Regression

2022/10/12

http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2017/Lecture/Regression.pdf

Regression

- Stock Market Forecast

$f($  $) =$ 台股加權股價指數

- Self-driving Car

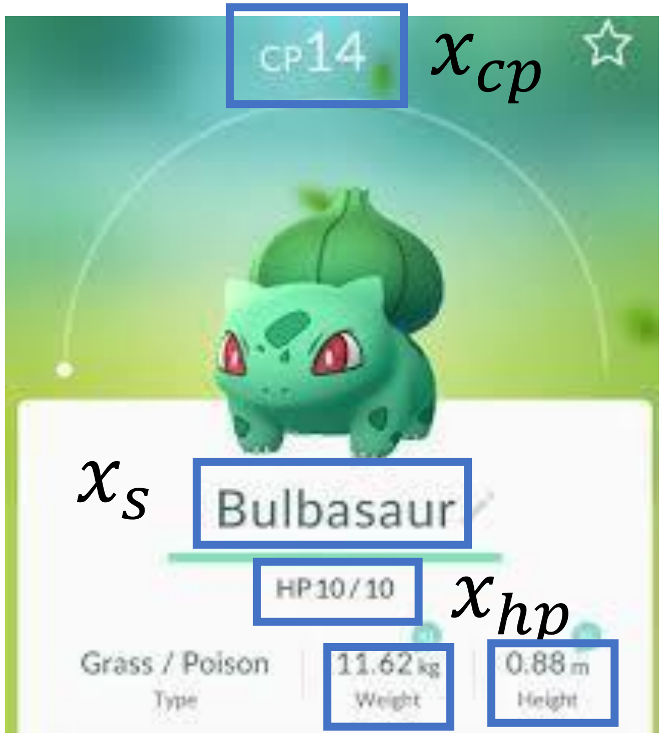
$f($  $) =$ 方向盤角度

- Recommendation

$f($ 使用者A 商品B $) =$ 購買可能性

Example Application

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

$f($ 

x_{cp} x_s x_{hp} x_w x_h

$) =$ **CP after evolution**

x y

Step 1 : Model

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

w and b are parameters
(can by 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

Model

A set of function

$f_1, f_2 \dots\dots$

$f($  x $) =$ y

CP after evolution

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

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

$w_i: weight, b: bias$

Step 2 : Goodness of Function

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

Model

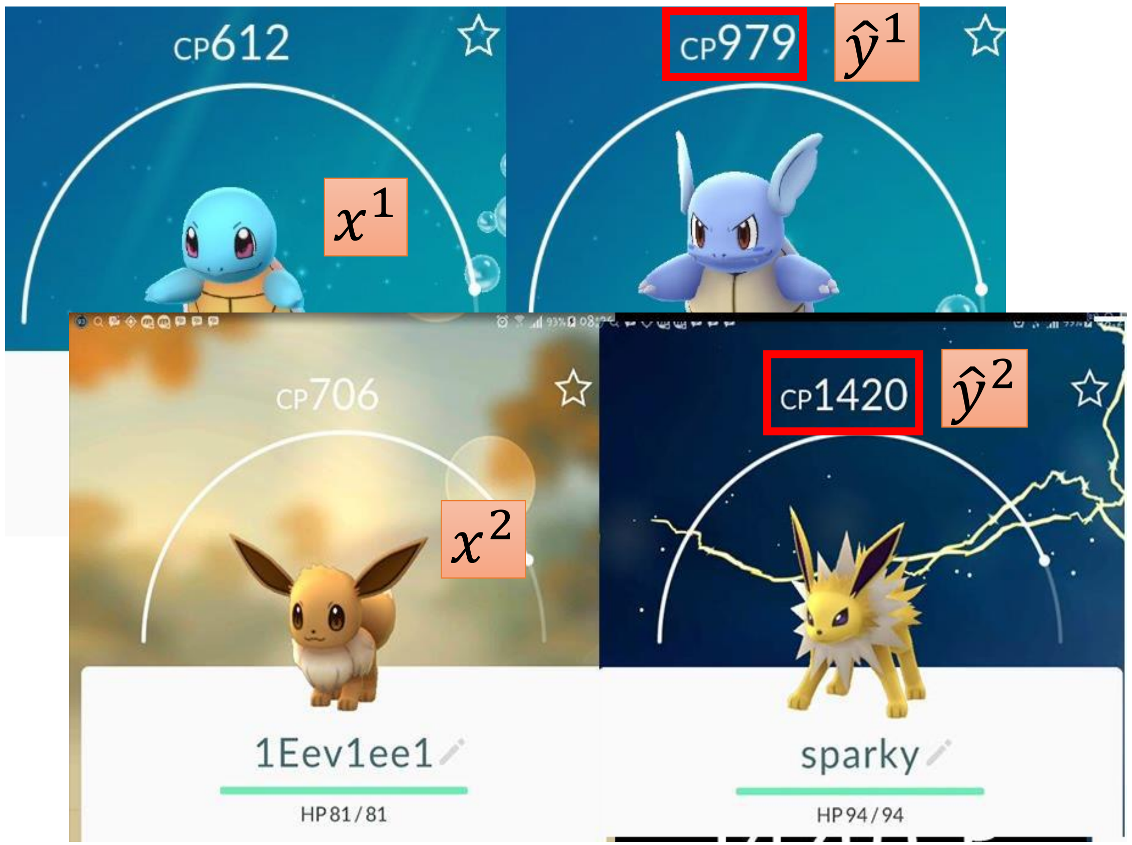
A set of function

$f_1, f_2 \cdots \cdots$

Training Data

function input:

function output (scalar):



Step 2 : Goodness of Function

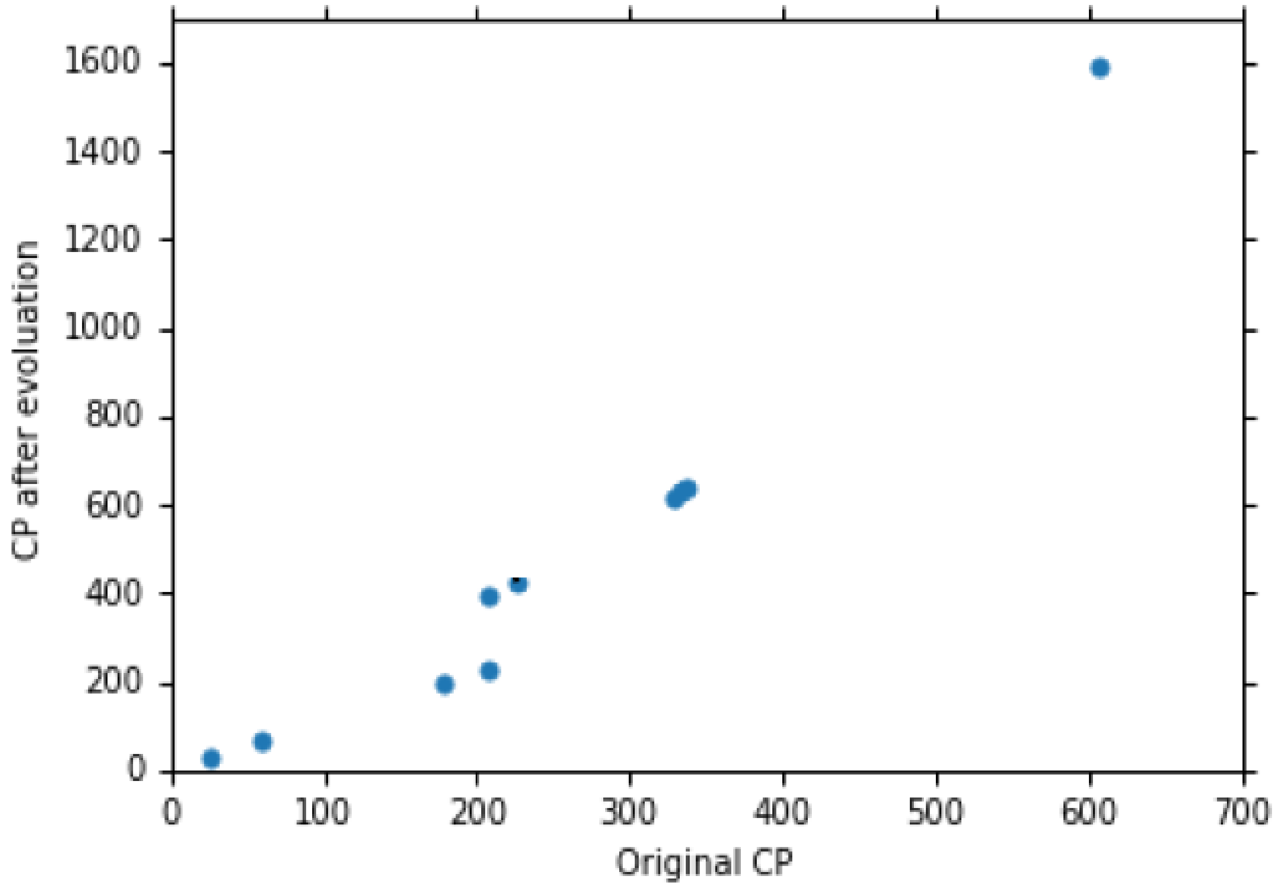
Training data:
10 pokemons

$(x^1, \hat{y}^1)$
$(x^2, \hat{y}^2)$

$\vdots$

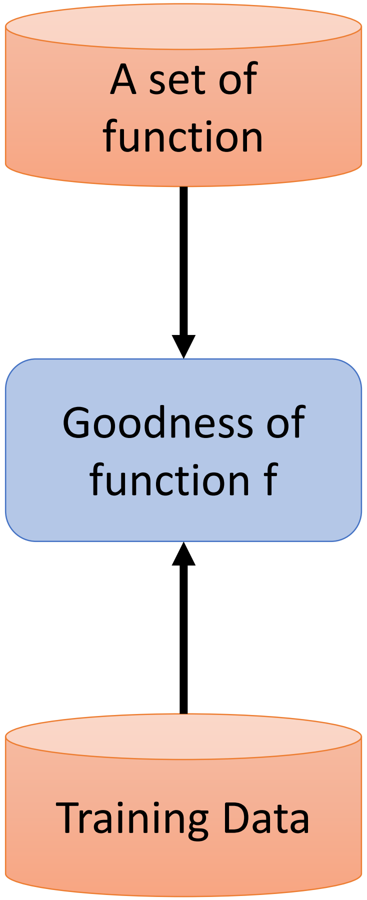
$(x^{10}, \hat{y}^{10})$

This is real data.



