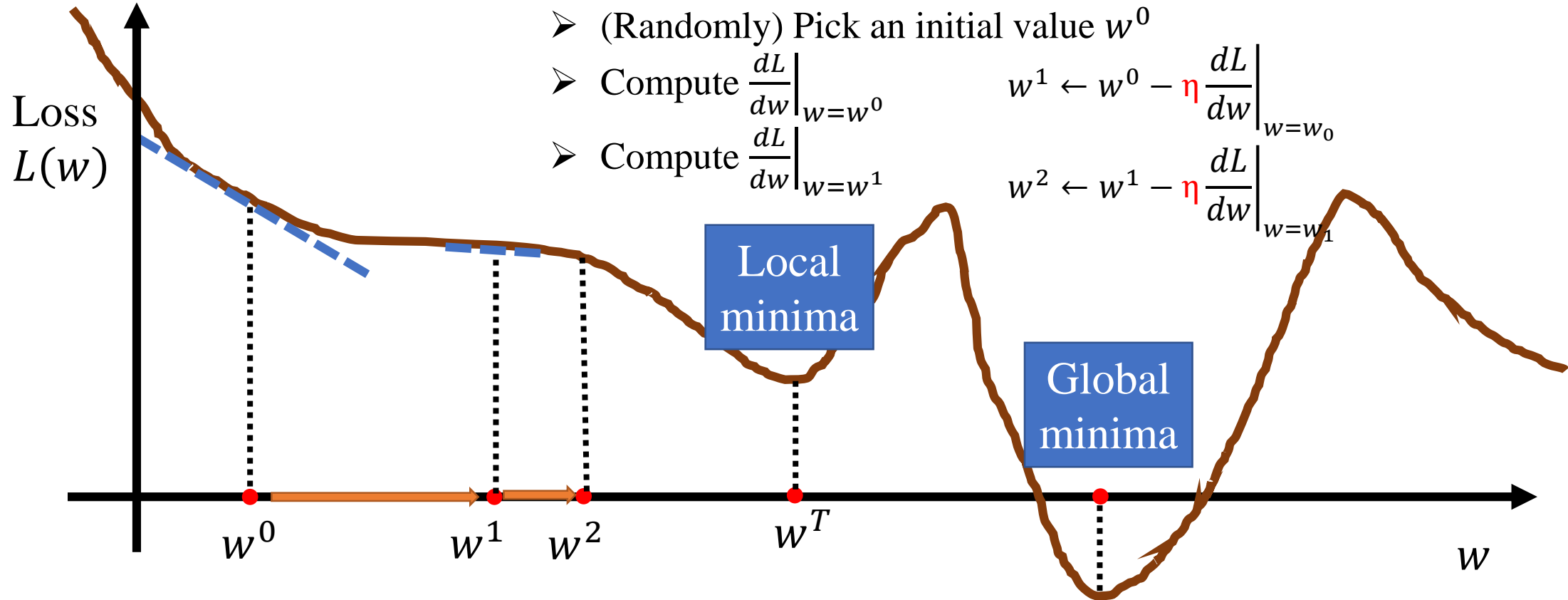


Step 3 : Gradient Descent

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

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

- (Randomly) Pick an initial value w^0
- Compute $\left. \frac{dL}{dw} \right|_{w=w^0}$ $w^1 \leftarrow w^0 - \eta \left. \frac{dL}{dw} \right|_{w=w^0}$
- Compute $\left. \frac{dL}{dw} \right|_{w=w^1}$ $w^2 \leftarrow w^1 - \eta \left. \frac{dL}{dw} \right|_{w=w^1}$



Step 3 : Gradient Descent

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

- How about two parameters?

$$w^*, b^* = \arg \min_{w, b} L(w, b)$$

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

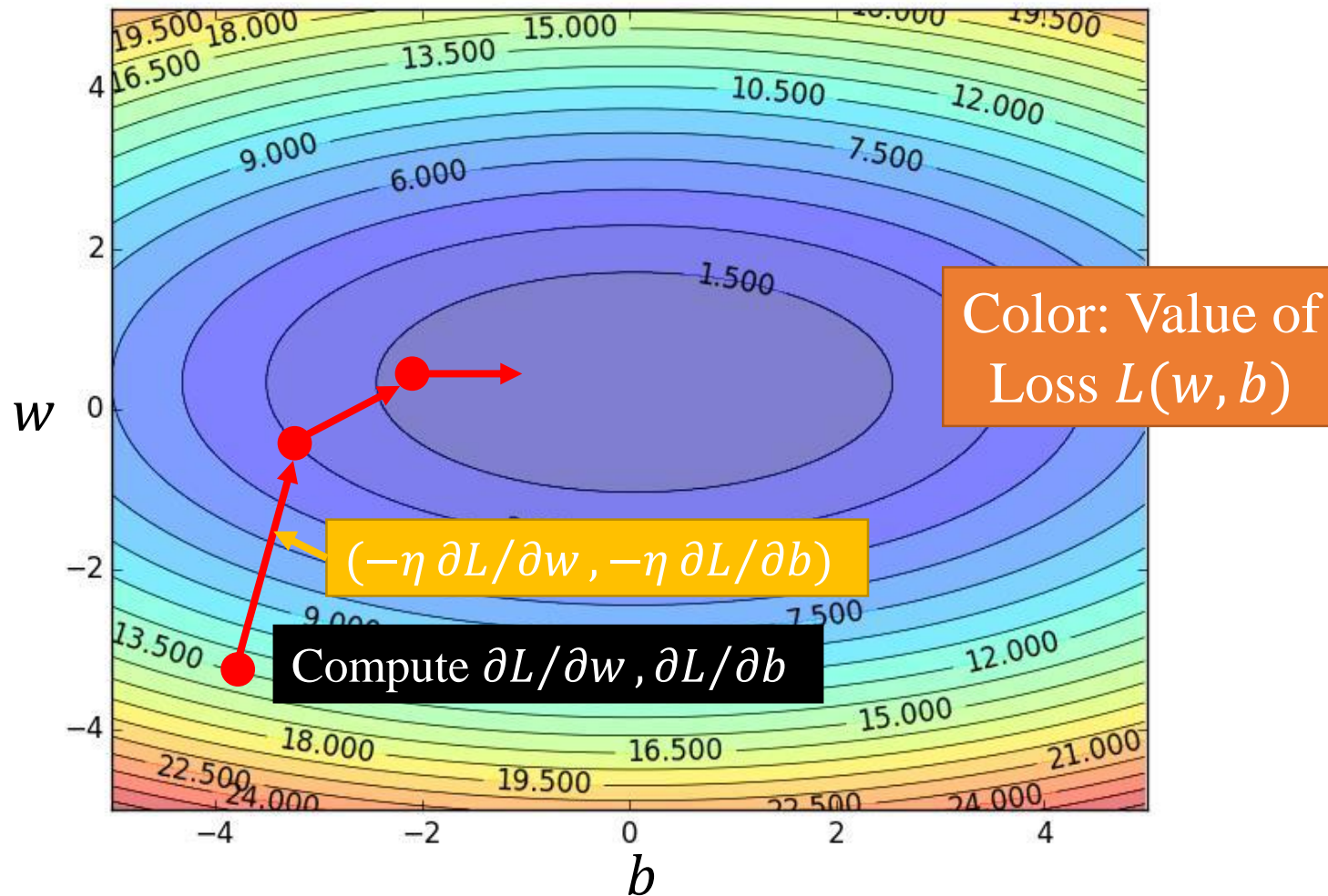
➤ Compute $\frac{\partial L}{\partial w} \Big|_{w=w^0, b=b^0}, \frac{\partial L}{\partial b} \Big|_{w=w^0, b=b^0}$

