

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

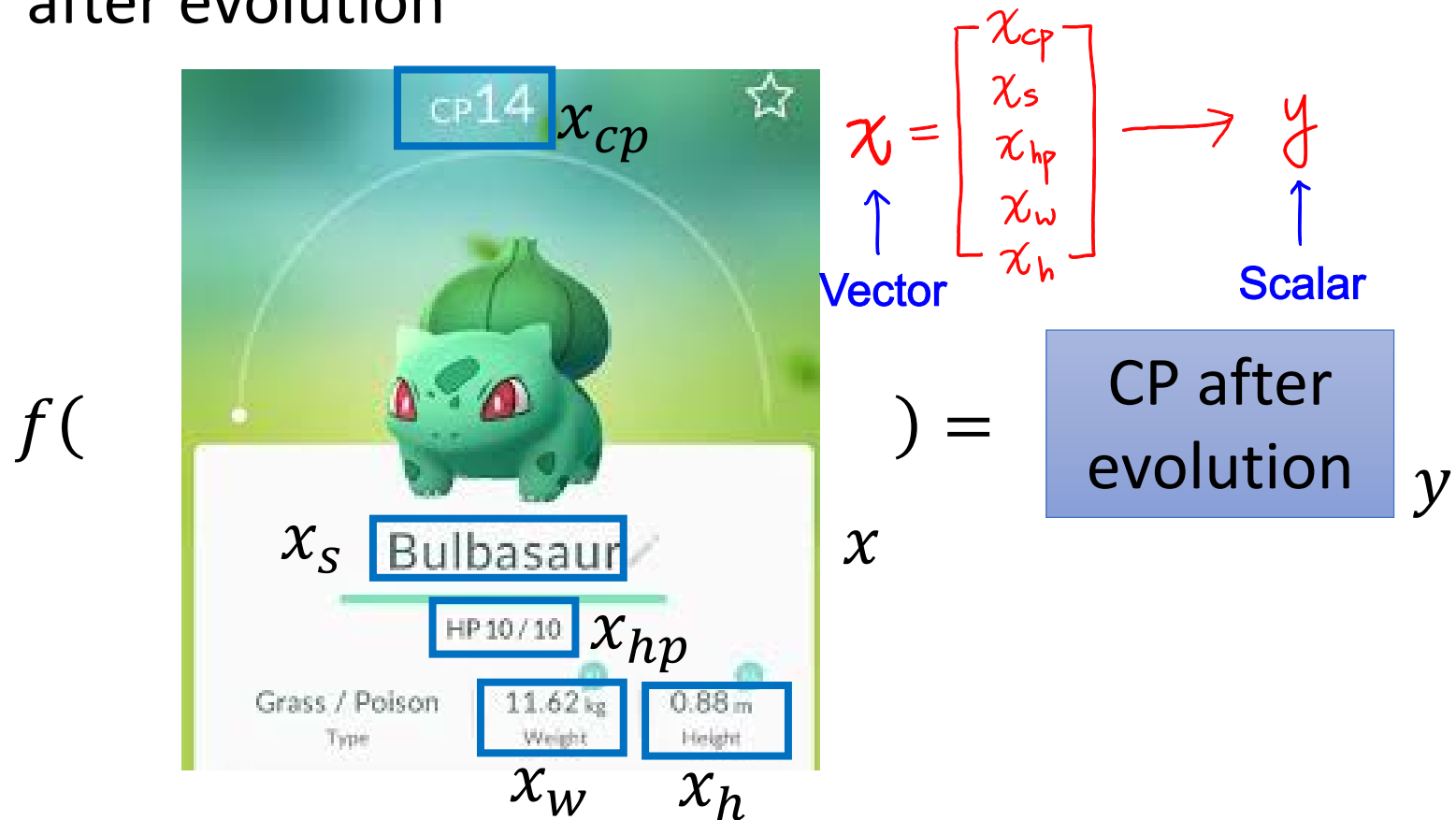
$$f(\text{Image of a self-driving car on a road}) = \text{方向盤角度}$$

- 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

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

A set of
function

Model

$f_1, f_2 \dots$

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

$f($



$x) =$

CP after
evolution

y

ex:

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

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

A set of
function

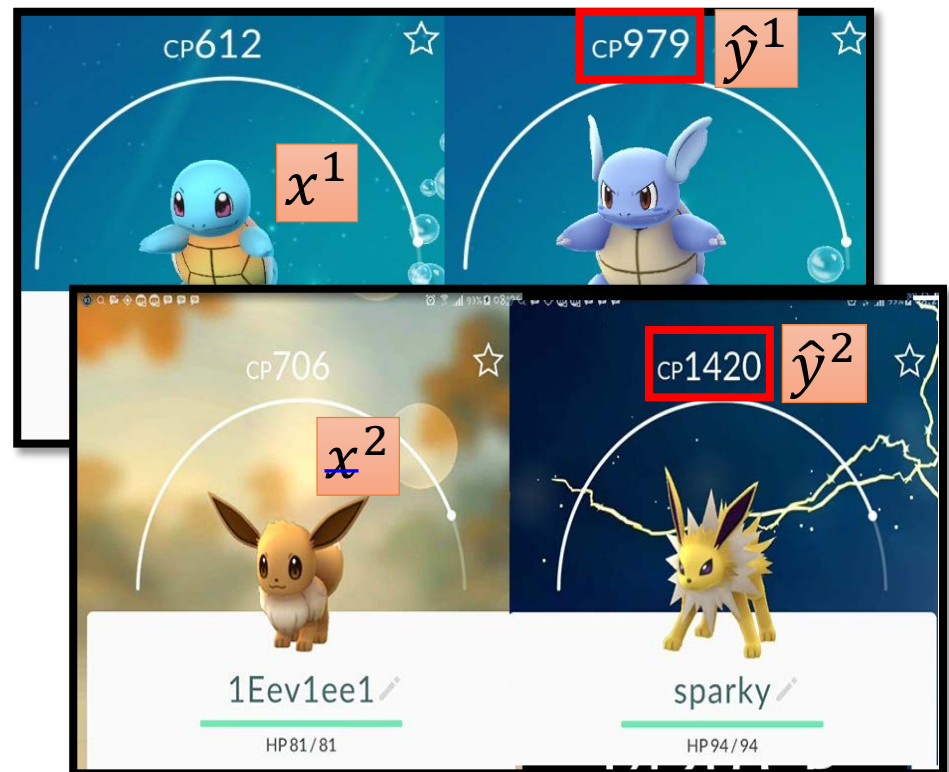
Model

$f_1, f_2 \dots$

Training
Data

function
input:

function
Output (scalar):



Step 2: Goodness of Function

Training Data:
10 pokemons

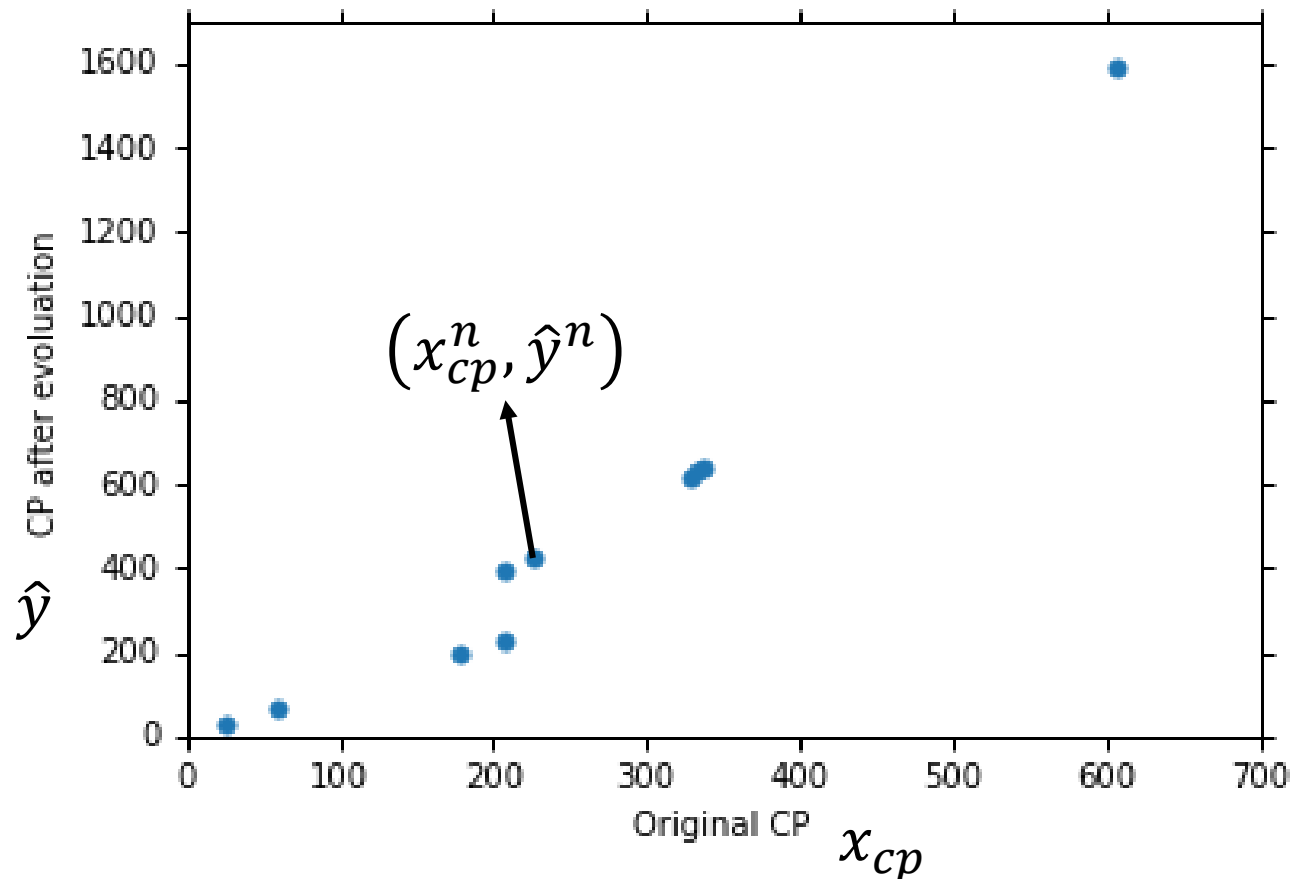
$$(x^1, \hat{y}^1)$$

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

⋮

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

This is real data.



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

The opposite of the loss function is the objective function.

Step 2: Goodness of Function

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

A set of
function

Model

$f_1, f_2 \dots$

Goodness of
function f

Training
Data

(The output is the bad level.)

It's a function that evaluate the performance.

Loss function L :

Input: a function, output:
how bad it is

Ground truth

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

Sum over examples

Estimated y based
on input function

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

The heatmap of the loss function.