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

Step 2 : Goodness of Function



Model

$f_1, f_2 \dots \dots$

Loss function L:

input: a function,
output: how bad it is

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

estimated y based on input function

sum over examples

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

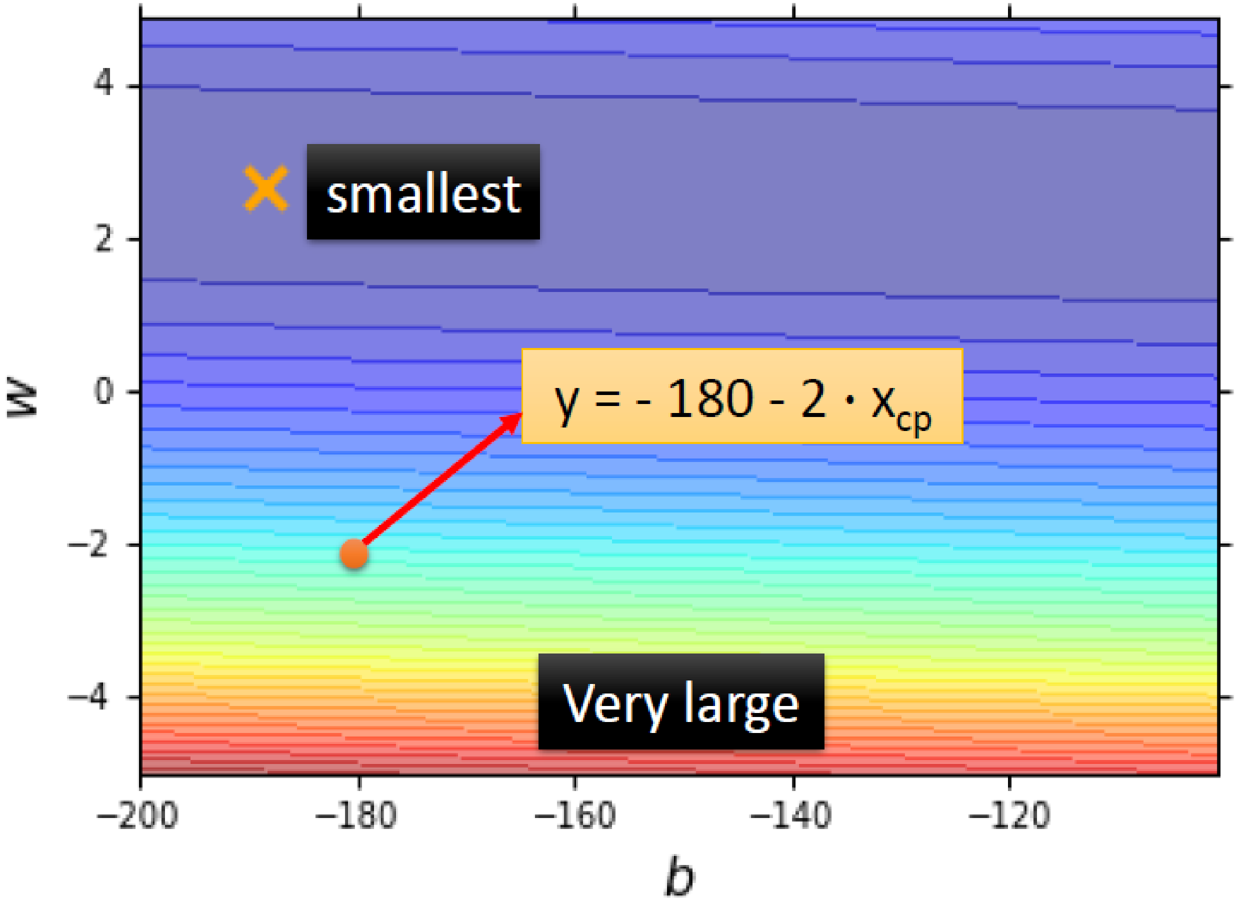
Step 2 : Goodness of Function

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

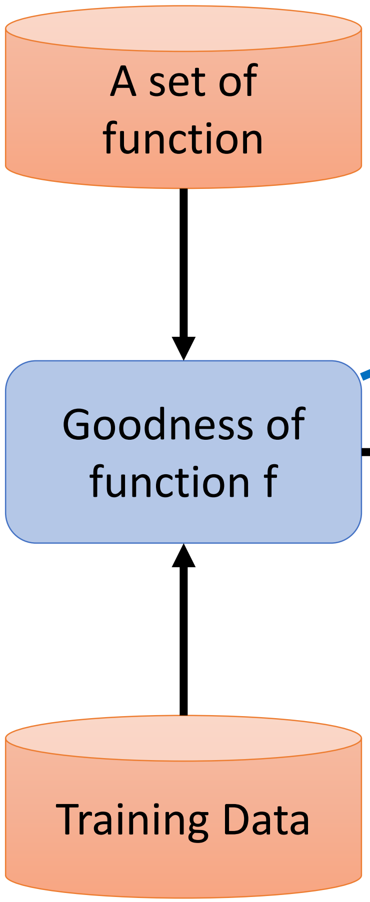
- Loss function

Each point in the figure is a function

The color represents $L(w,b)$



Step 3 : Best Function



Model

$f_1, f_2 \dots \dots$

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

Pick the "Best" function

f^*

$$f^* = \arg min_f L(f)$$

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

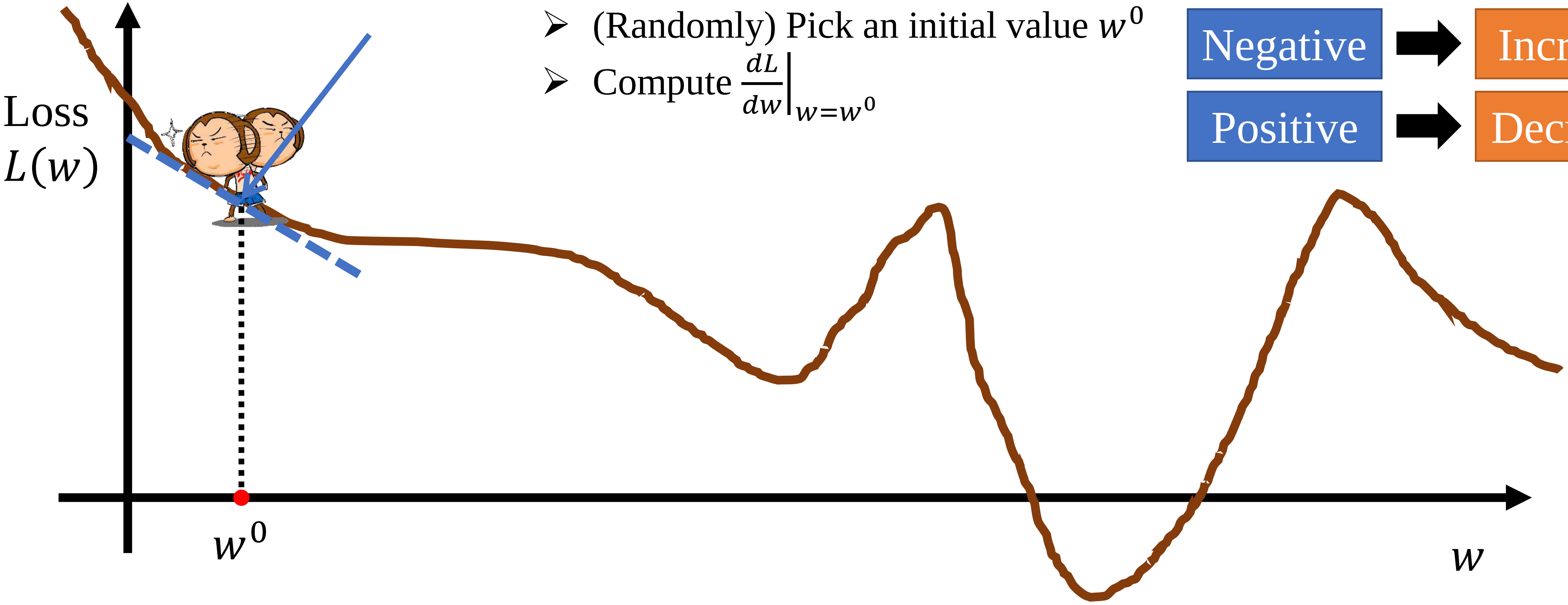
$$= \arg min_{w,b} \sum_{n=1}^{10} \left(\hat{y}^n - (b + w \cdot x_{cp}^n)\right)^2$$

Gradient Descent

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 $\left.\frac{dL}{dw}\right|_{w=w^0}$