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

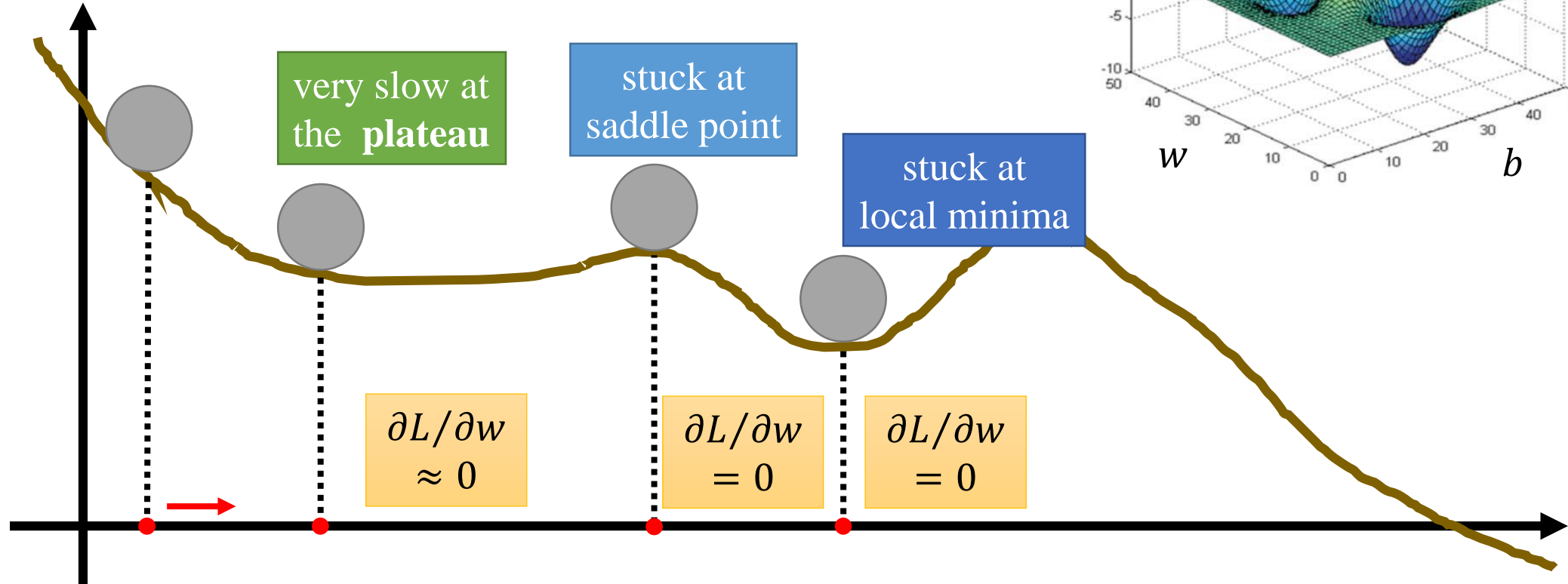
➤ Compute $\frac{\partial L}{\partial w} \Big|_{w=w^1, b=b^1}, \frac{\partial L}{\partial b} \Big|_{w=w^1, b=b^1}$

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

Step 3 : Gradient Descent




Step 3 : Gradient Descent



Step 3 : Gradient Descent

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

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


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

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

How's the results?

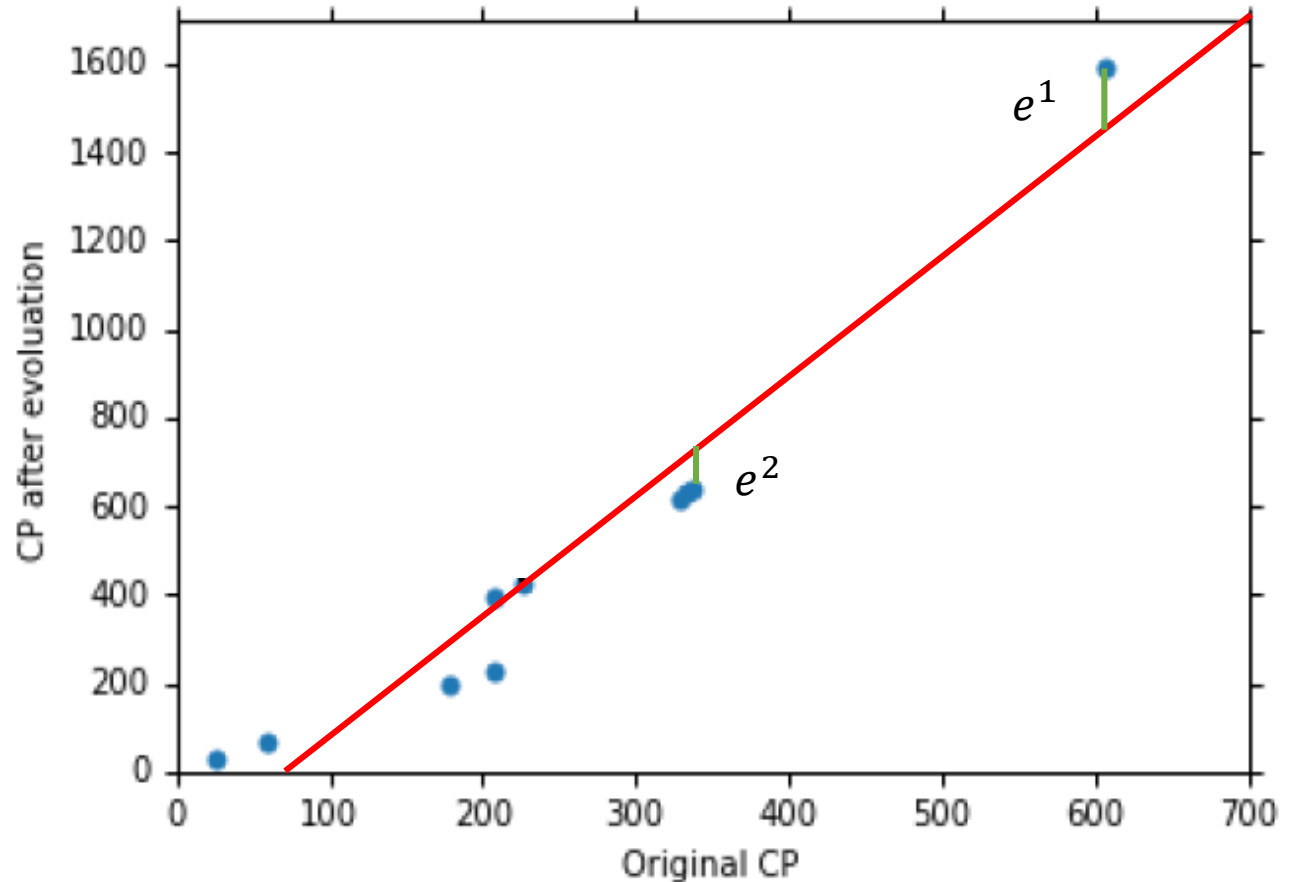
$$y = b + w \cdot x_{cp}$$

$$b = -188.4$$

$$w = 2.7$$

average error on
training data

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



How's the results?