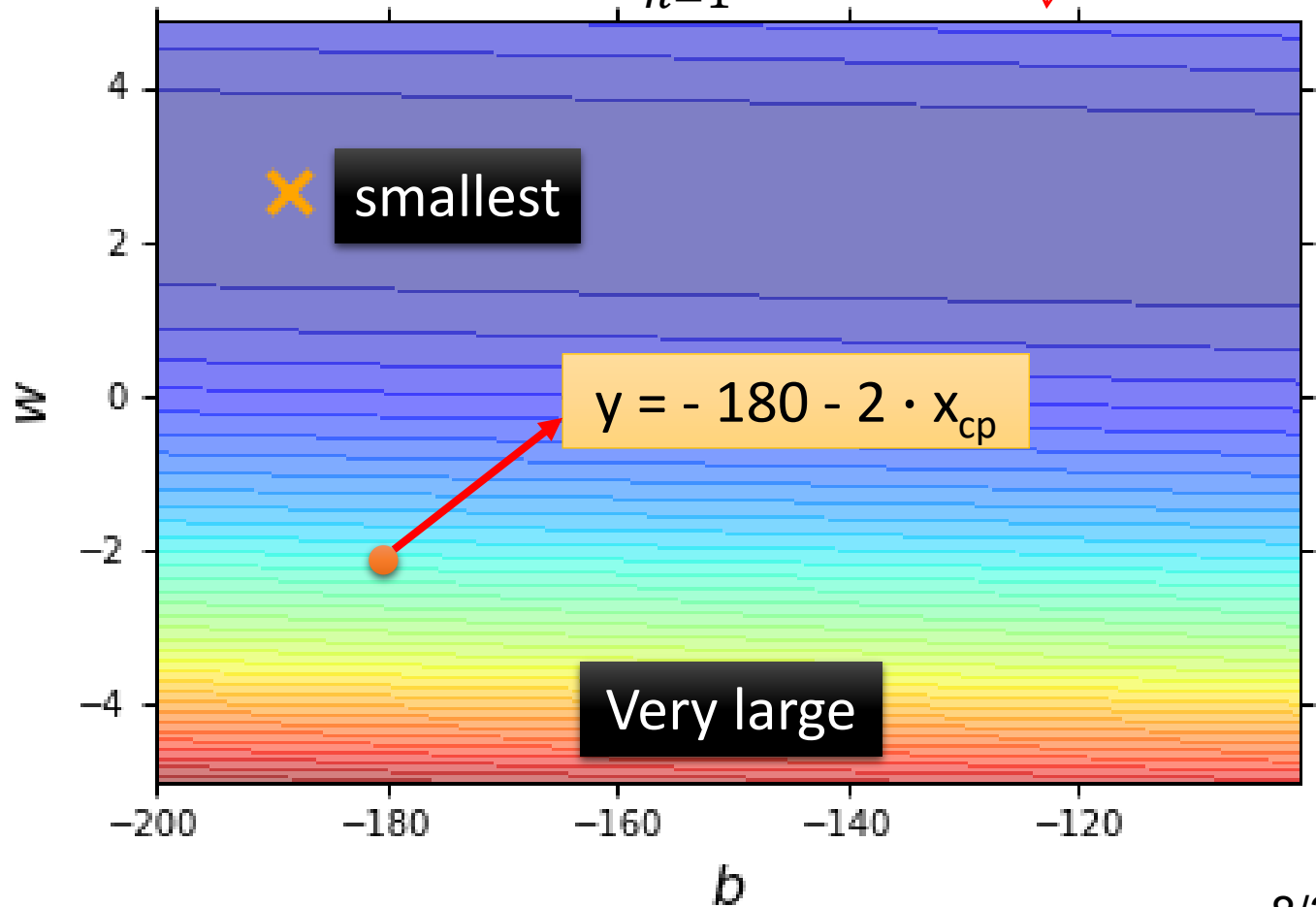
Step 2: Goodness of Function

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

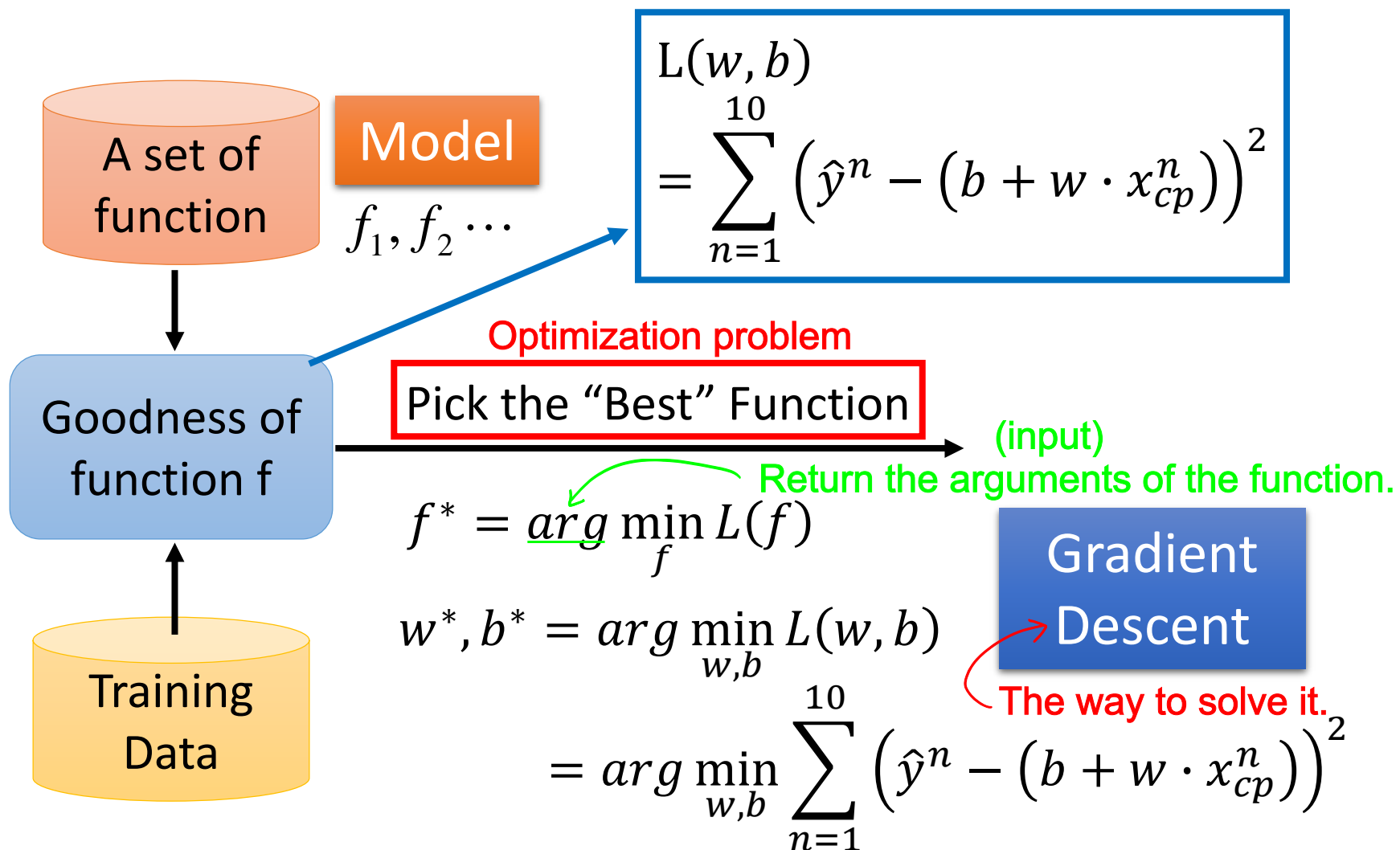
- Loss Function

Each point in
the figure is a
function
{parameter set

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



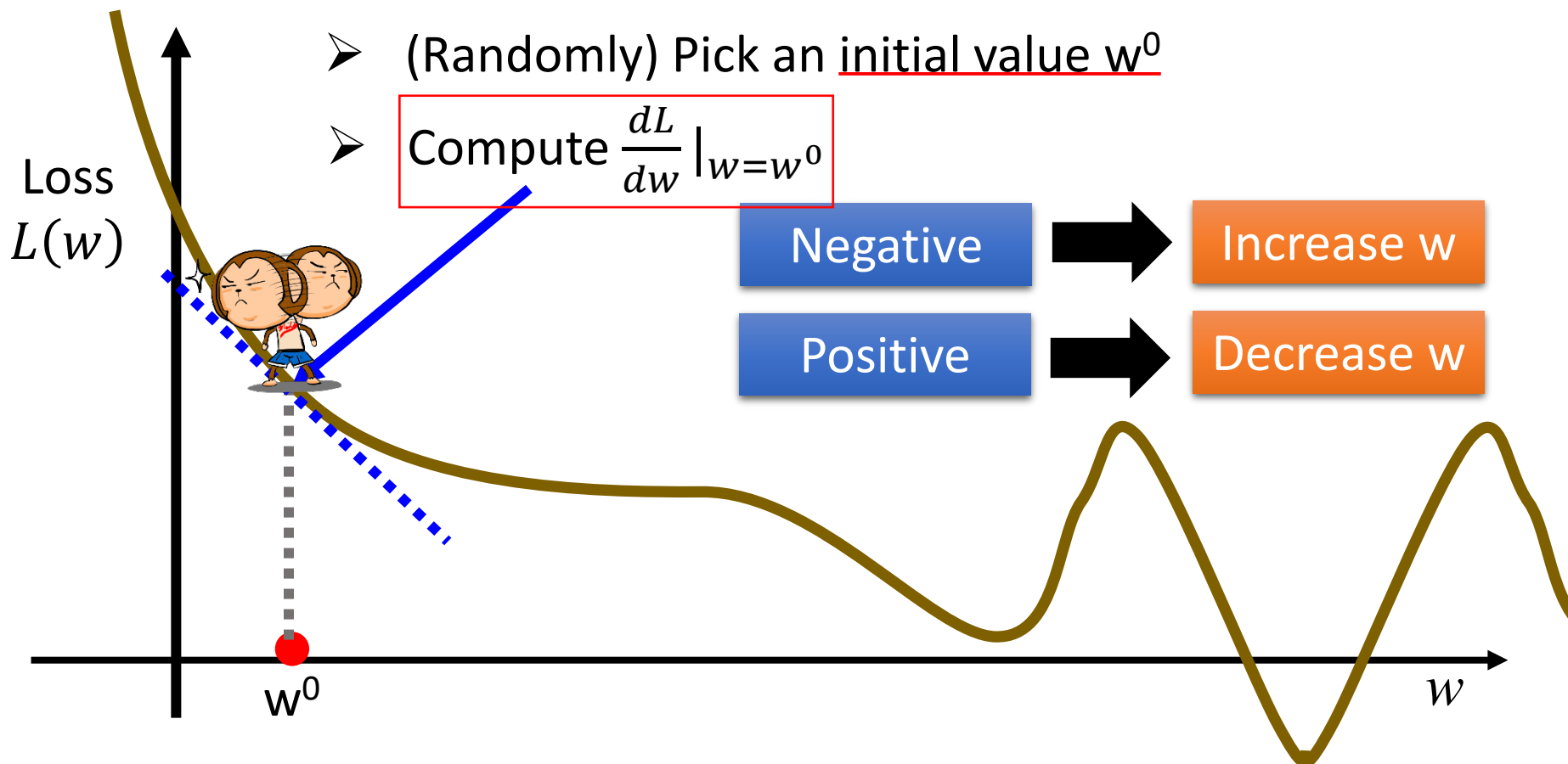
Step 3: Best Function



Step 3: Gradient Descent

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

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



Step 3: Gradient Descent

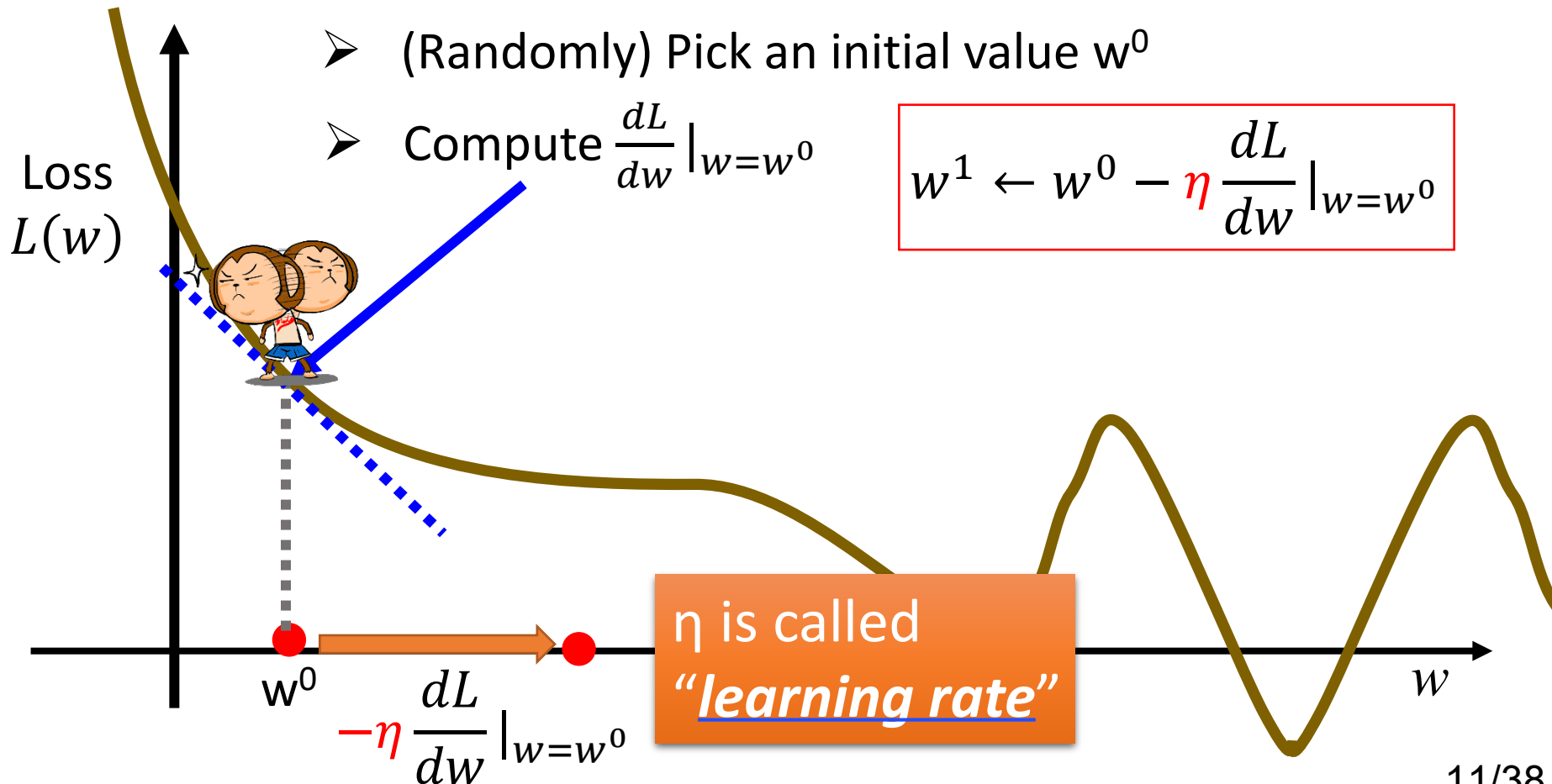
$$w^* = \arg \min_w L(w)$$

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

➤ (Randomly) Pick an initial value w^0