Negative → Increase w

Positive → Decrease w

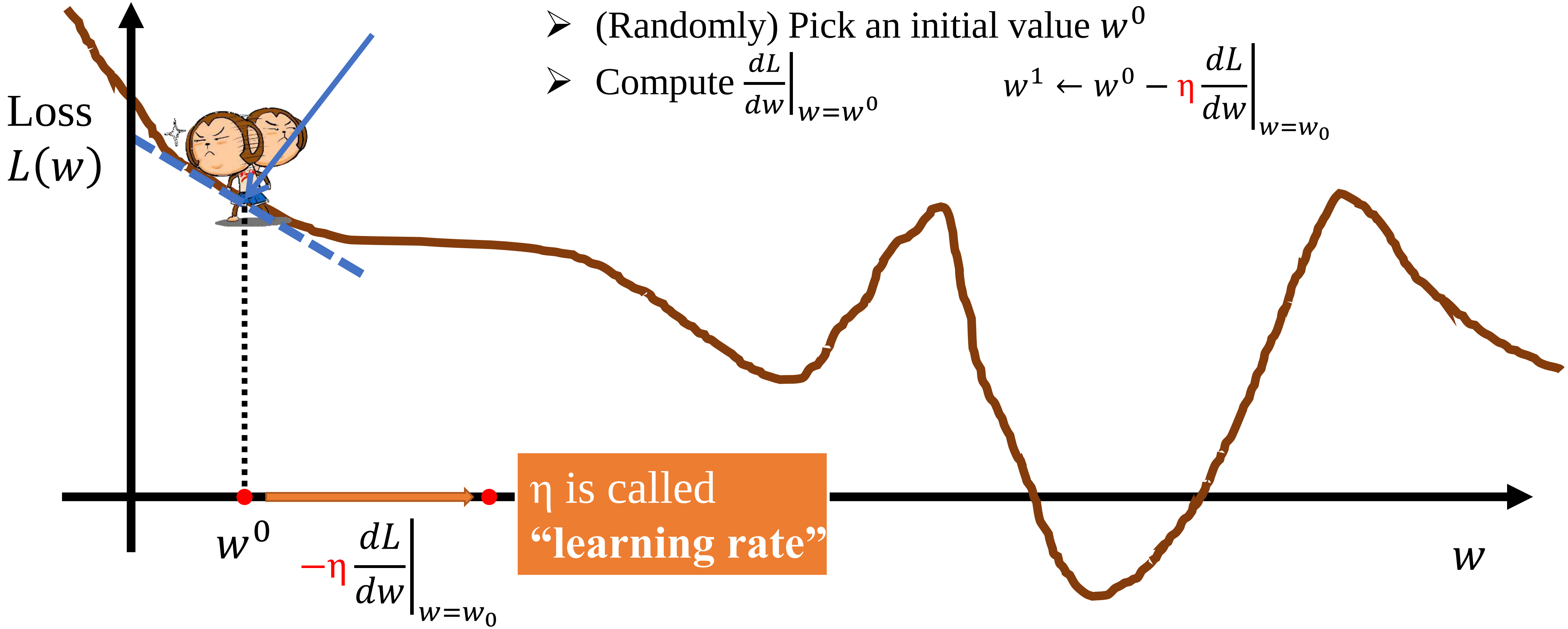
Step 3 : Gradient Descent

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

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

➢ (Randomly) Pick an initial value w^0

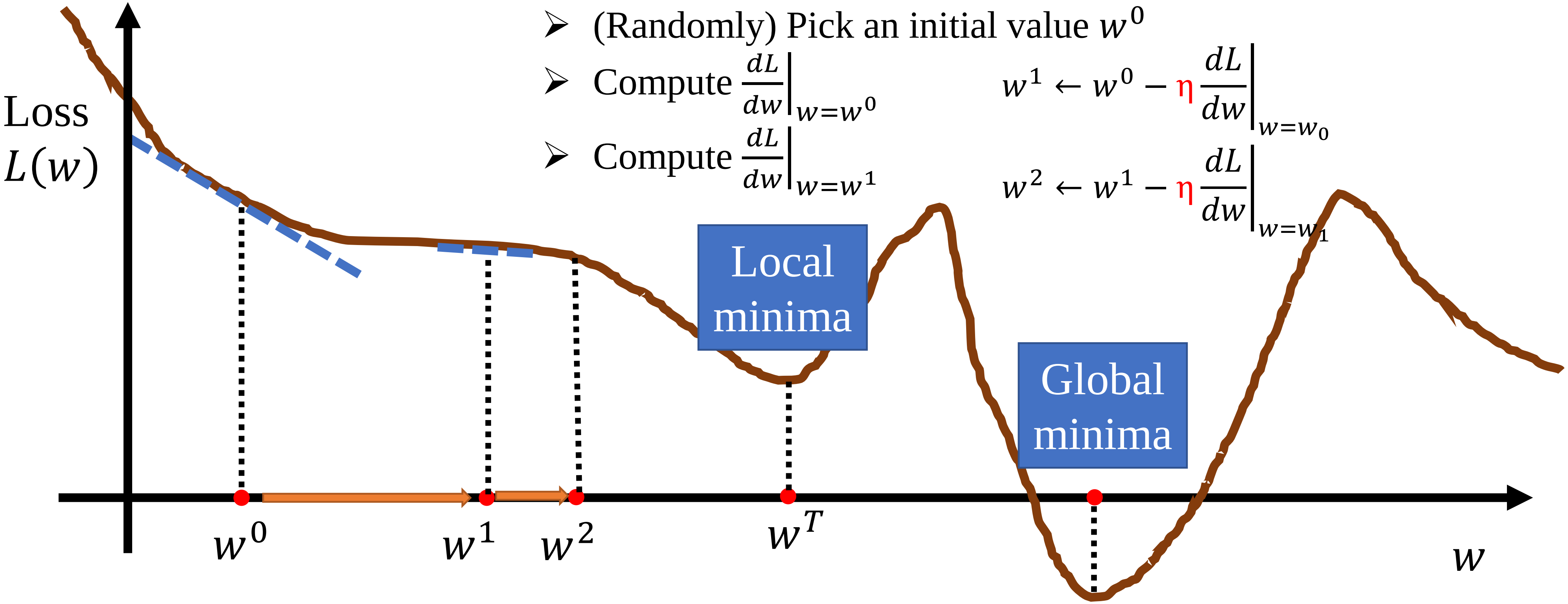
➢ Compute $\left.\frac{dL}{dw}\right|_{w=w^0}$ $\quad w^1 \leftarrow w^0 - \eta \left.\frac{dL}{dw}\right|_{w=w_0}$



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 $\left.\frac{dL}{dw}\right|_{w=w^0}$ $\quad w^1 \leftarrow w^0 - \eta \left.\frac{dL}{dw}\right|_{w=w_0}$
- Compute $\left.\frac{dL}{dw}\right|_{w=w^1}$ $\quad w^2 \leftarrow w^1 - \eta \left.\frac{dL}{dw}\right|_{w=w_1}$

Step 3 : Gradient Descent

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

- How about two parameters?

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

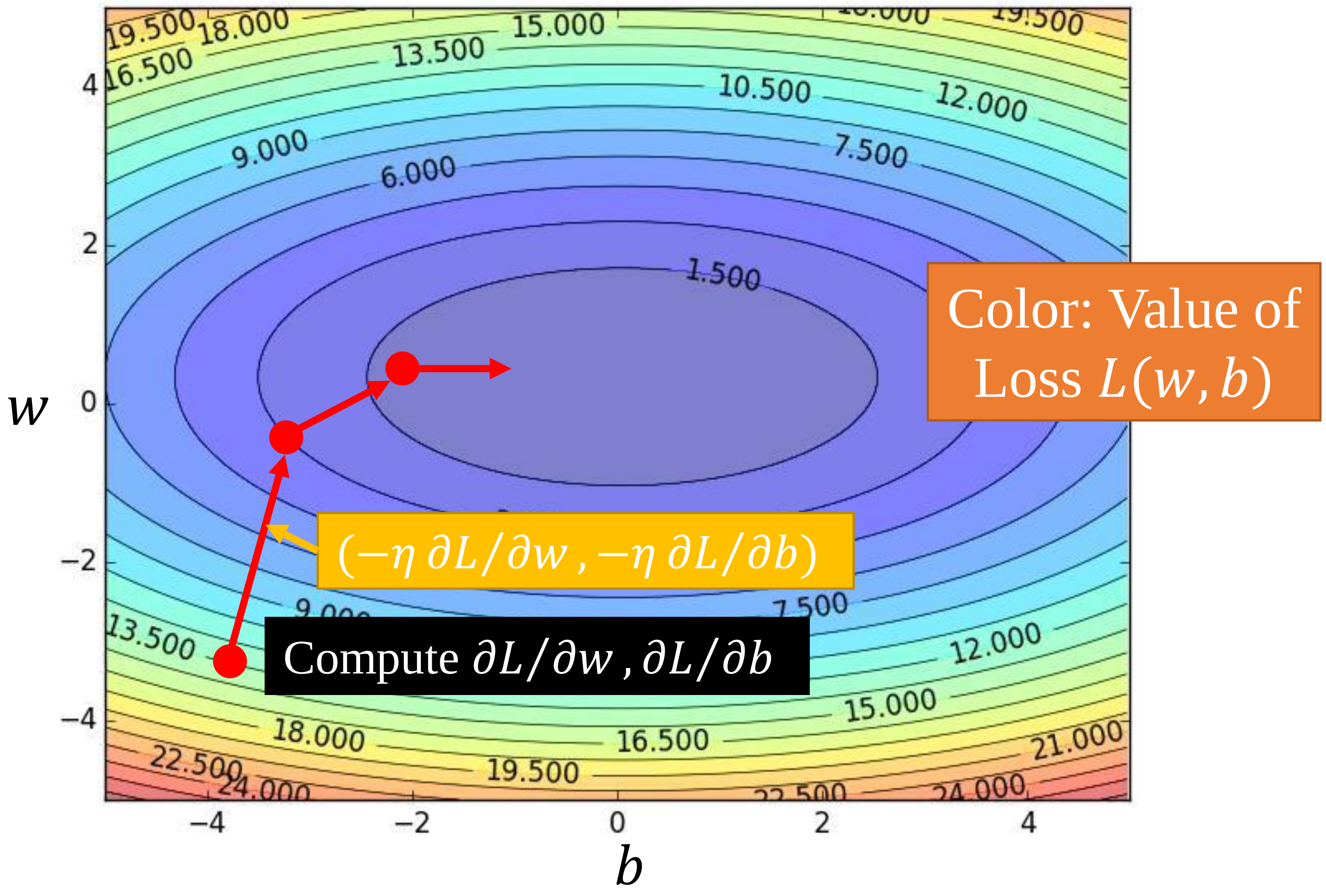
  - (Randomly) Pick an initial value w^0, b^0
  - Compute $\left.\frac{\partial L}{\partial w}\right|_{w=w^0,b=b^0}, \left.\frac{\partial L}{\partial b}\right|_{w=w^0,b=b^0}$

$$w^1 \leftarrow w^0 - \eta \left.\frac{\partial L}{\partial w}\right|_{w=w^0,b=b^0} \qquad b^1 \leftarrow b^0 - \eta \left.\frac{\partial L}{\partial b}\right|_{w=w^0,b=b^0}$$