- Generalization

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

$$b = -188.4$$

$$w = 2.7$$

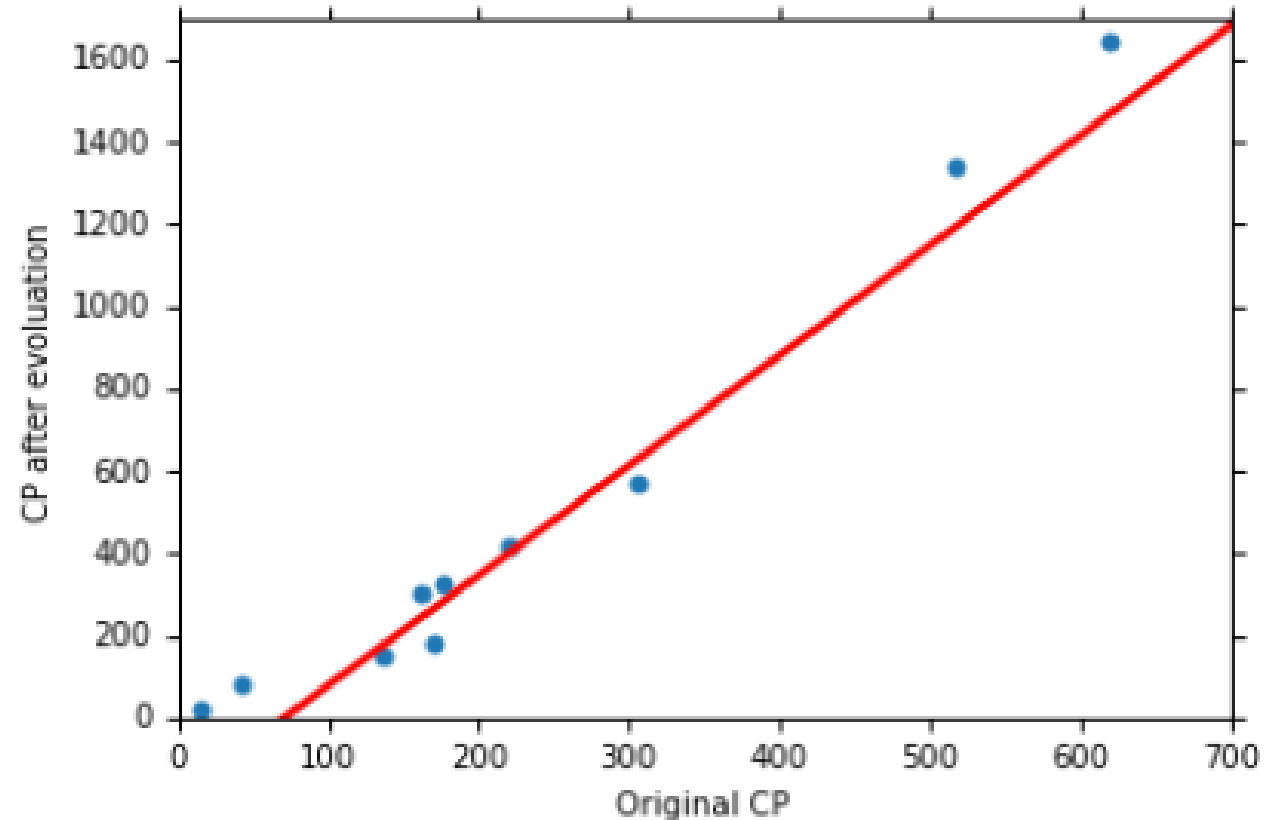
average error on
training data

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

> average error on
training data (31.9)

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

Another 10 pokemons as testing data



Selecting another model

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

best function:

$$b = -10.3$$

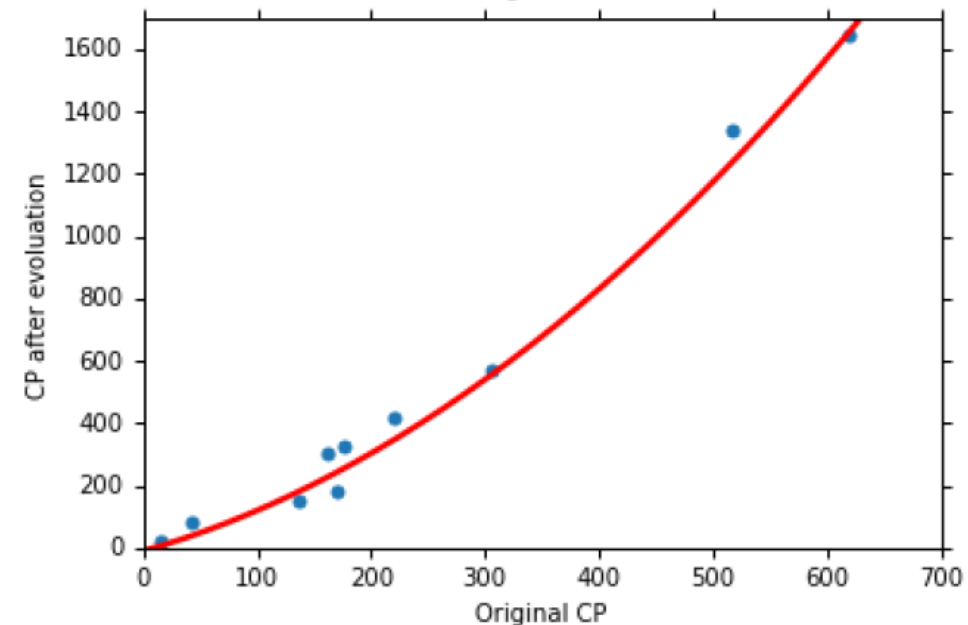
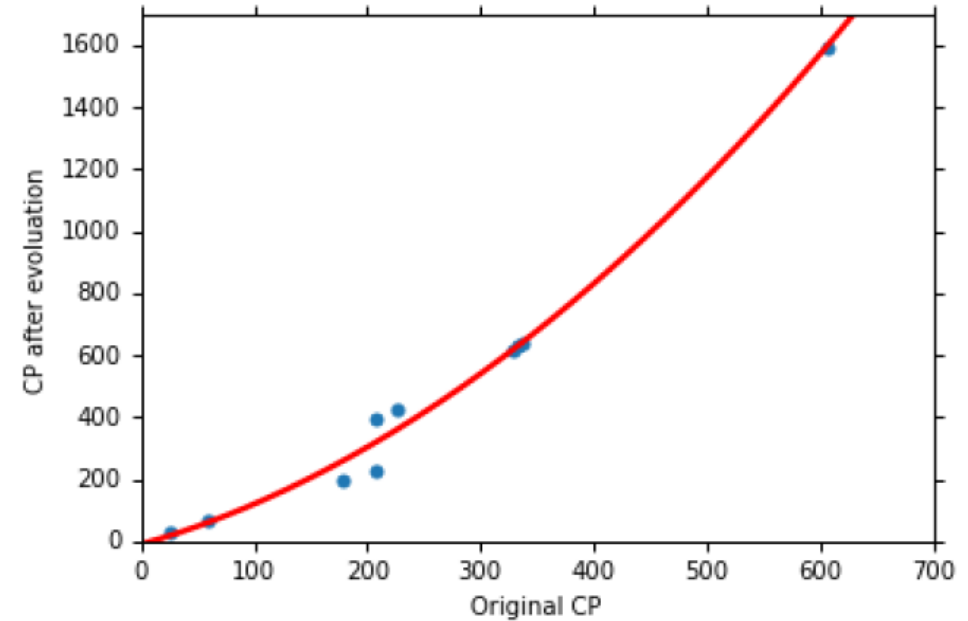
$$w_1 = 1.0, w_2 = 2.7 \times 10^{-3}$$