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

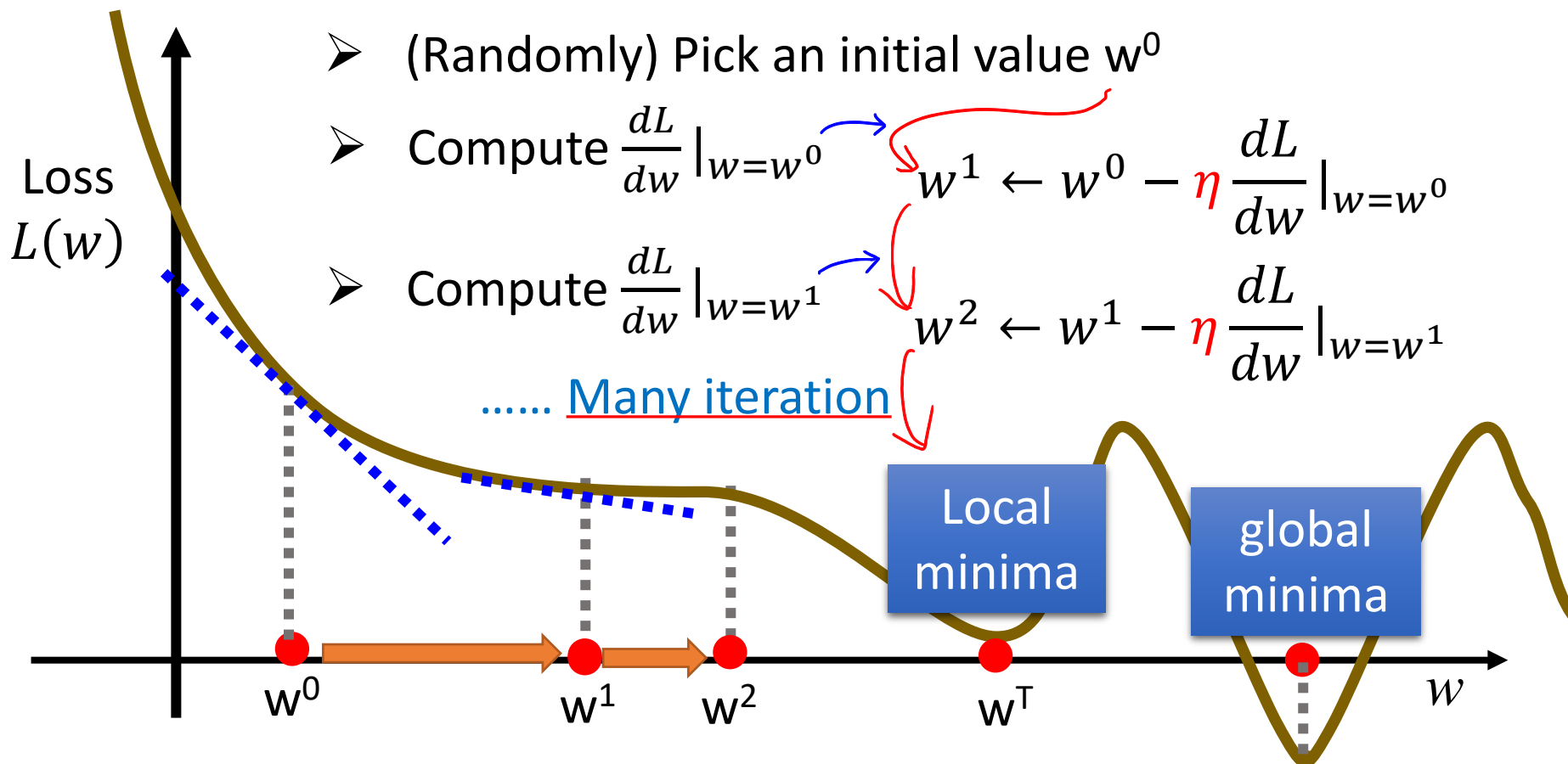
$$w^1 \leftarrow w^0 - \eta \frac{dL}{dw} \big|_{w=w^0}$$



Step 3: Gradient Descent

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

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



It's possible to stuck at local minima instead of global minima.

Step 3: Gradient Descent

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

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

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

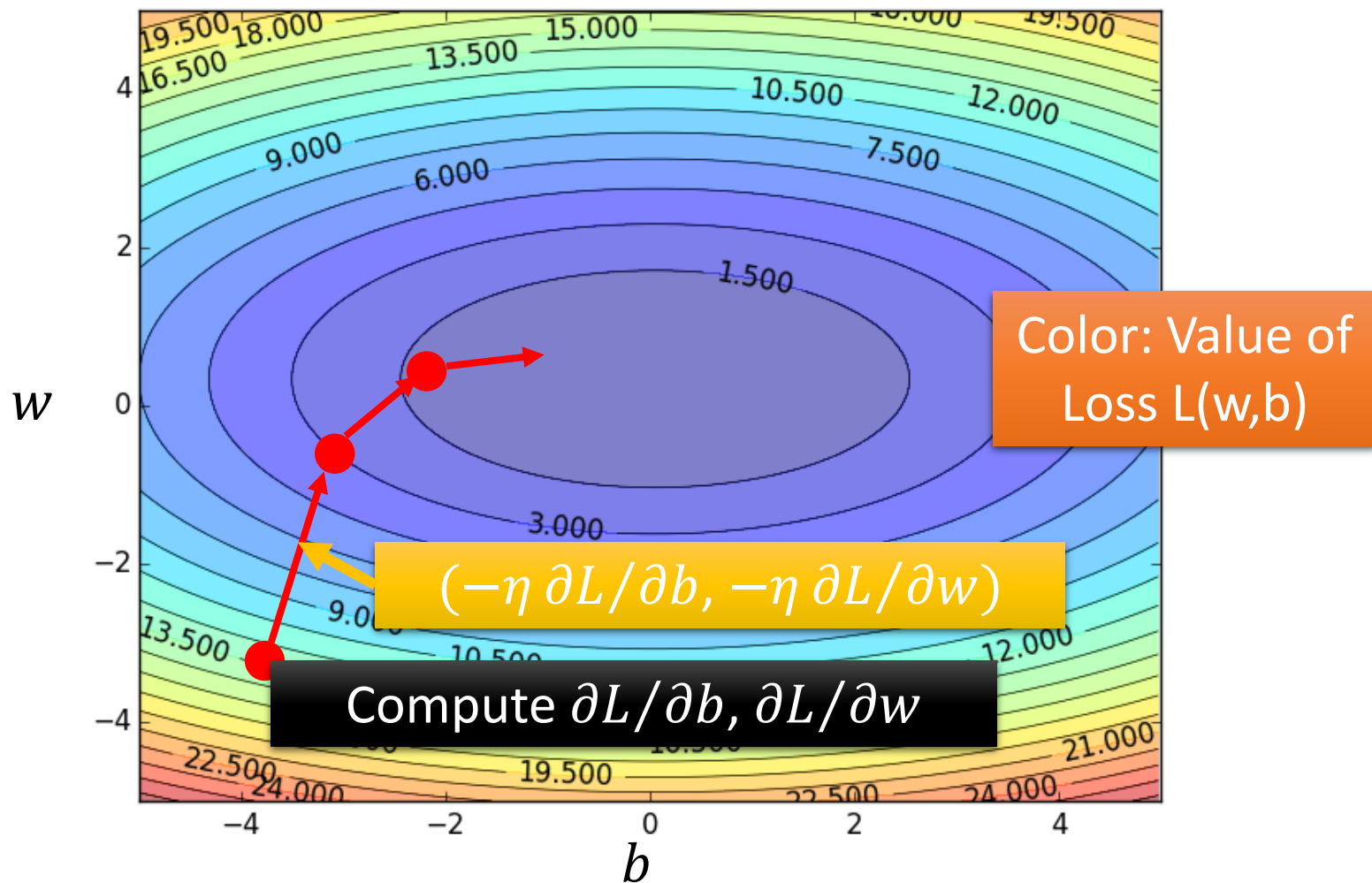
➤ Compute $\frac{\partial L}{\partial w} \Big|_{w=w^0, b=b^0}, \frac{\partial L}{\partial b} \Big|_{w=w^0, b=b^0}$