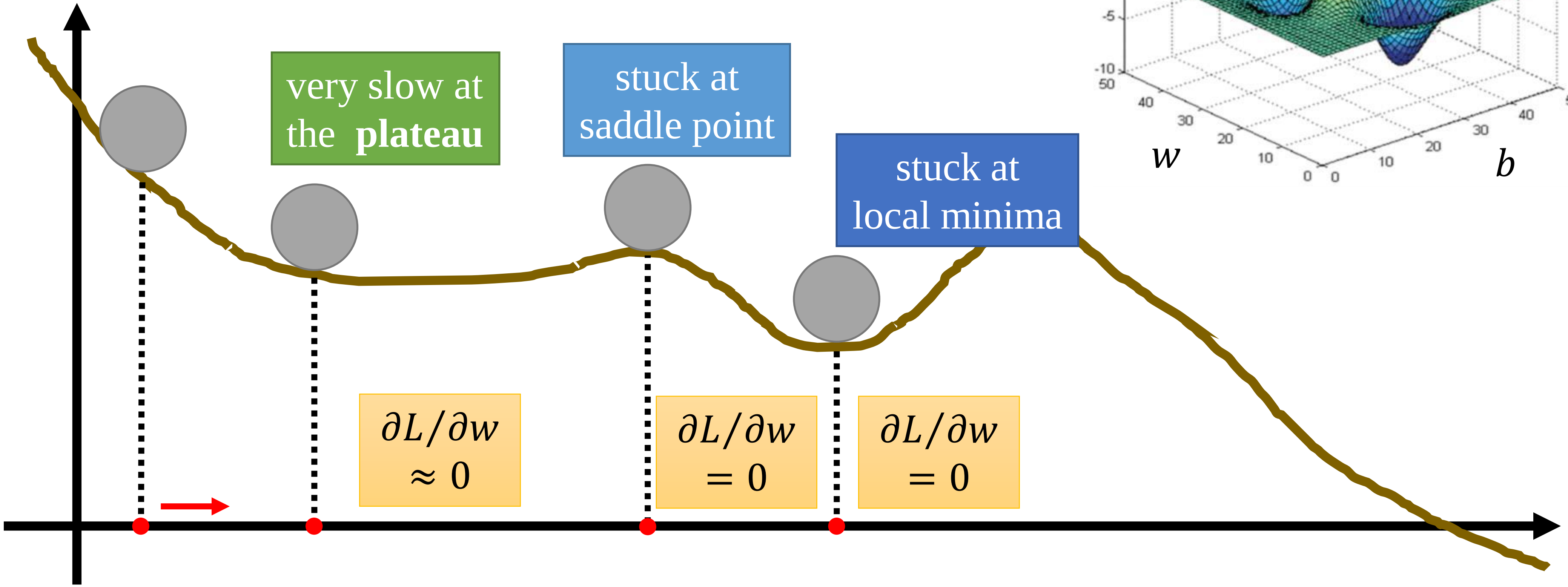
  - Compute $\left.\frac{\partial L}{\partial w}\right|_{w=w^1,b=b^1}, \left.\frac{\partial L}{\partial b}\right|_{w=w^1,b=b^1}$

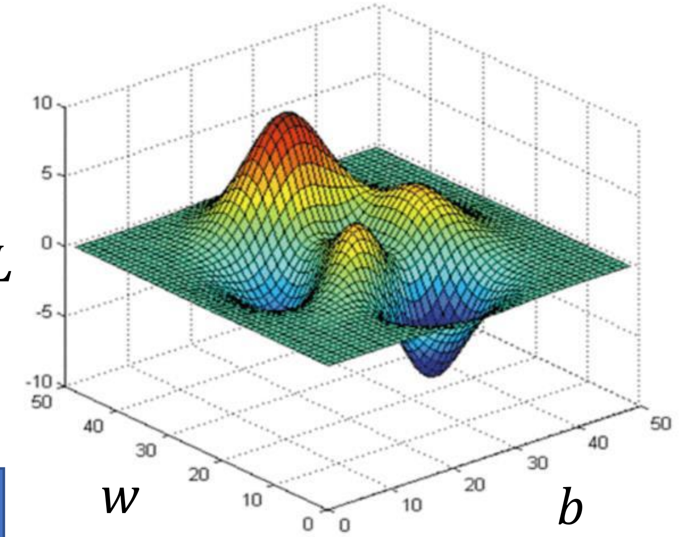
$$w^2 \leftarrow w^1 - \eta \left.\frac{\partial L}{\partial w}\right|_{w=w^1,b=b^1} \qquad b^2 \leftarrow b^1 - \eta \left.\frac{\partial L}{\partial b}\right|_{w=w^1,b=b^1}$$

Step 3 : Gradient Descent



Step 3 : Gradient Descent





Step 3 : Gradient Descent

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

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

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

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

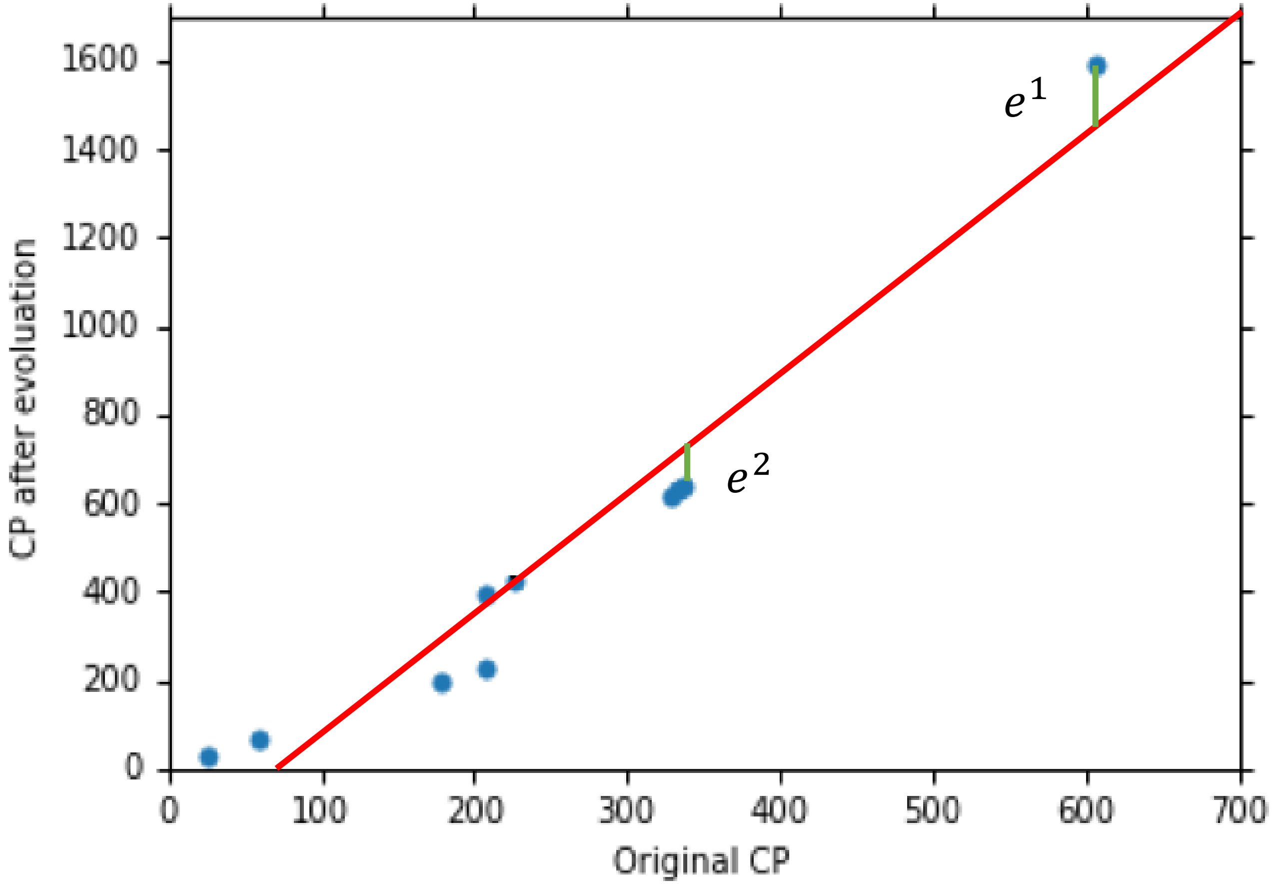
How's the results?