average error = 15.4

testing:

average error = 18.4

better! could it be even better?



Selecting another model

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

best function:

$$b = 6.4$$

$$w_1 = 0.66, w_2 = 4.3 \times 10^{-3},$$

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

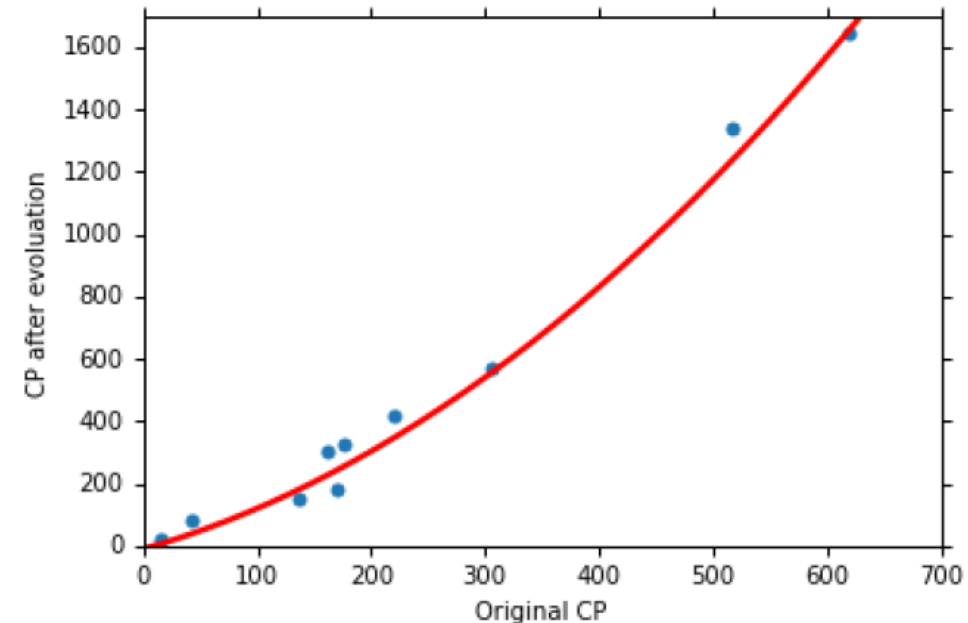
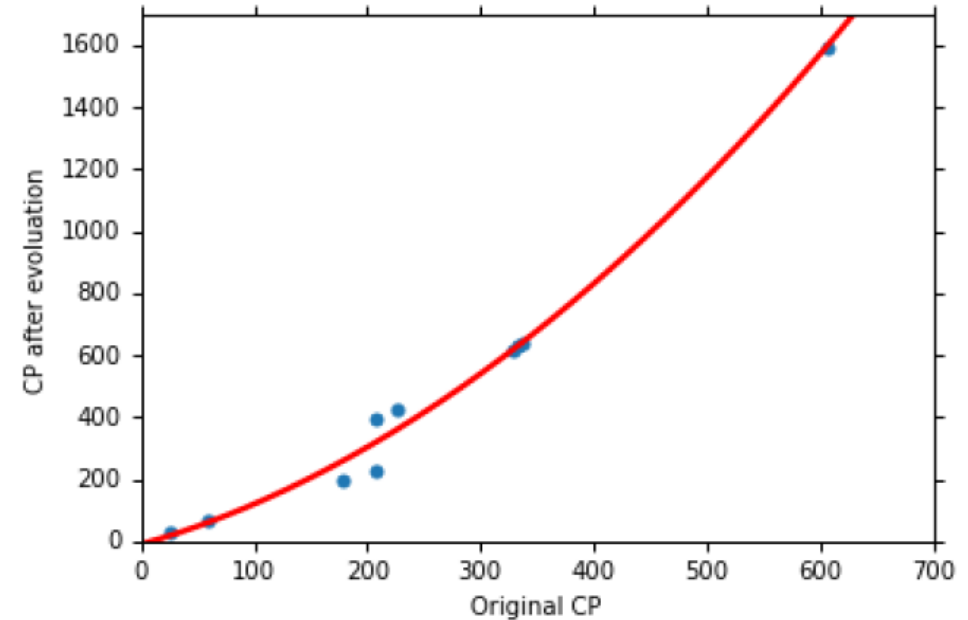
average error = 15.3