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

➤ 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

- When solving:

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

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

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

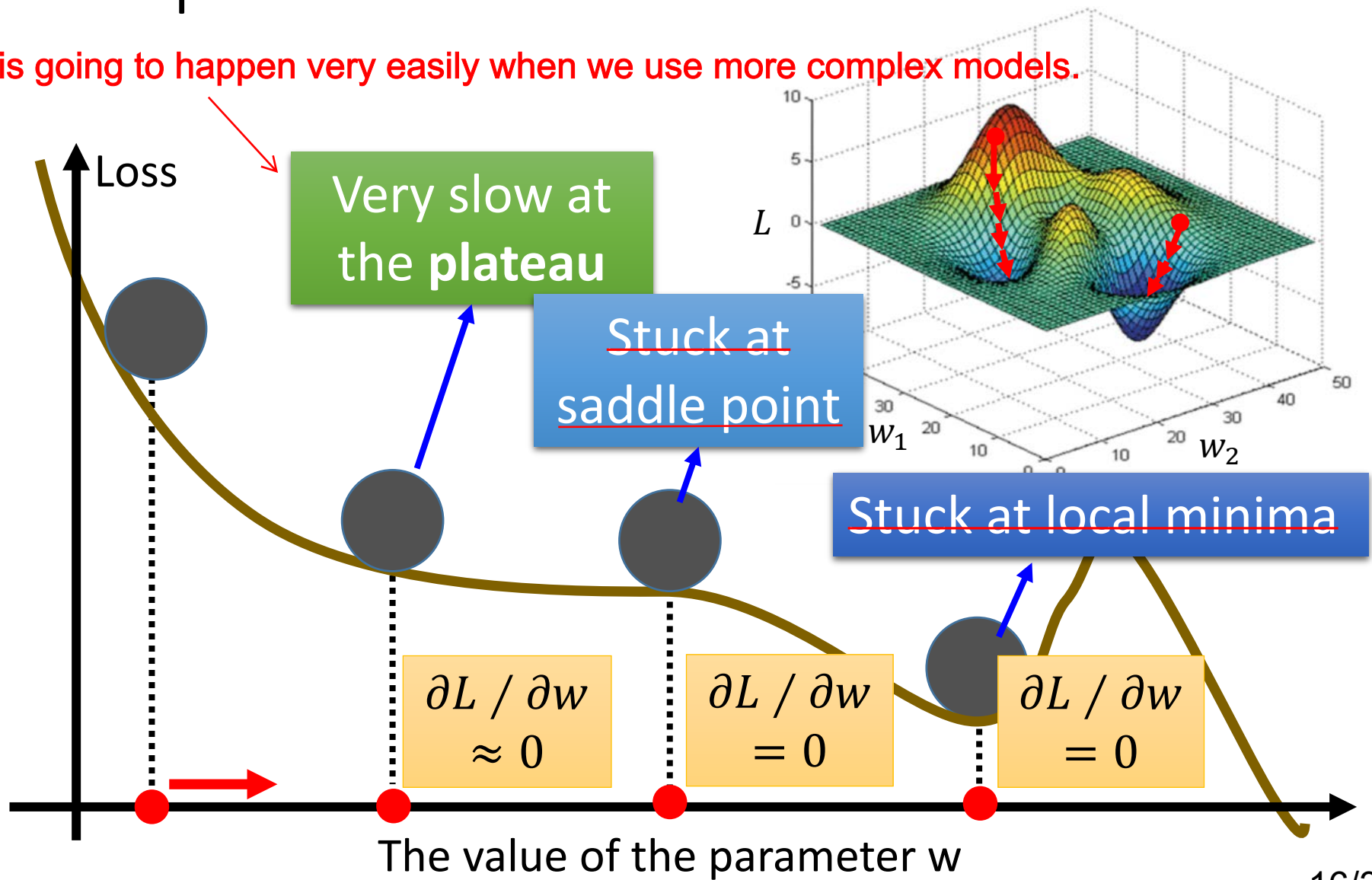
Our expectation.

Is this statement correct? **No**

\because Learning rate might be too large.


Step 3: Gradient Descent

It is going to happen very easily when we use more complex models.



Step 3: Gradient Descent

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

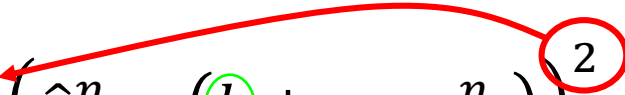
$$L(w, b) = \sum_{n=1}^{10} \left(\hat{y}^n - (b + \underbrace{w \cdot x_{cp}^n}_{\text{blue underline}}) \right)^2$$


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

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

Step 3: Gradient Descent

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

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


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

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

How's the results?

Training

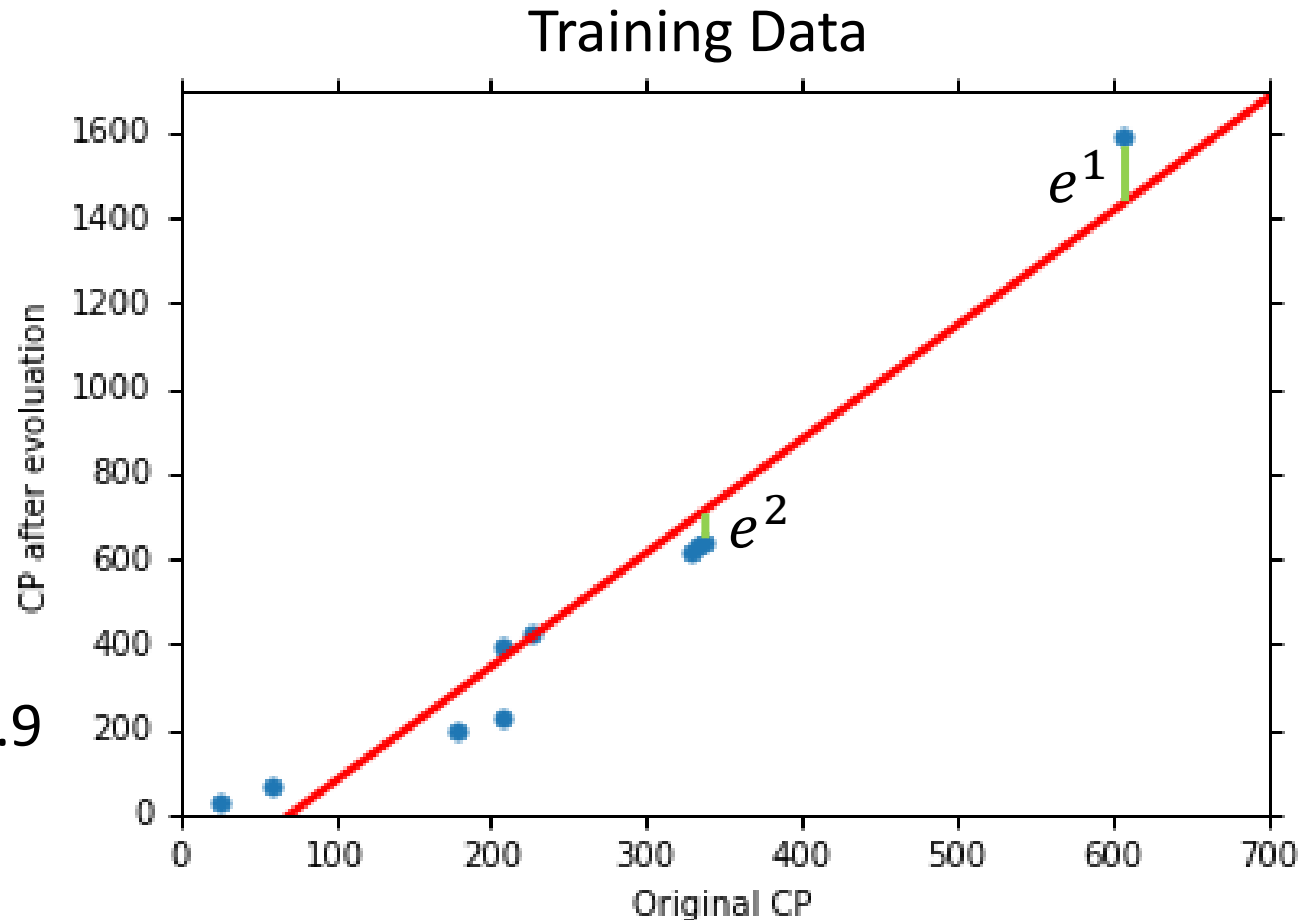
$$y = b + w \cdot x_{cp}$$

$$b = -188.4$$

$$w = 2.7$$

Average Error on
Training Data

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



How's the results?

- Generalization

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

Testing

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

$$b = -188.4$$

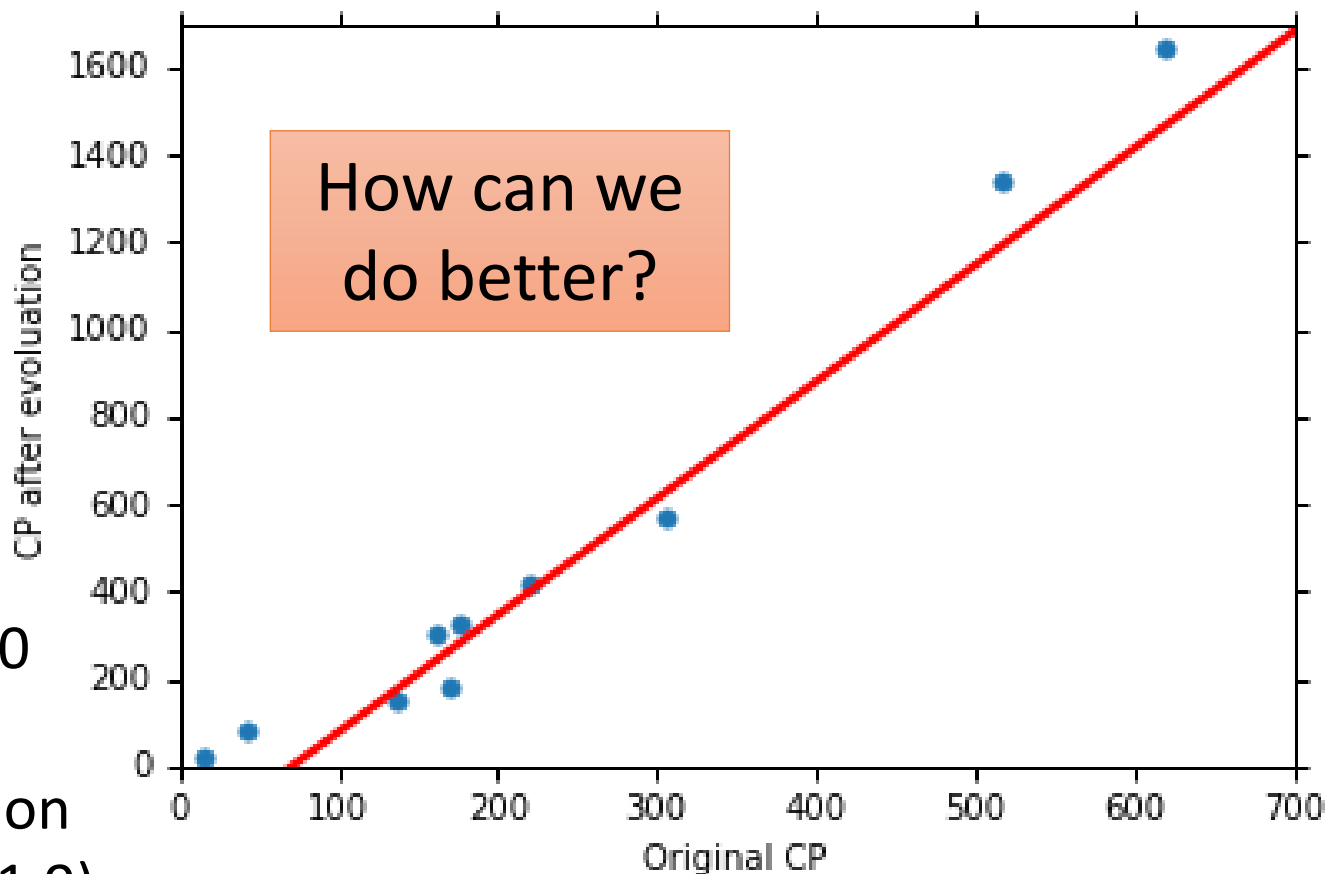
$$w = 2.7$$

Average Error on
Testing Data

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

> Average Error on
Training Data (31.9)

Another 10 pokemons as testing data



Selecting another Model

We may need some domain knowledge to choose the model.