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

$b = -188.4$
$w = 2.7$

average error on training data

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



How's the results? - Generalization

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

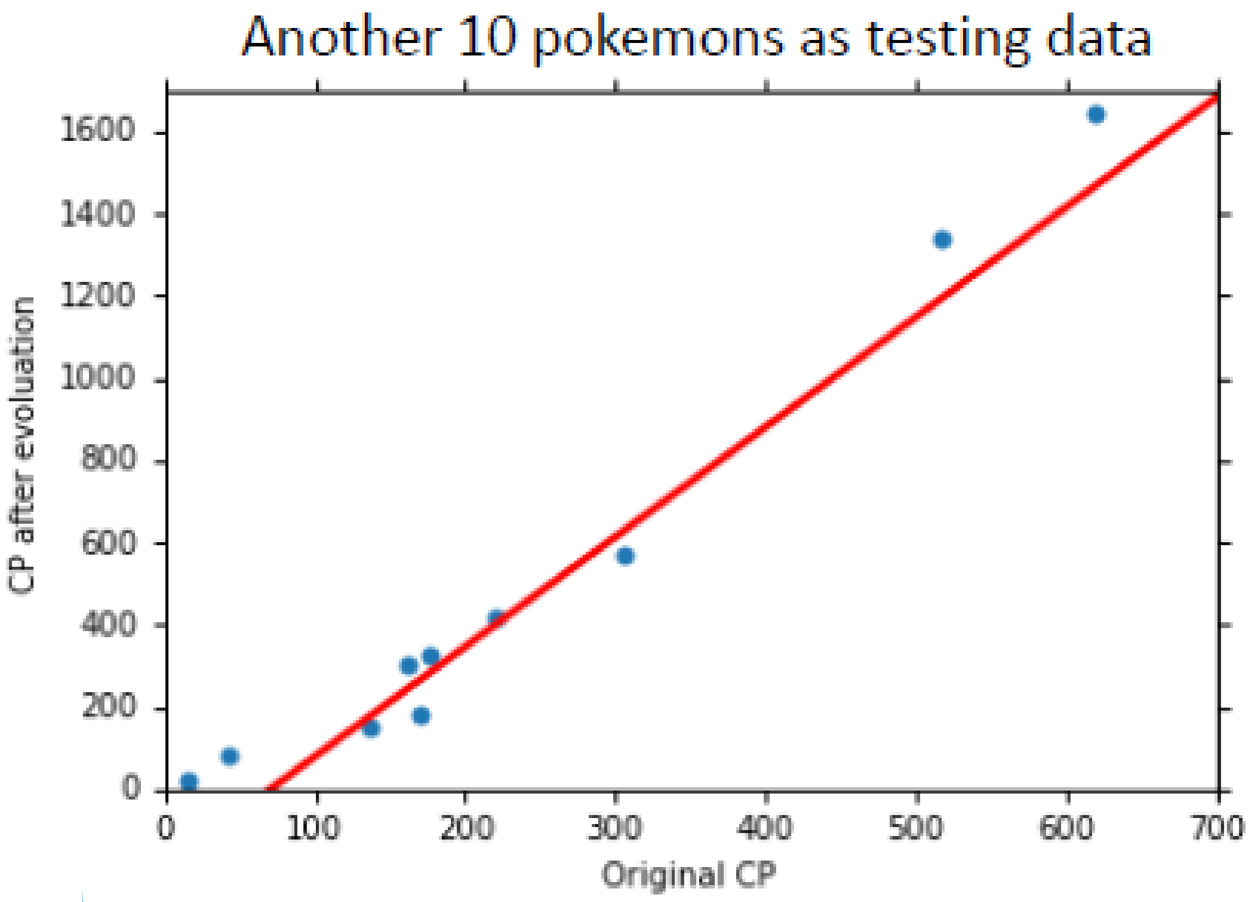
What we really care about is the error on new date (testing data)

$b = -188.4$
$w = 2.7$

average error on training data

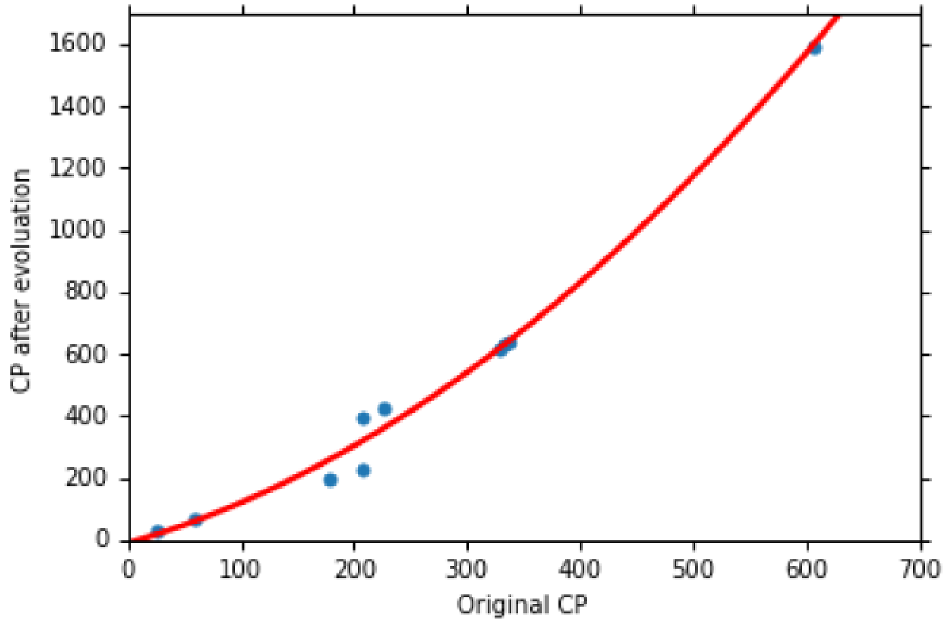
$$= \sum_{n=1}^{10} e^n = 35.0$$

> average error on training date (31.9)



Selecting another model

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

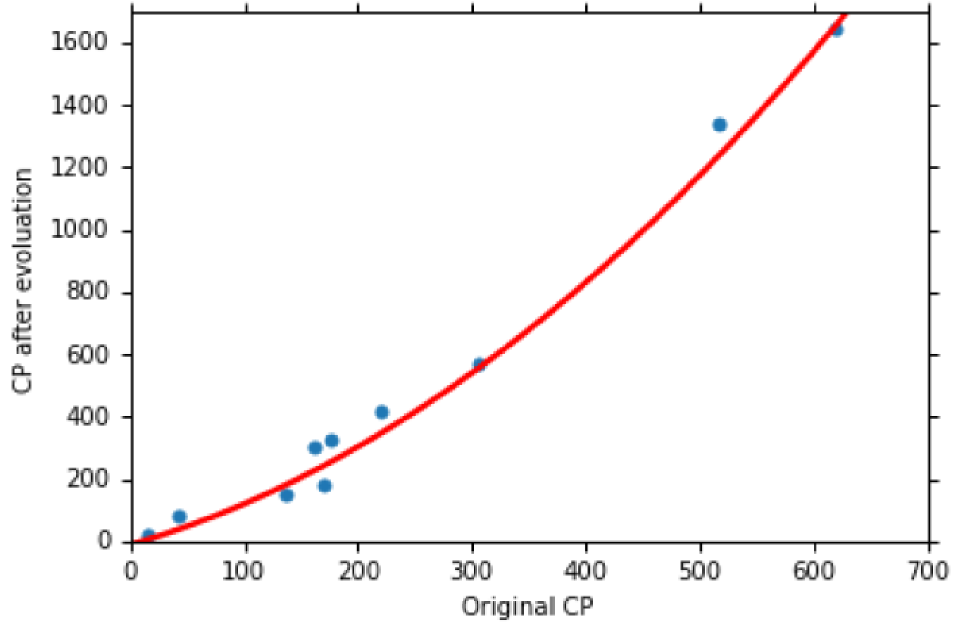


best function:

$b = -10.3$

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

average error = 15.4



testing:

average error = 18.4

better! could it be even better?

Selecting another model

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

best function:

$b = 6.4$

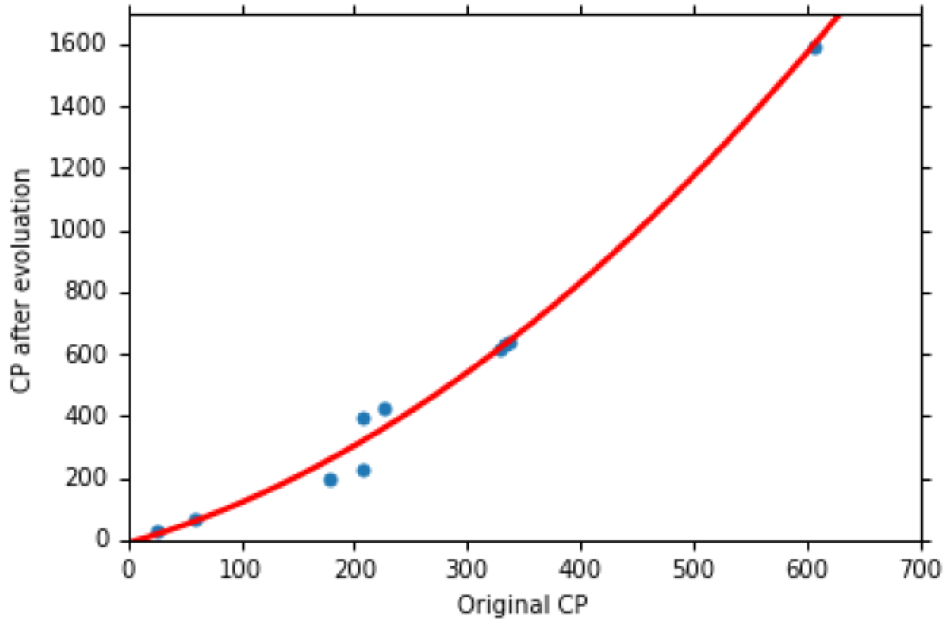
$w_1 = 0.66,\ w_2 = 4.3 \times 10^{-3},$
$w_3 = -1.8 \times 10^{-6}$

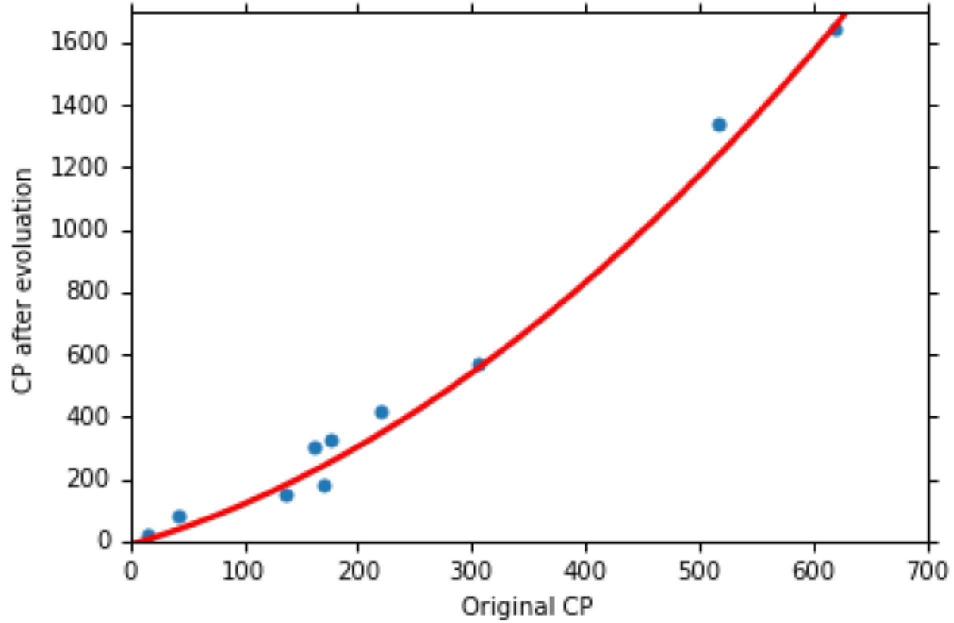
average error = 15.3

testing:

average error = 18.1

slightly better!
how about more complex model?