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

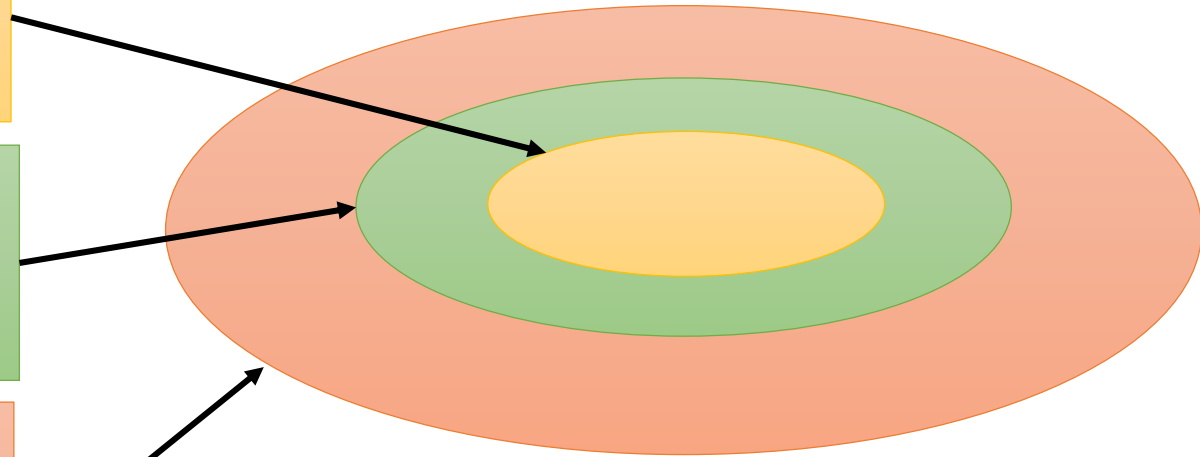
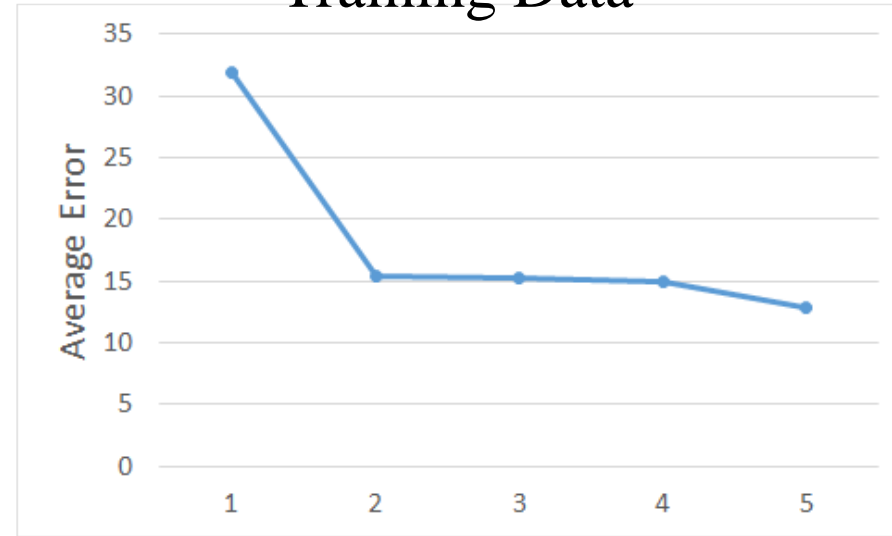
2. $y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2$

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

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

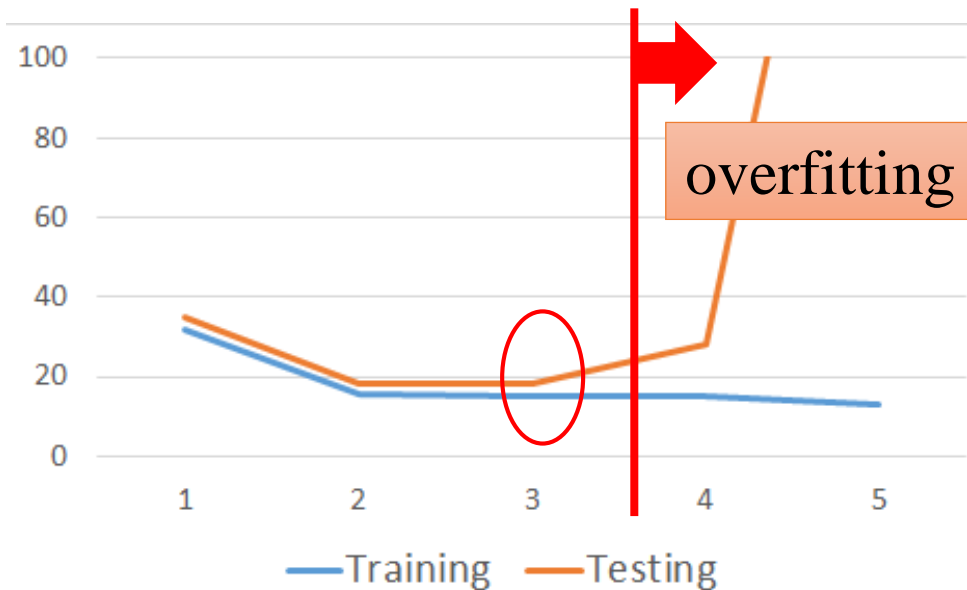
5. $y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2 + w_3 \cdot (x_{cp})^3 + w_4 \cdot (x_{cp})^4 + w_5 \cdot (x_{cp})^5$

Training Data



a more complex model yields
lower error on training data.
if we can truly find the best function