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

It's still a linear model. (x_{cp}^2 can just be another feature.)

Best Function

$$b = -10.3$$

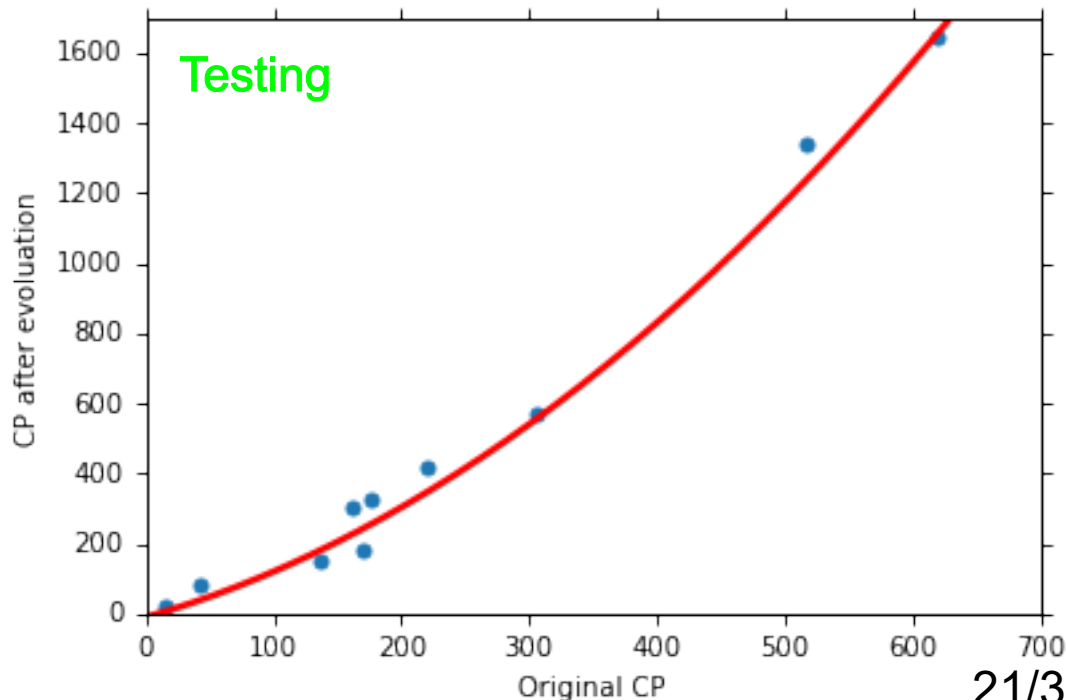
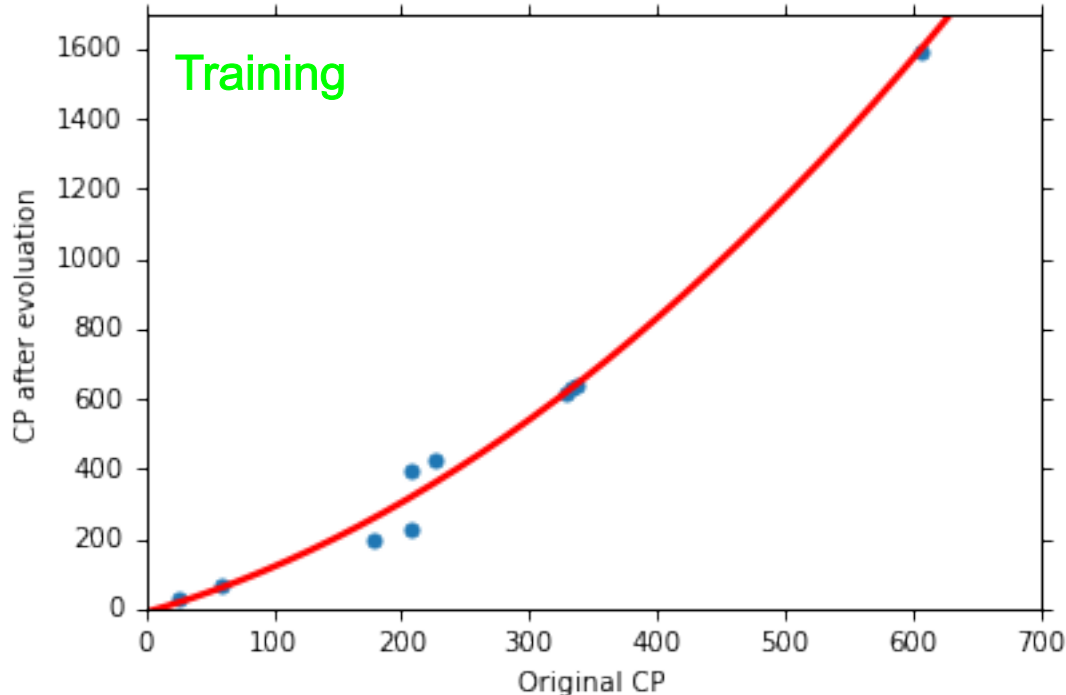
$$w_1 = 1.0, w_2 = 2.7 \times 10^{-3}$$

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

$$b = 6.4, w_1 = 0.66$$

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

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

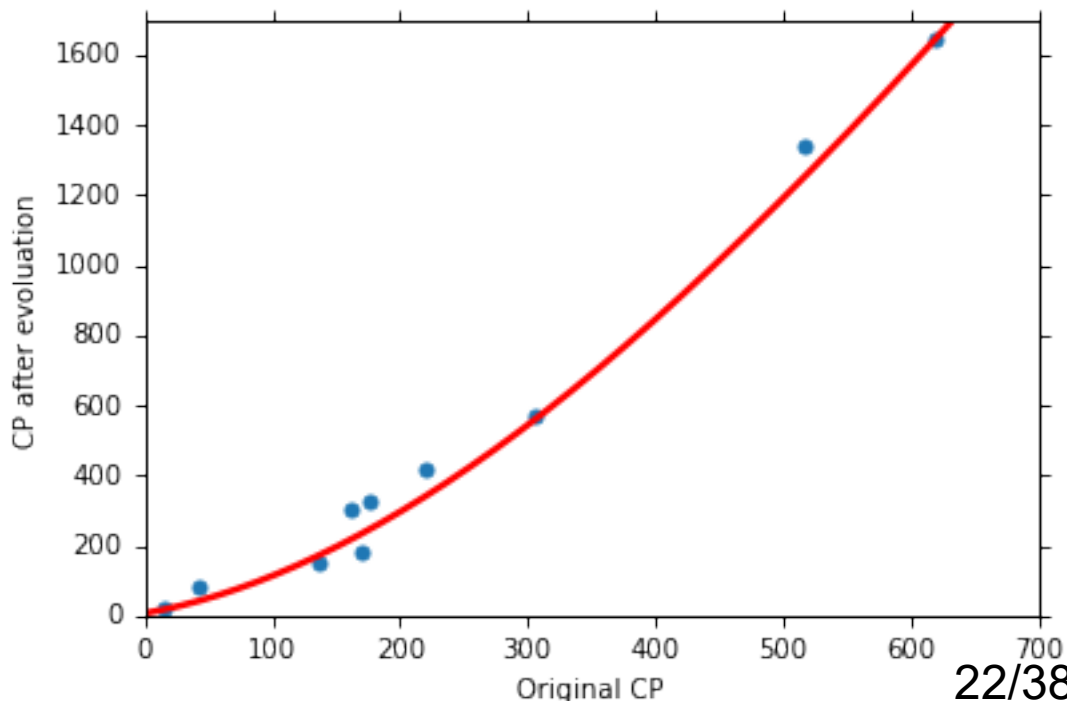
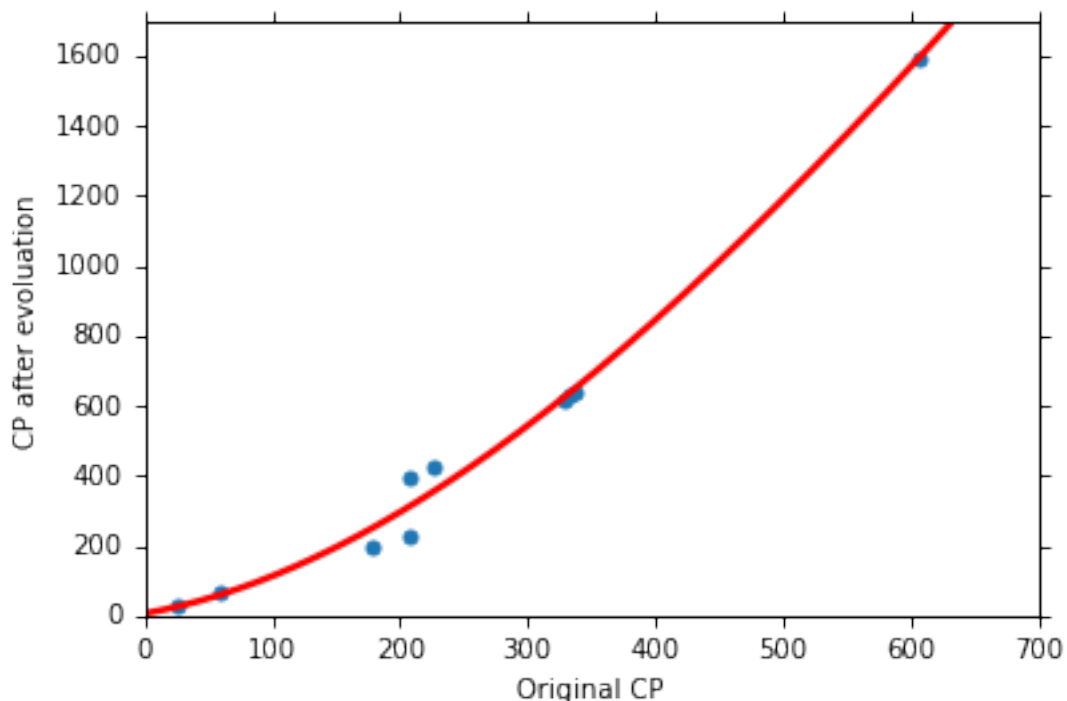
Average Error = 15.3

Testing:

Average Error = 18.1

Slightly better.