Selecting another model

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

best function:

average error = 14.9

testing:

average error = 28.8

the result become worse …

Selecting another model

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

best function:

average error = 12.8

testing:

average error = 232.1

the results are so bad.

Model Selection

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

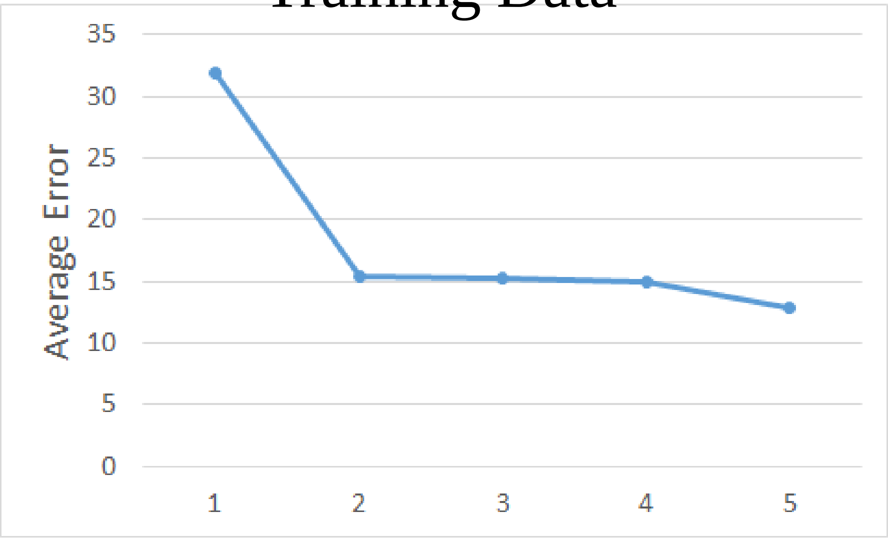
2. $y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2$

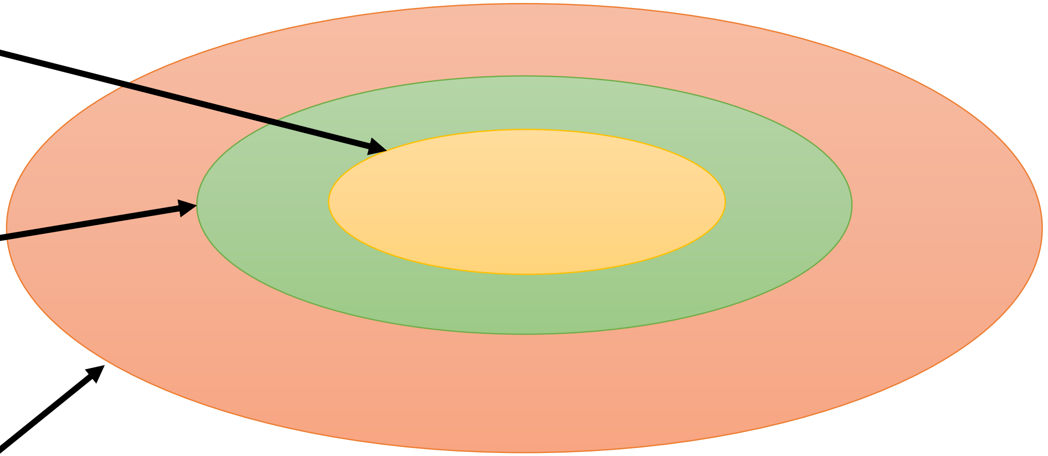
3. $y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2 + w_3 \cdot (x_{cp})^3$

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

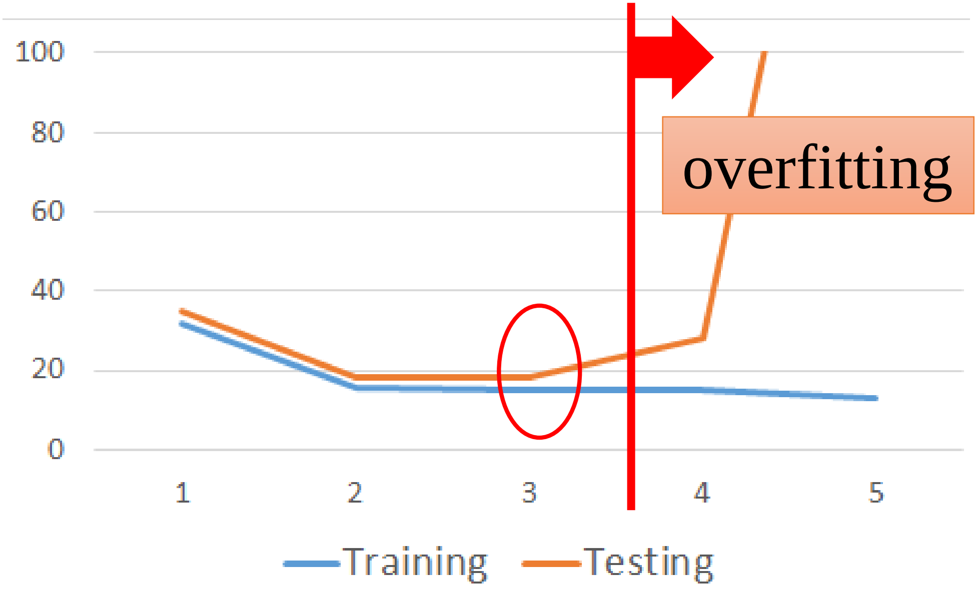




a more complex model yields lower error on training data.