Model Selection

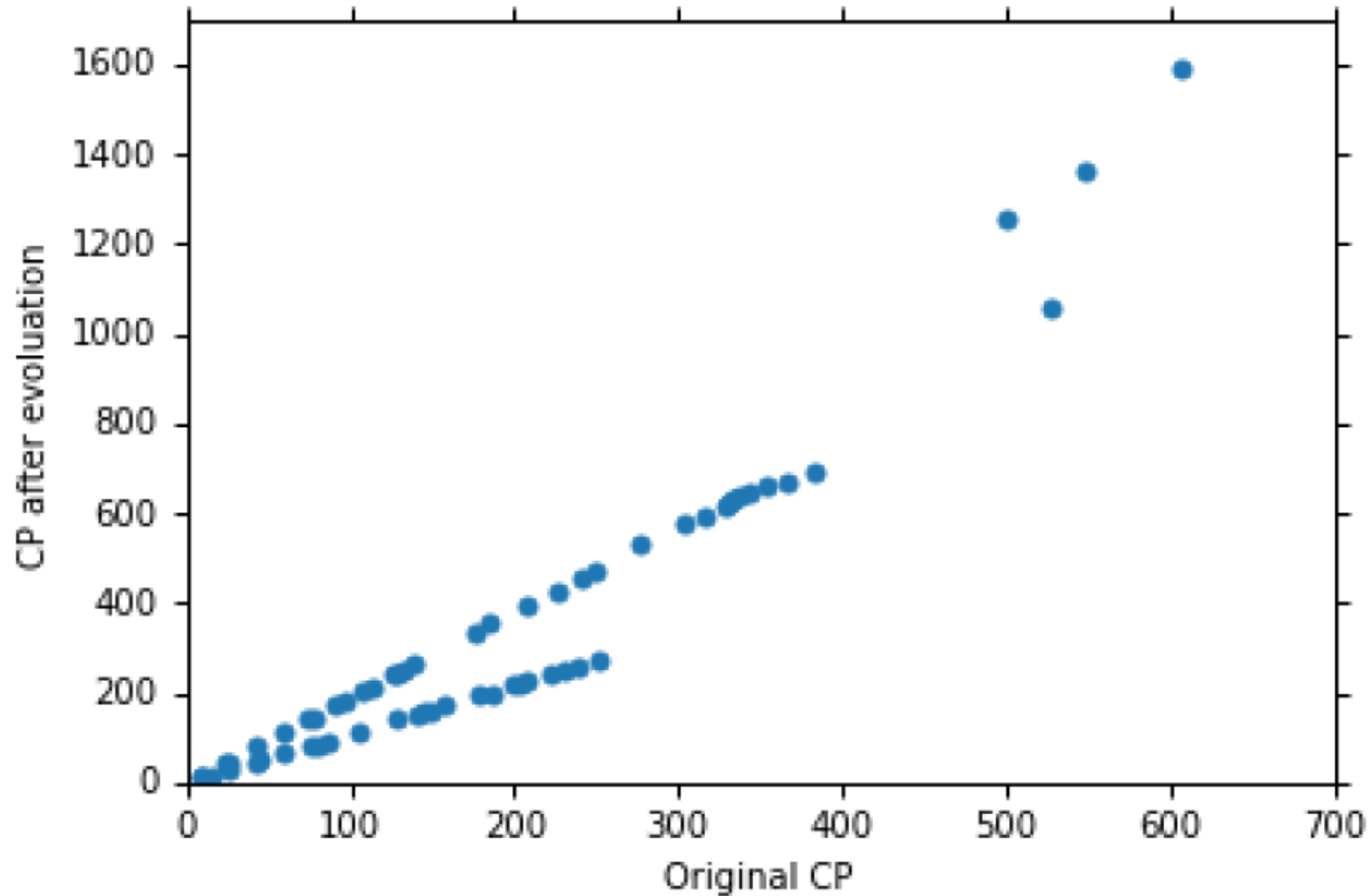


	Training	Testing
1	31.9	35.0
2	15.4	18.4
3	15.3	18.1
4	14.9	28.2
5	12.8	232.1

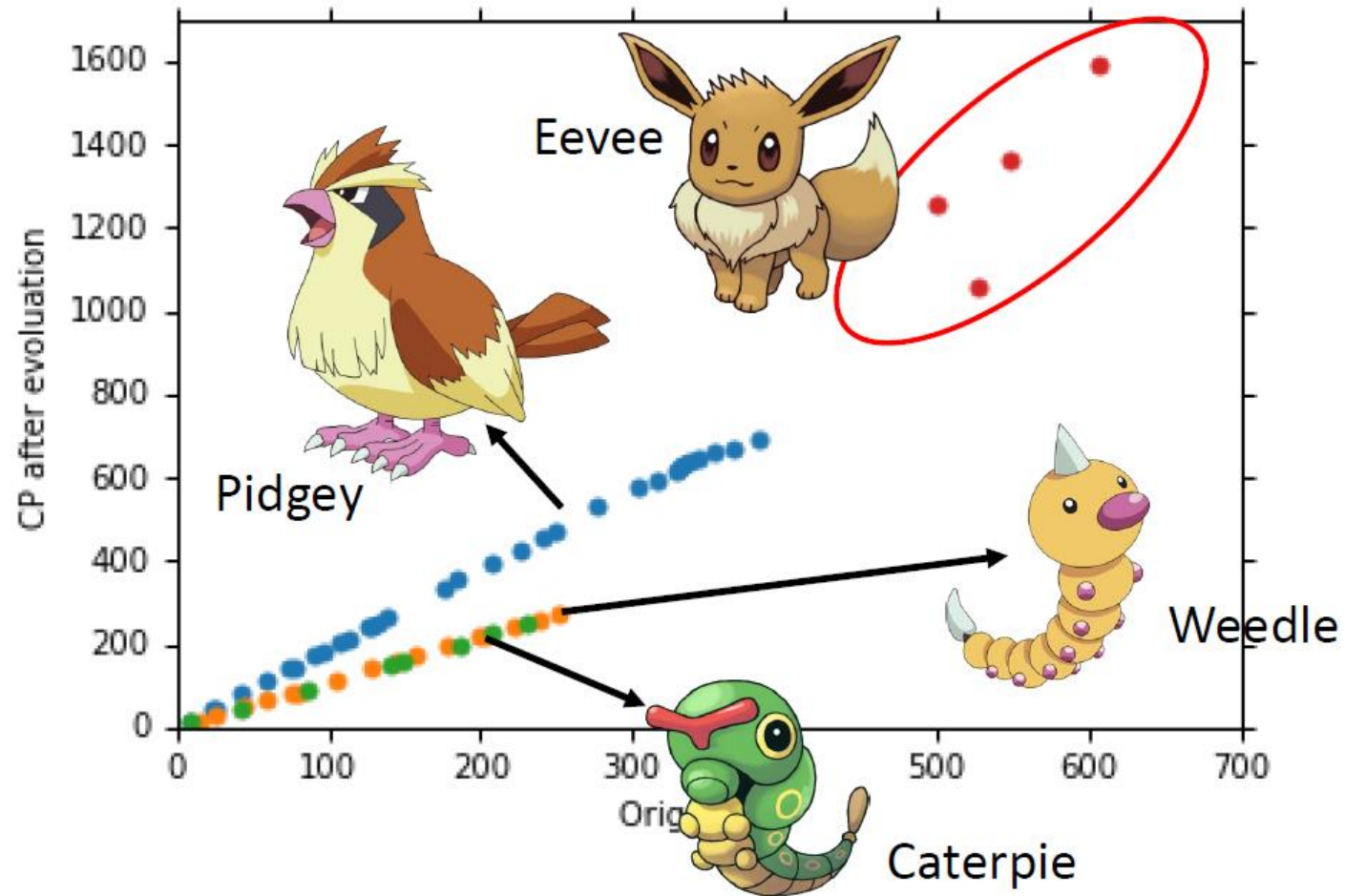
a more complex model does not always lead to better performance on testing data.

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?

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

$$\begin{aligned} y = & b_1 \cdot \delta(x_s = \textit{Pidgey}) \\ & + w_1 \cdot \delta(x_s = \textit{Pidgey})x_{cp} \\ & + b_2 \cdot \delta(x_s = \textit{Weedle}) \\ & + w_2 \cdot \delta(x_s = \textit{Weedle})x_{cp} \\ & + b_3 \cdot \delta(x_s = \textit{Caterpie}) \\ & + w_3 \cdot \delta(x_s = \textit{Caterpie})x_{cp} \\ & + b_4 \cdot \delta(x_s = \textit{Eevee}) \end{aligned}$$

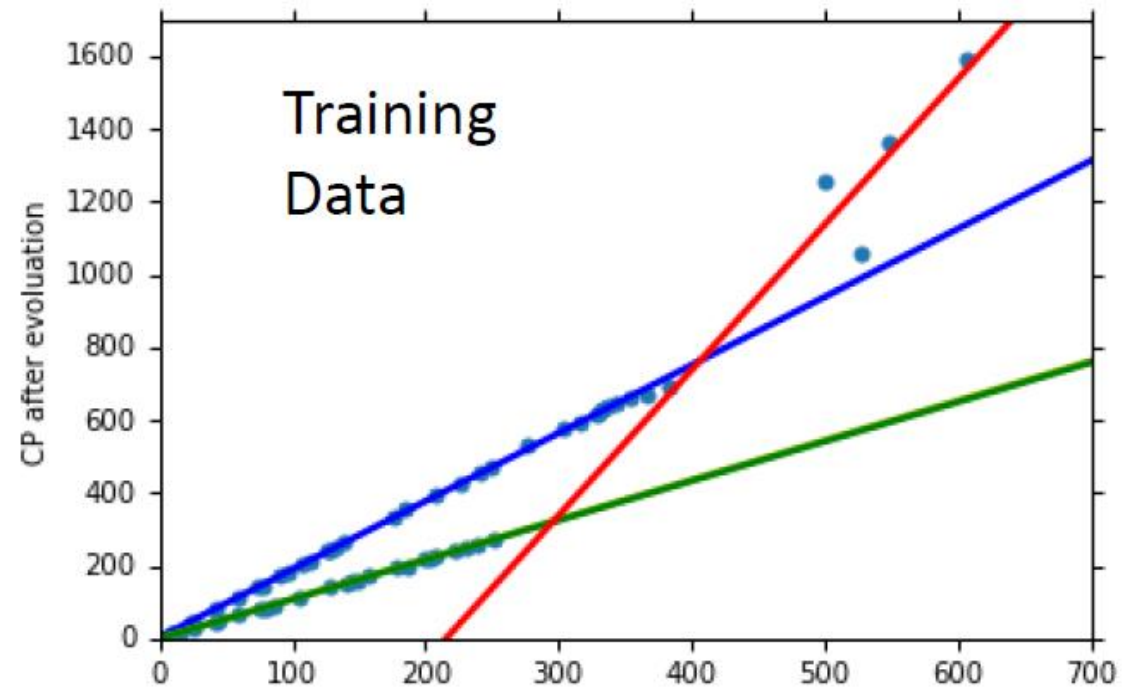
$$y = b + \sum w_i x_i$$

linear model?