How about more complex model?



Selecting another Model

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

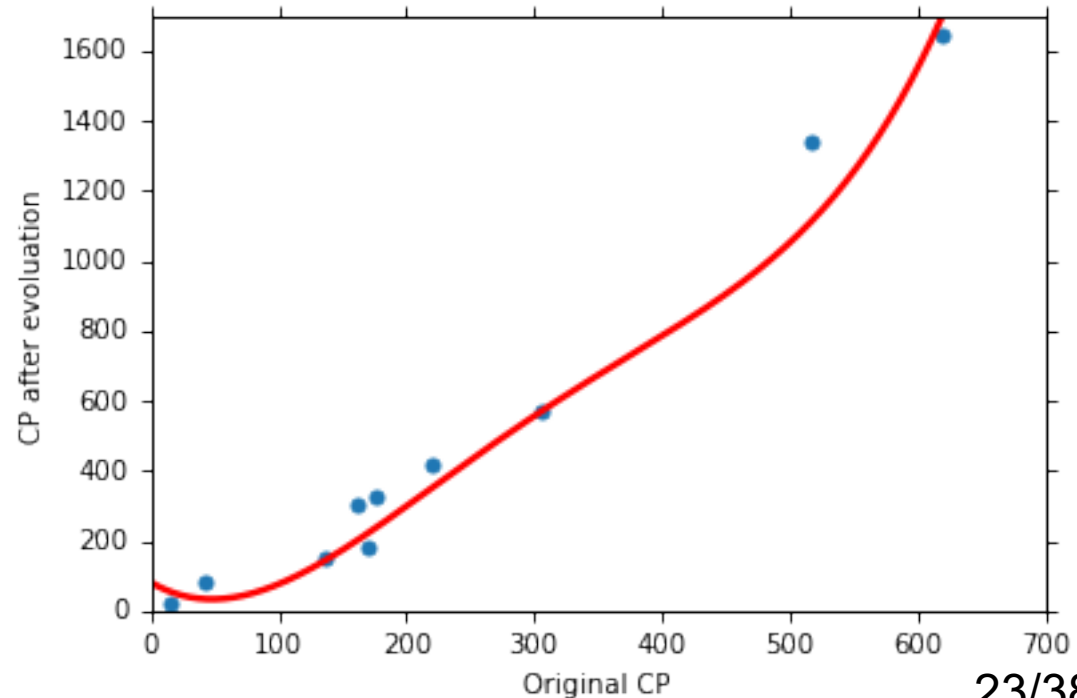
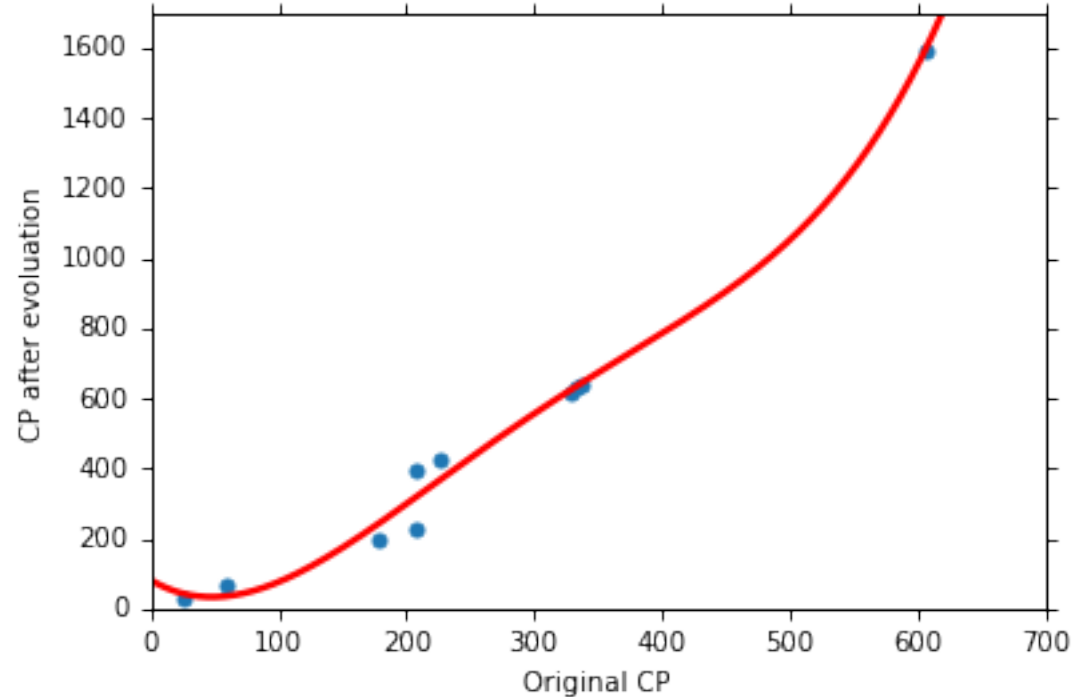
Best Function

Average Error = 14.9

Testing:

Average Error = 28.8

The results become worse ...



Selecting another Model

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

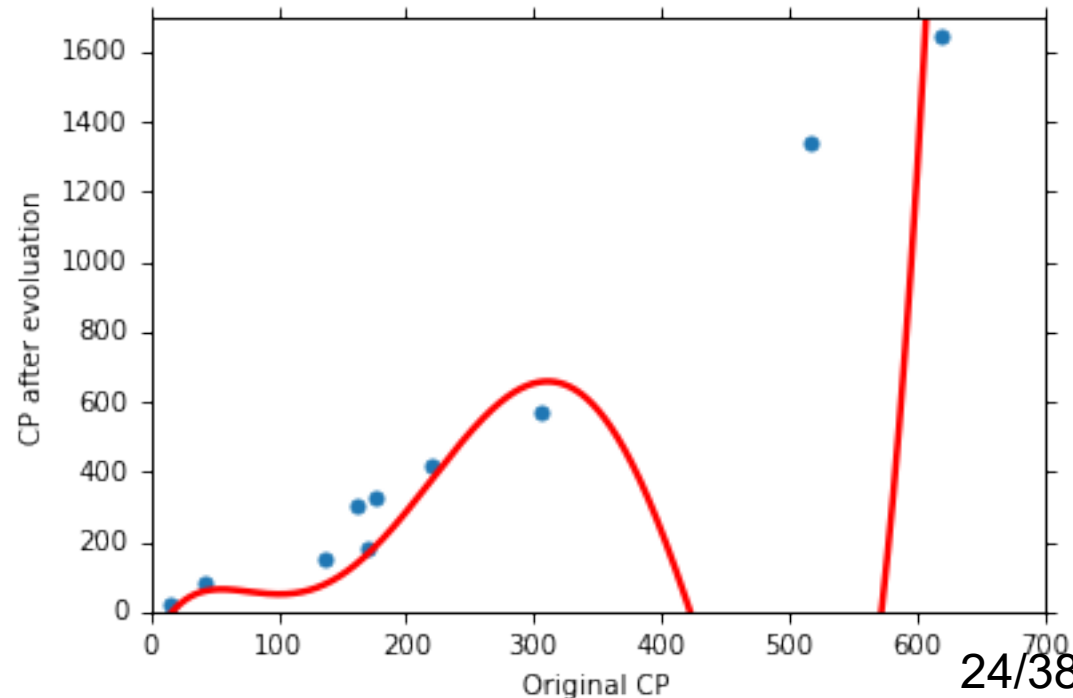
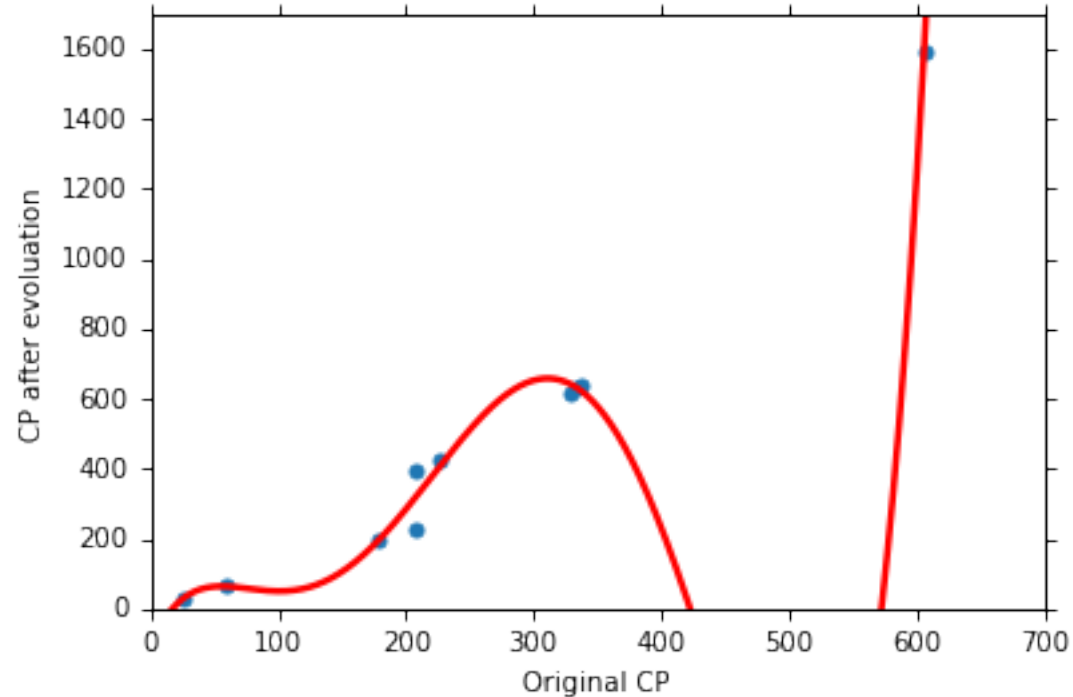
Best Function

Average Error = 12.8

Testing:

Average Error = 232.1

The results are so bad.