if we can truly find the best function

Model Selection

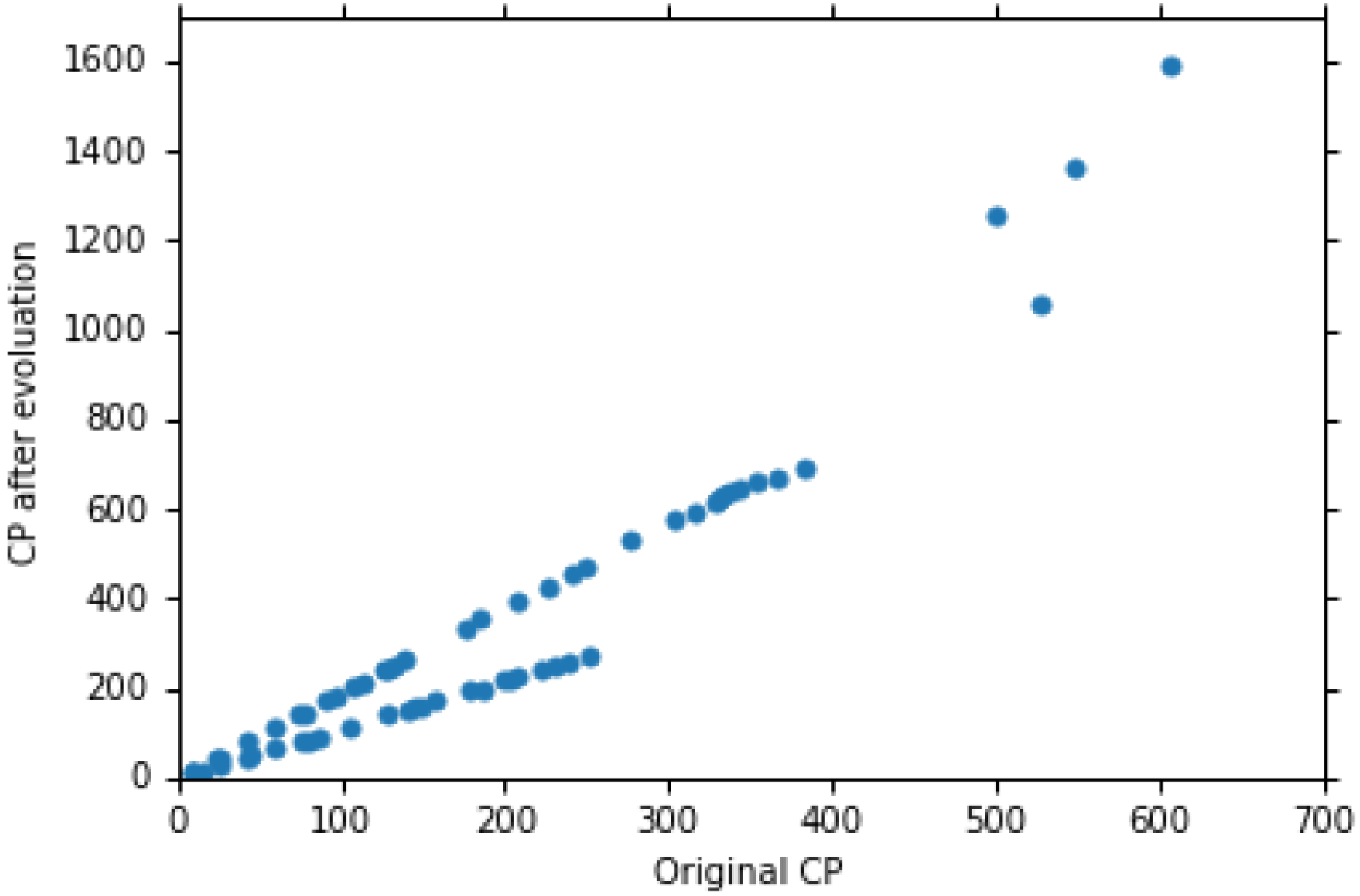


	Training	Testing
1	31.9	35.0
2	15.4	18.4
3	15.3	18.1
4	14.9	28.2
5	12.8	232.1

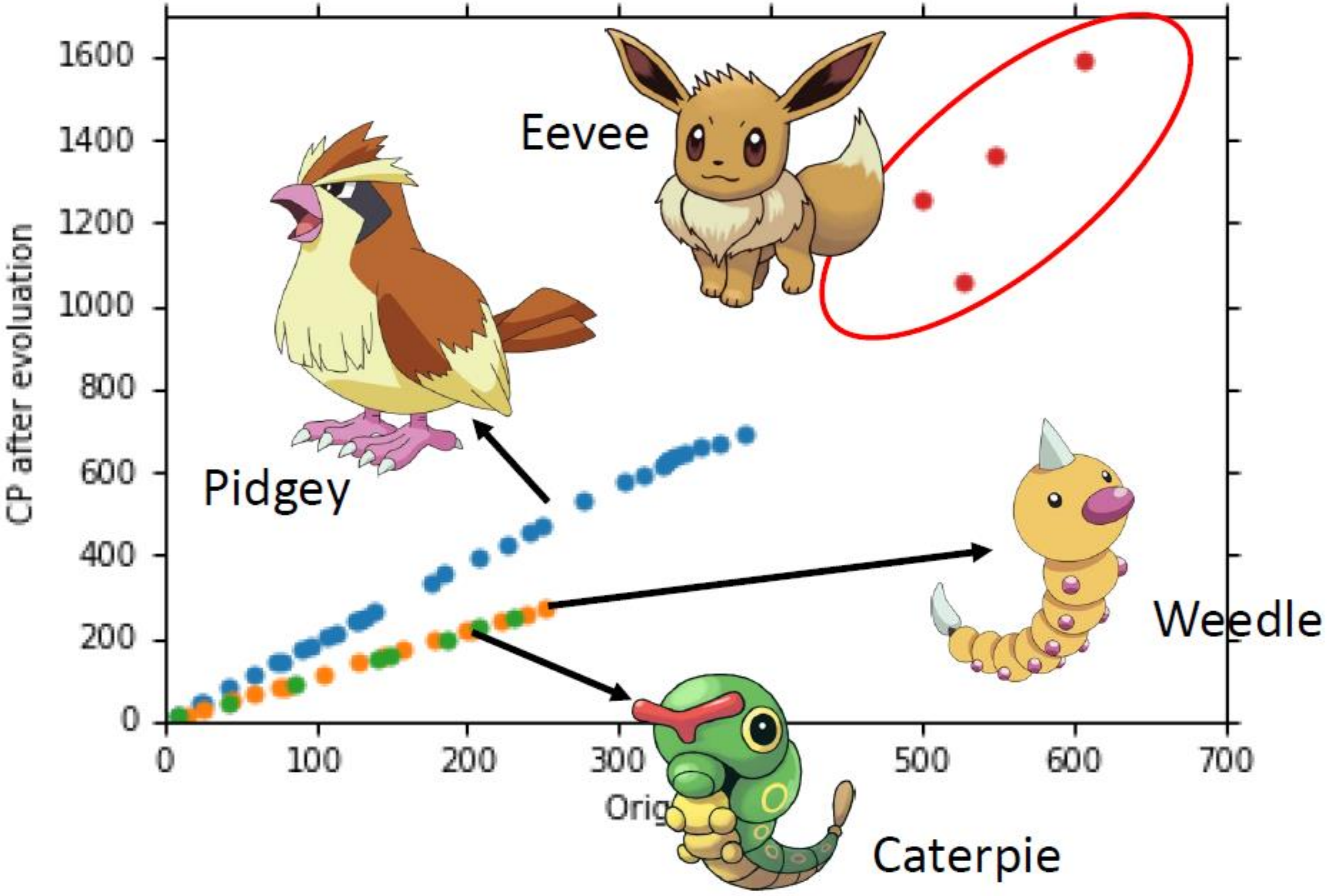
a more complex model does not always lead to better performance on **testing data.**

this is ***overfitting***. → select suitable model

Let's collect more data



What are the hidden factors?



Bach 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

Bach to step 1: redesign the model

$$y = b + \sum w_i \boxed{x_i}$$

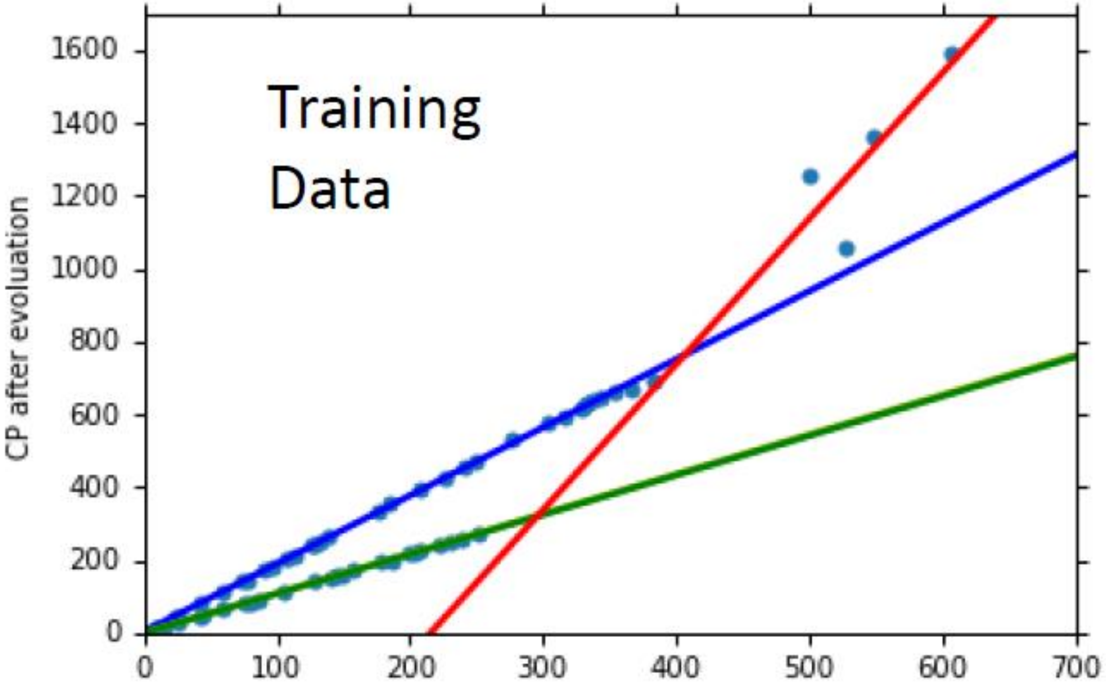
linear model?