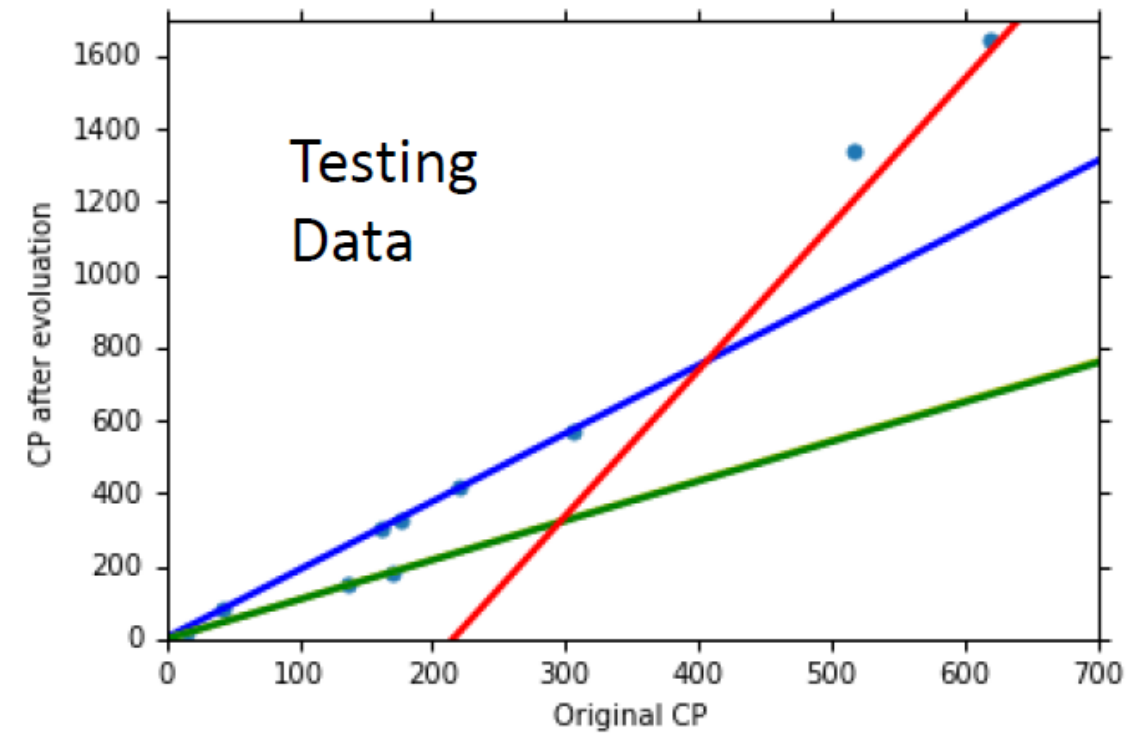
$$\delta(x_s = \textit{Pidgey})$$

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

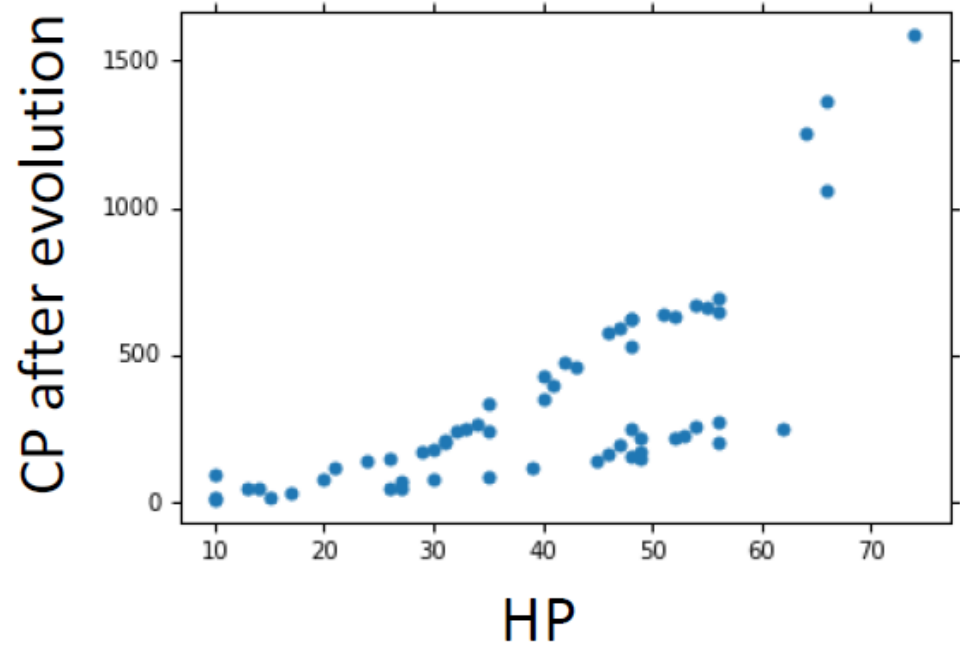
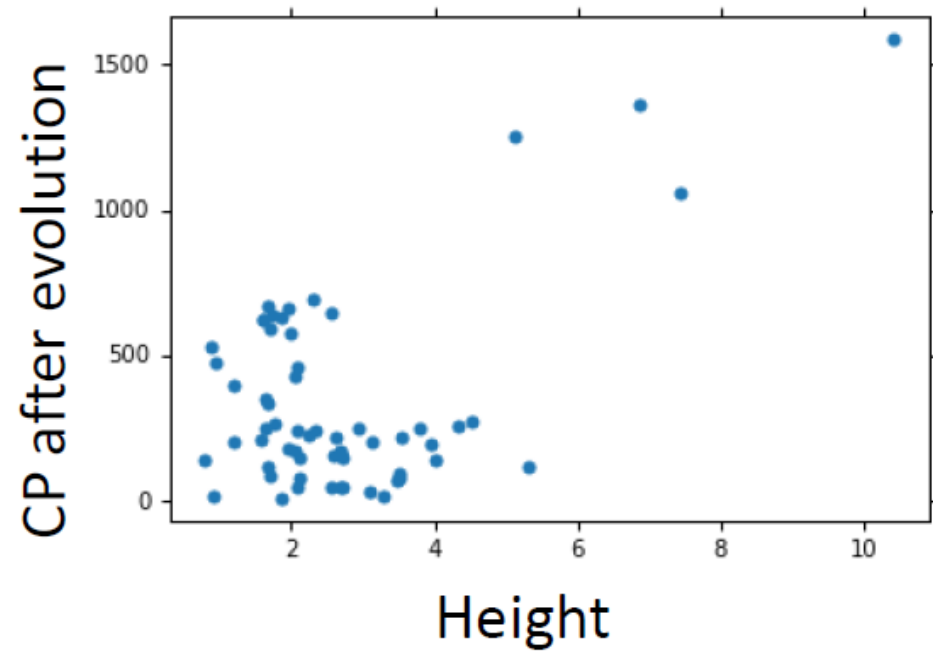
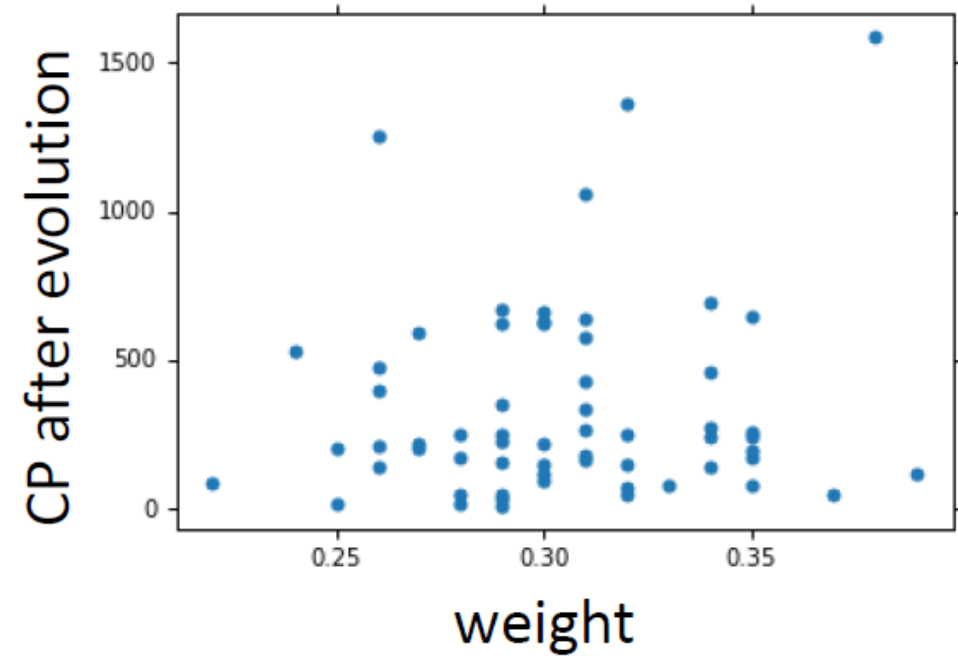
average error = 3.8