Model Selection

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

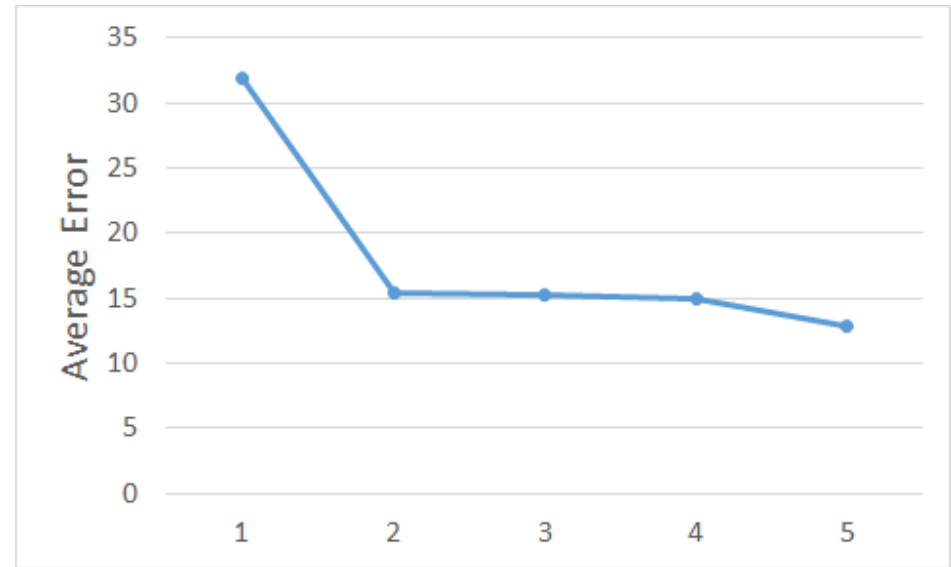
2. $y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2$

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

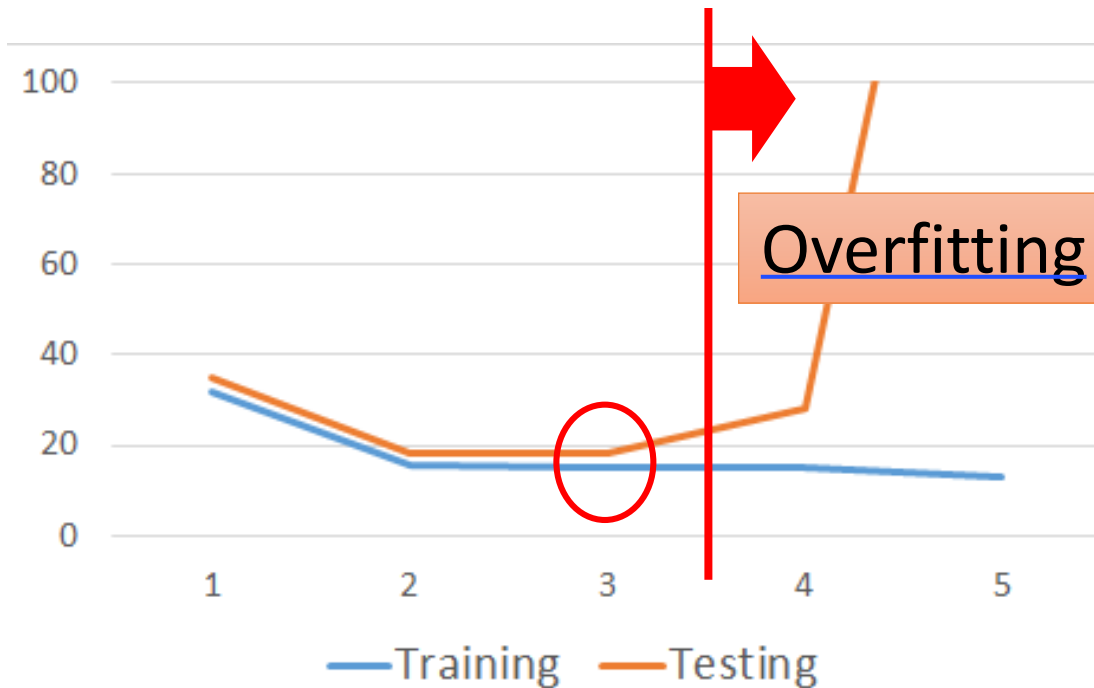


Not testing data

A more complex model yields lower error on training data.

If we can truly find the best function

Model Selection

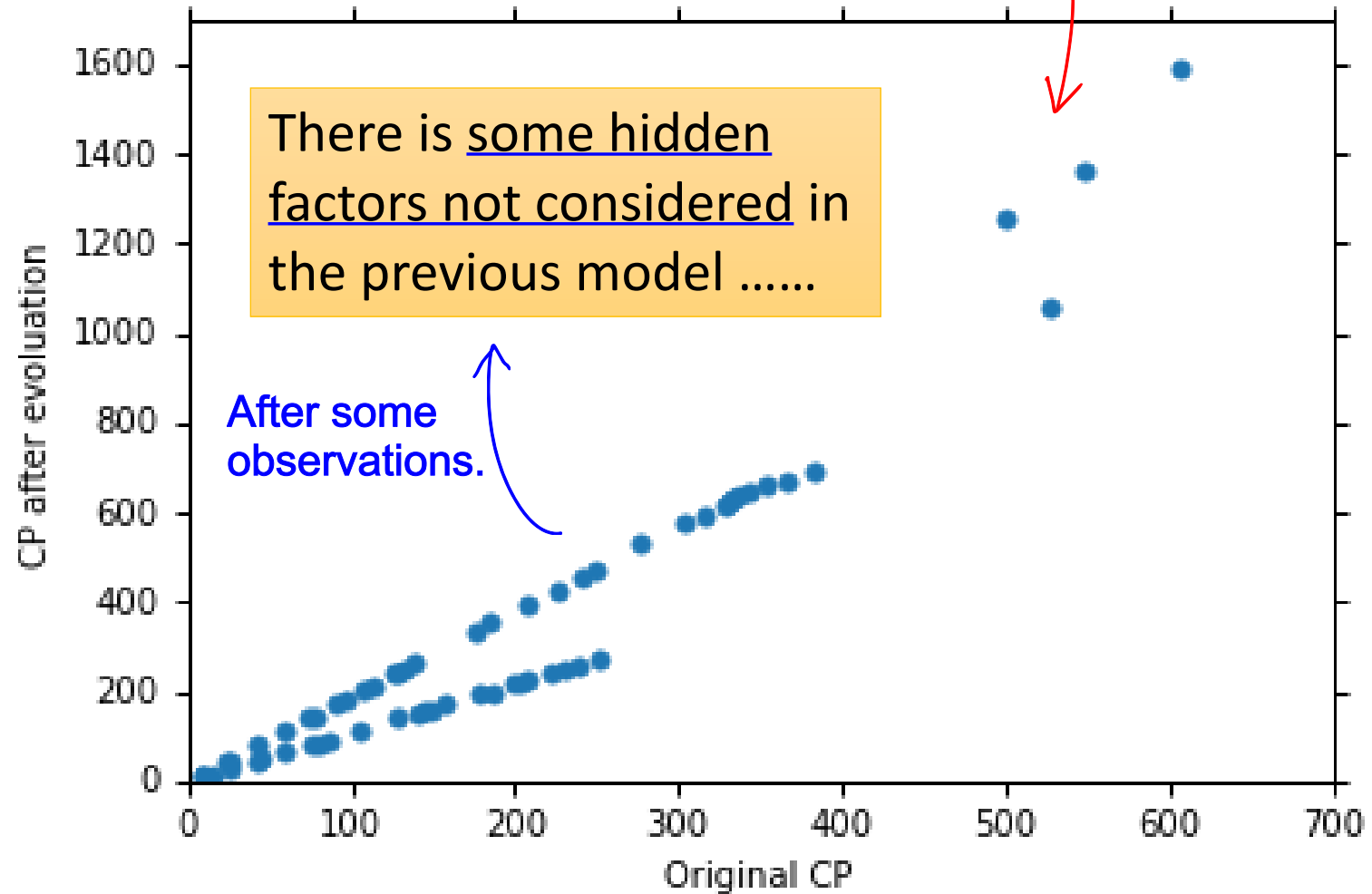


	Training	Testing
1	31.9	35.0
2	15.4	18.4
3	15.3	18.1
4	14.9	28.2
5	12.8	232.1

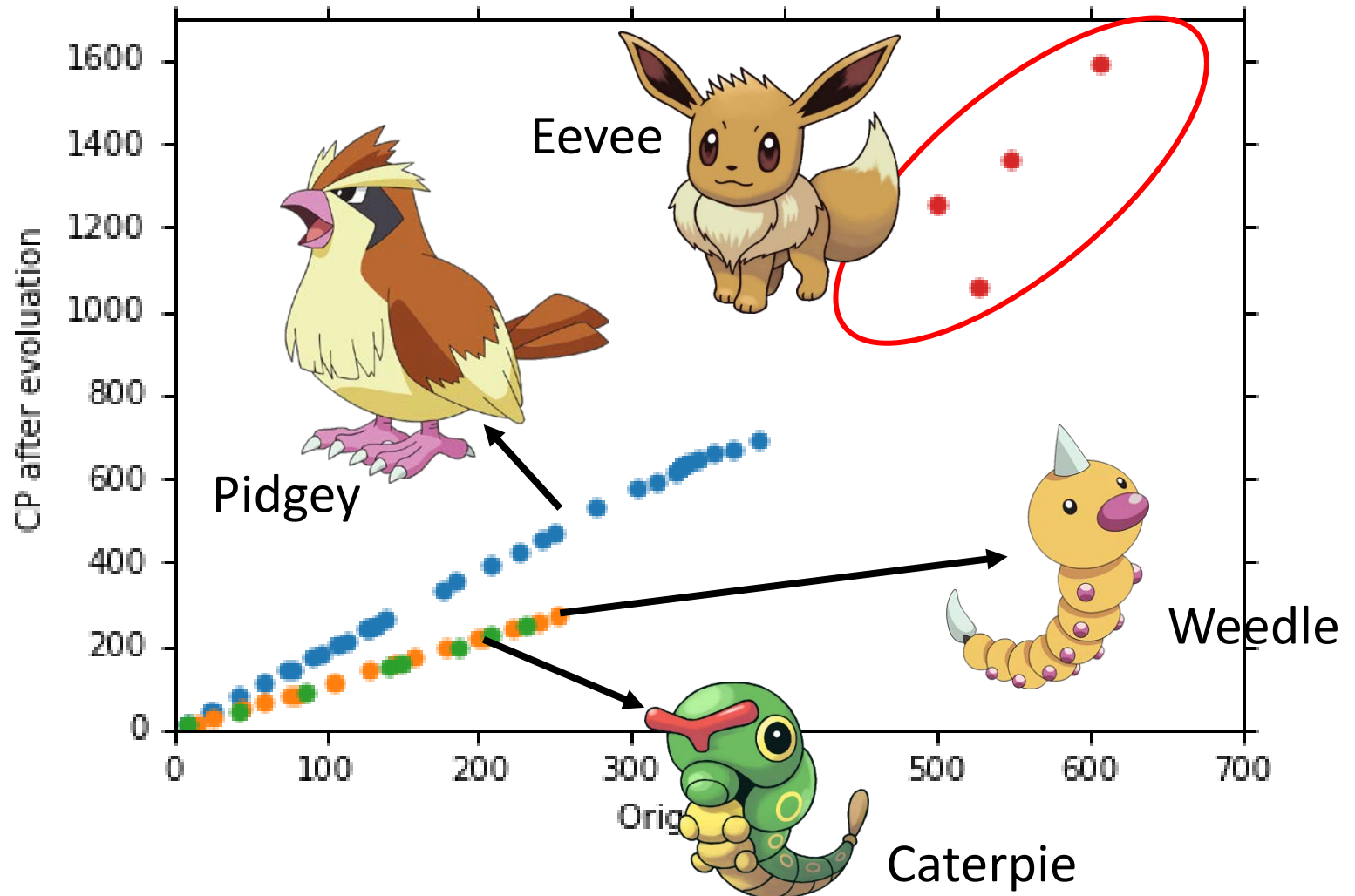
A more complex model does not always lead to better performance on testing data.

This is Overfitting.  Select suitable model

Let's collect more data



What are the hidden factors? Species?



Back to step 1: Redesign the Model

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

Each species has its own model.
 w, b

x_s = species of x

x



Linear model?

How to combine these
functions into one single
linear model?