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

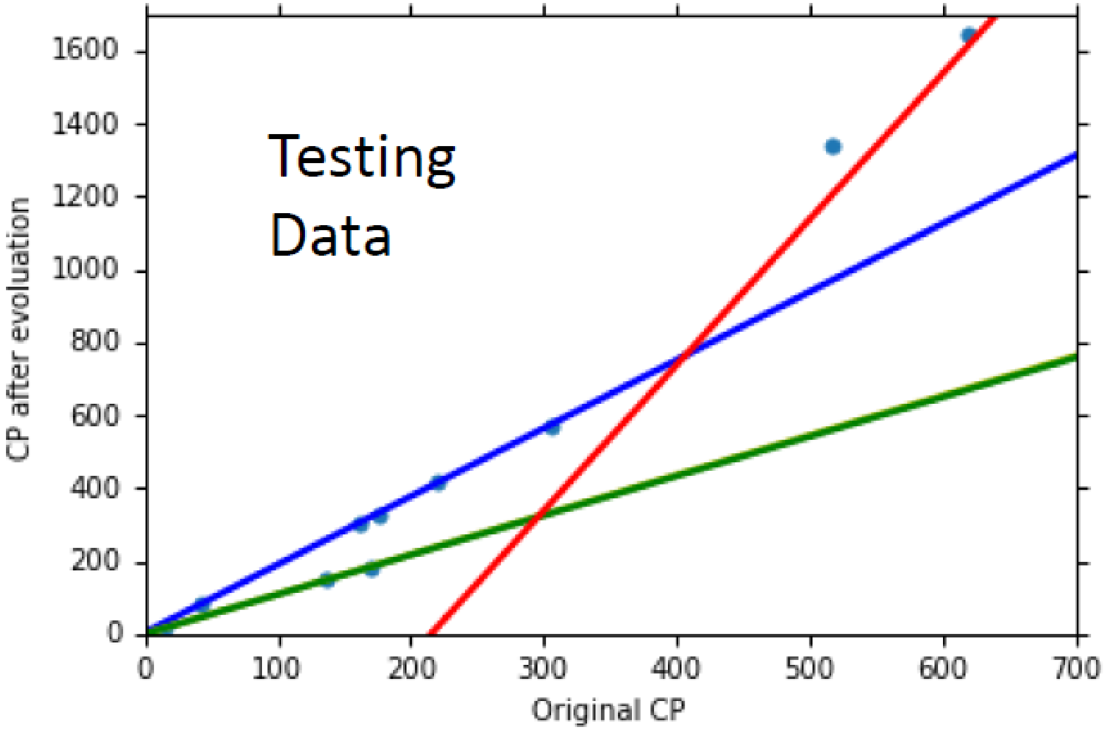
$$\delta(x_s = Pidgey)$$

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

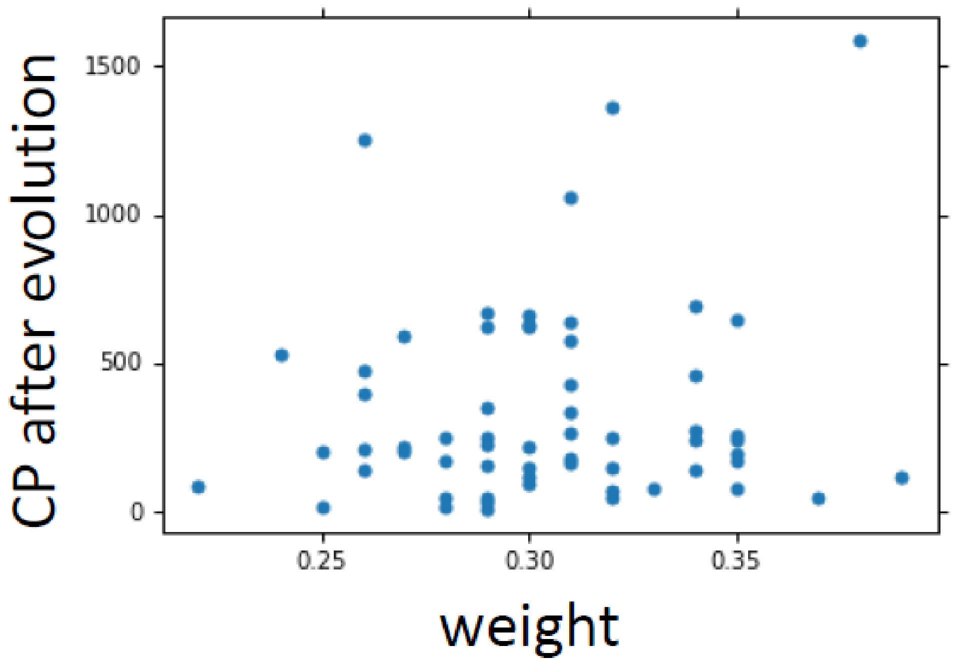
average error = 3.8

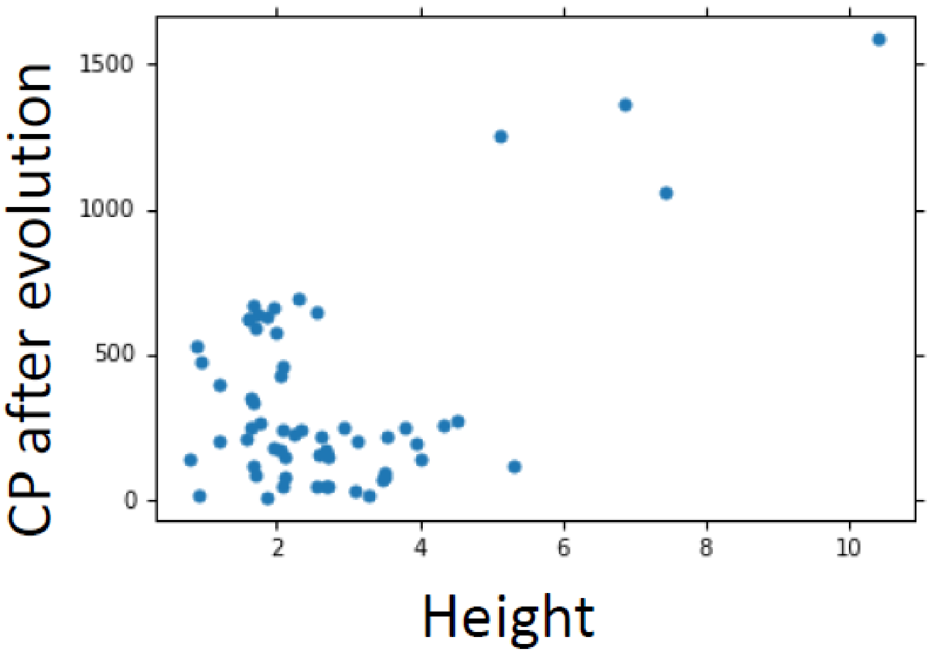


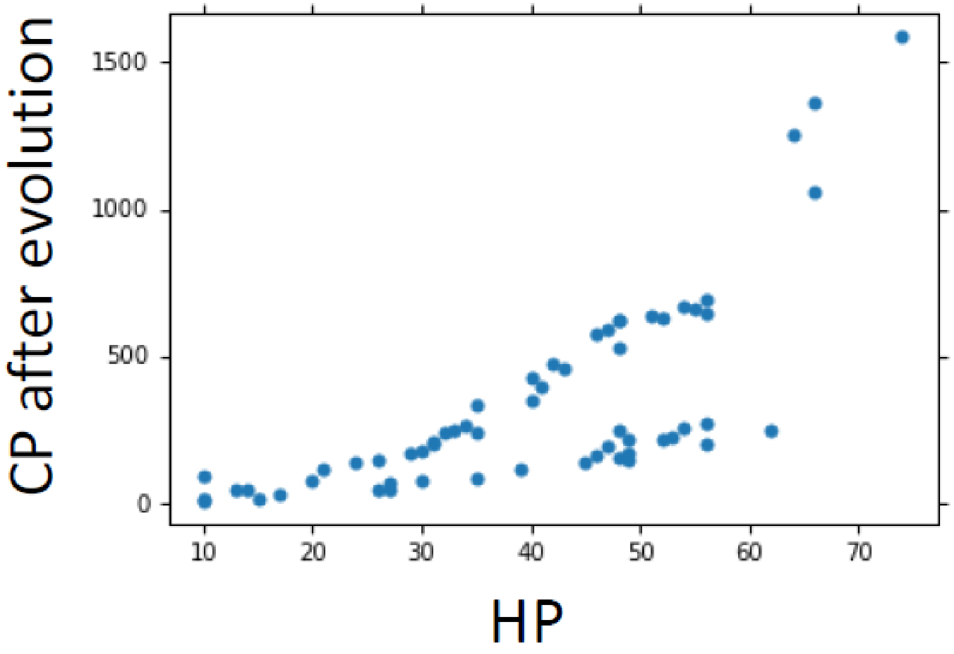
average error = 14.3



are there any other hidden factors?







Bach to step 1: redesign the model again

x

$$
\begin{aligned}
&if\ x_s = Pidgey: && y' = b_1 + w_1 \cdot x_{cp} + w_5 \cdot (x_{cp})^2 \\
&if\ x_s = Weedle: && y' = b_2 + w_2 \cdot x_{cp} + w_6 \cdot (x_{cp})^2 \\
&if\ x_s = Caterpie: && y' = b_3 + w_3 \cdot x_{cp} + w_7 \cdot (x_{cp})^2 \\
&if\ x_s = Eevee: && y' = b_4 + w_4 \cdot x_{cp} + w_8 \cdot (x_{cp})^2
\end{aligned}
$$

$$
\begin{aligned}
&y = y' + w_9 \cdot x_{hp} + w_{10} \cdot \left(x_{hp}\right)^2 \\
&+w_{11} \cdot x_h + w_{12} \cdot (x_h)^2 + w_{13} \cdot x_w + w_{14} \cdot (x_w)^2
\end{aligned}
$$

y

Training error =1.9

Testing error =102.3

overfitting!

Back to step 2: Regularization

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

The functions with smaller w_i are better

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

- Smaller w_i means… smoother

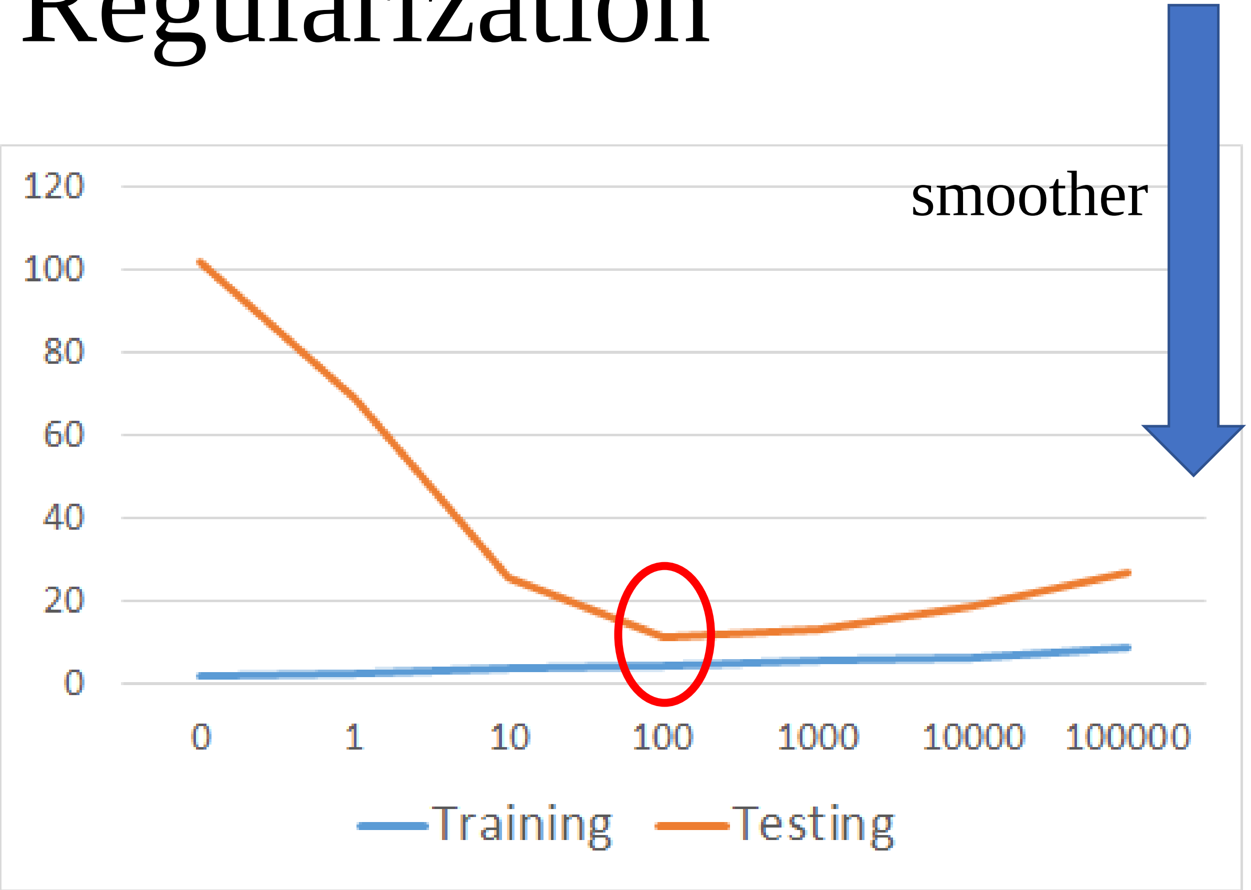
$$y = b + \sum w_i x_i$$

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

- 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.