average error = 14.3



are there any other hidden factors?



Back to step 1:
redesign the model again

x



if $x_s = Pidgey$:

$$y' = b_1 + w_1 \cdot x_{cp} + w_5 \cdot (x_{cp})^2$$

if $x_s = Weedle$:

$$y' = b_2 + w_2 \cdot x_{cp} + w_6 \cdot (x_{cp})^2$$

if $x_s = Caterpie$:

$$y' = b_3 + w_3 \cdot x_{cp} + w_7 \cdot (x_{cp})^2$$

if $x_s = Eevee$:

$$y' = b_4 + w_4 \cdot x_{cp} + w_8 \cdot (x_{cp})^2$$

$$y = y' + w_9 \cdot x_{hp} + w_{10} \cdot (x_{hp})^2$$

$$+ w_{11} \cdot x_h + w_{12} \cdot (x_h)^2 + w_{13} \cdot x_w + w_{14} \cdot (x_w)^2$$



y

Training error
=1.9

Testing error
=102.3

overfitting!

Back to step 2: Regularization

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

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

The functions with
smaller w_i are better

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

➤ Smaller w_i means... smoother

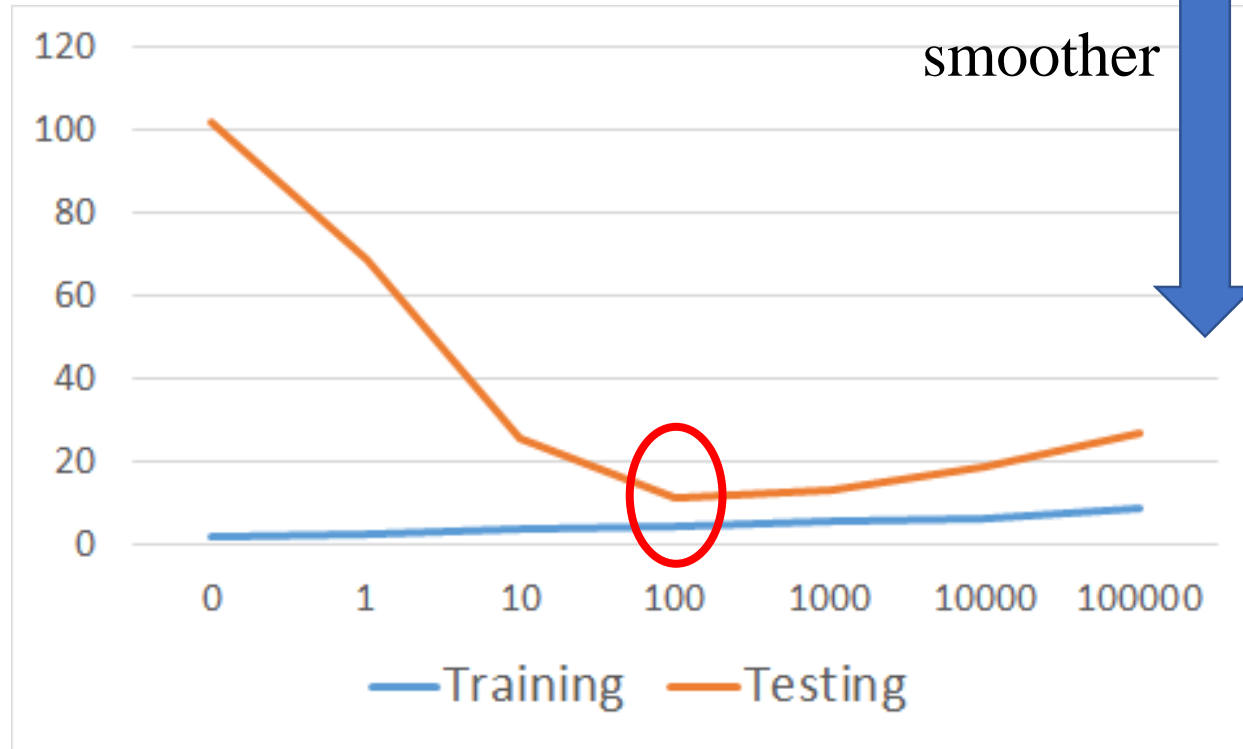
$$y = b + \sum w_i x_i$$

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

➤ 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

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.