If $x_s = \text{Pidgey}$:

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

If $x_s = \text{Weedle}$:

$$y = b_2 + w_2 \cdot x_{cp}$$

If $x_s = \text{Caterpie}$:

$$y = b_3 + w_3 \cdot x_{cp}$$

If $x_s = \text{Eevee}$:

$$y = b_4 + w_4 \cdot x_{cp}$$



y

Back to step 1: Redesign the Model

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

Linear model?

$$y = b_1 \cdot \delta(x_s = \text{Pidgey})$$

$$+w_1 \cdot \delta(x_s = \text{Pidgey})x_{cp}$$

$$+b_2 \cdot \delta(x_s = \text{Weedle})$$

$$+w_2 \cdot \delta(x_s = \text{Weedle})x_{cp}$$

$$+b_3 \cdot \delta(x_s = \text{Caterpie})$$

$$+w_3 \cdot \delta(x_s = \text{Caterpie})x_{cp}$$

$$+b_4 \cdot \delta(x_s = \text{Eevee})$$

$$+w_4 \cdot \delta(x_s = \text{Eevee})x_{cp}$$

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

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

Back to step 1: Redesign the Model

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

Linear model?

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

When the species is Pidgery.

x_{cp}

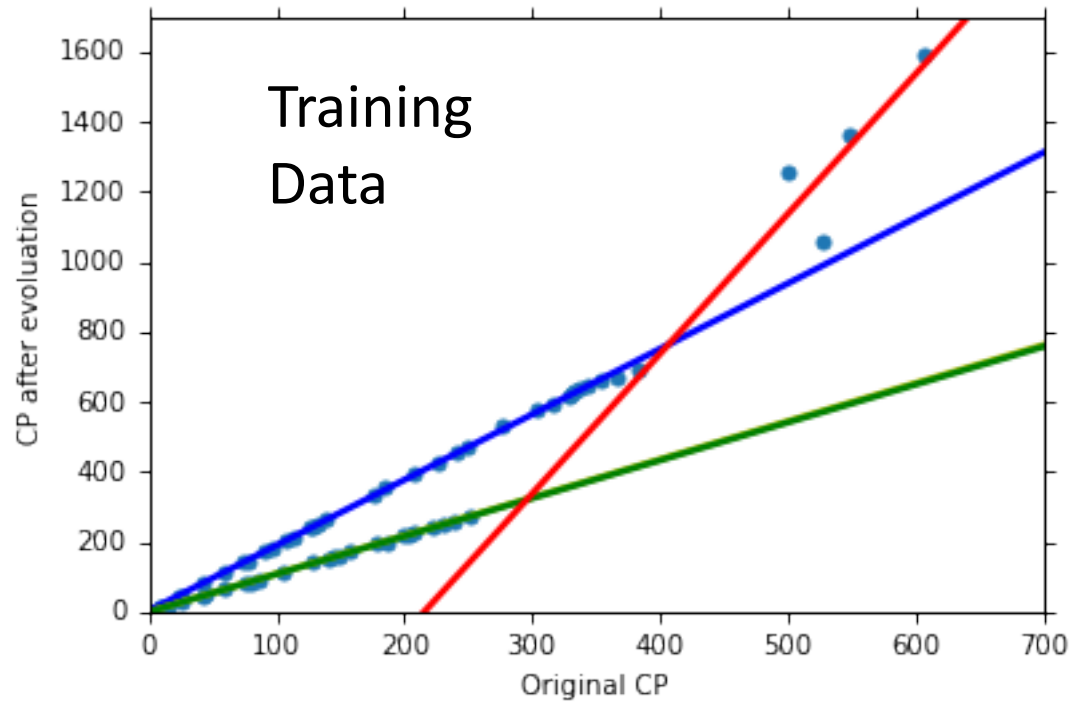
$$\delta(x_s = \text{Pidgery})$$

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

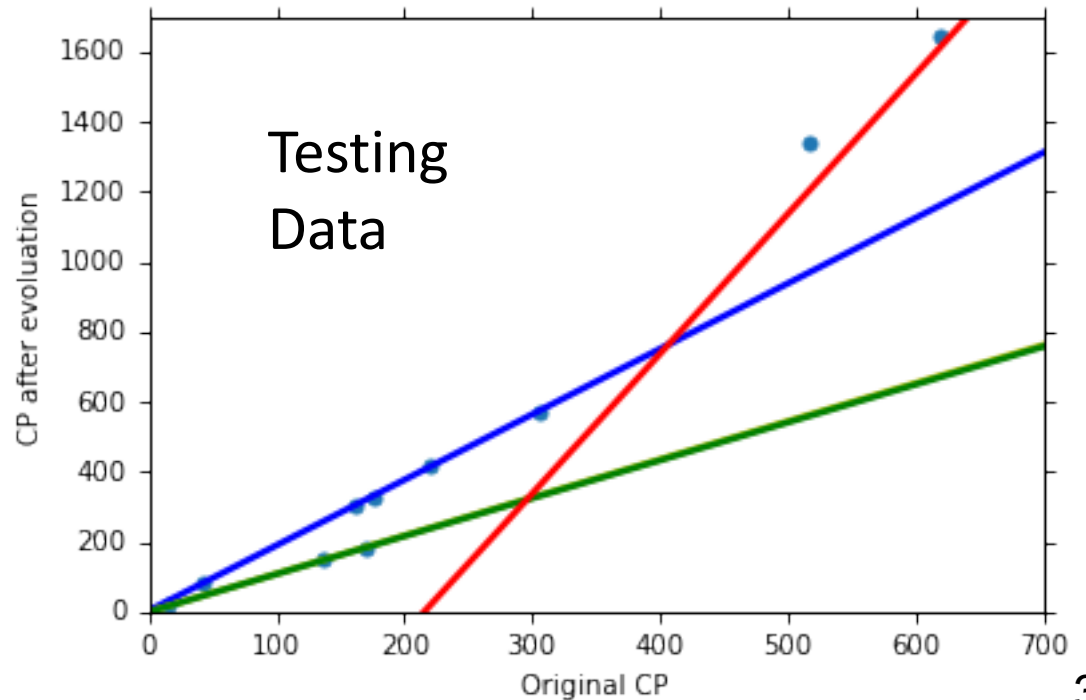
If $x_s = \text{Pidgery}$

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

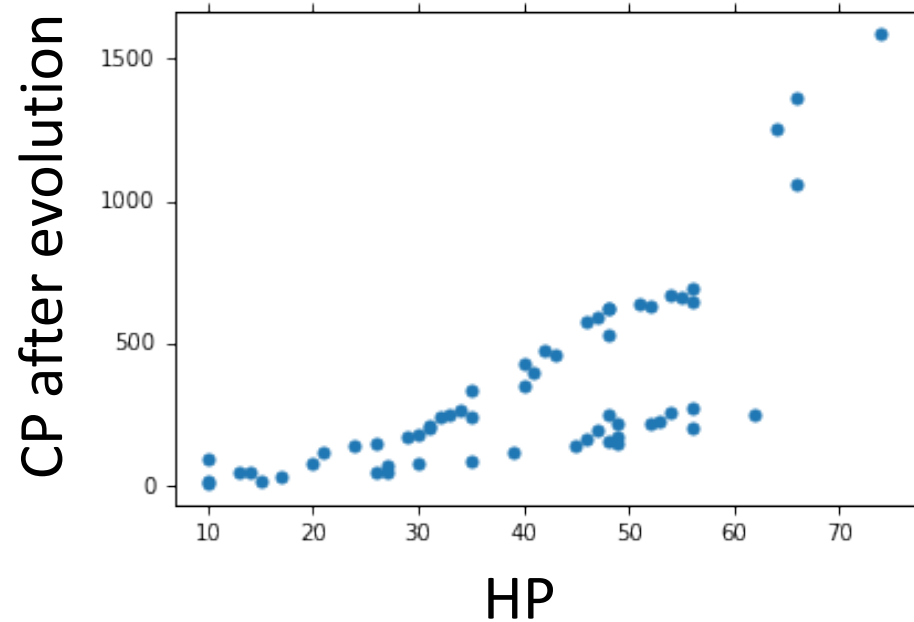
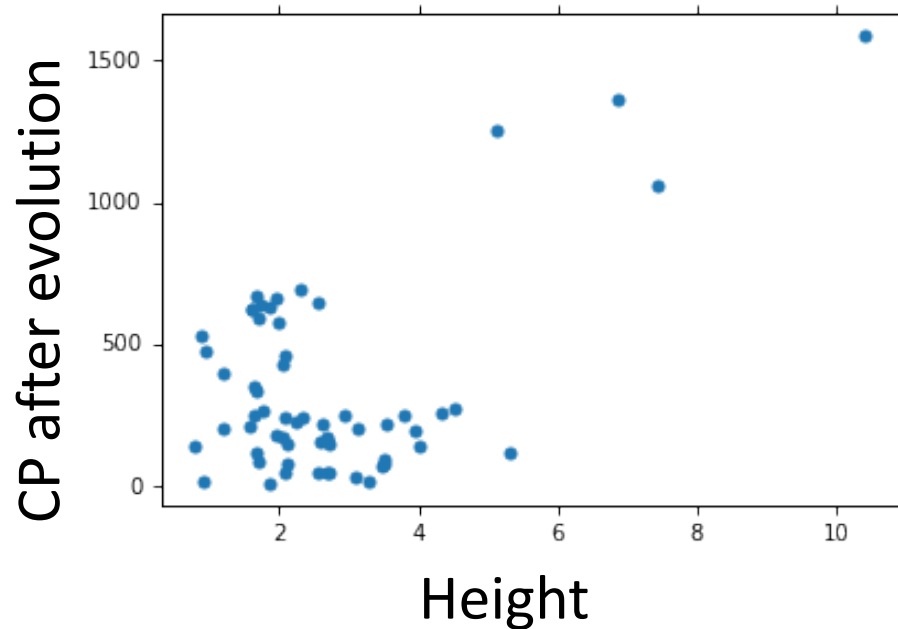
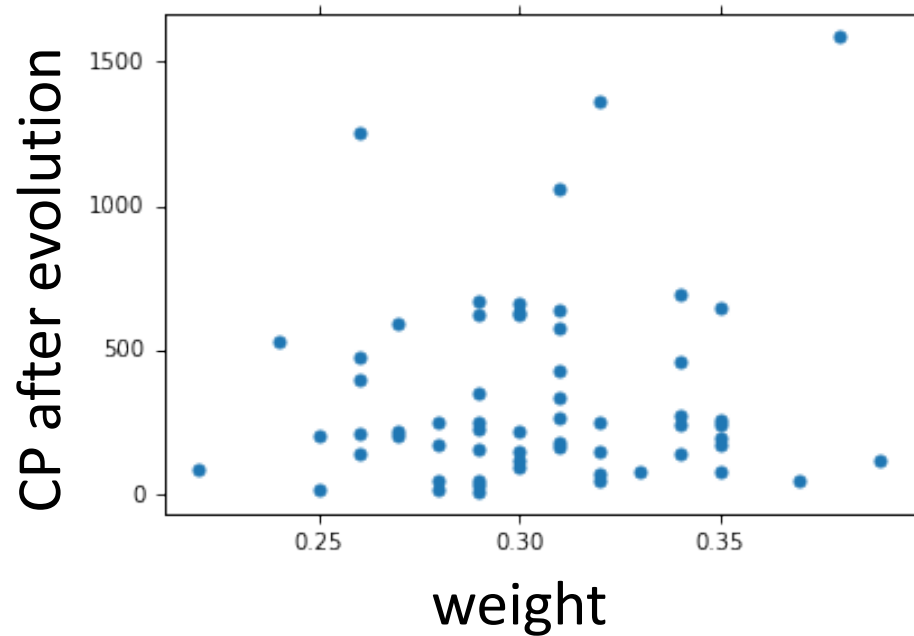
Average error
= 3.8



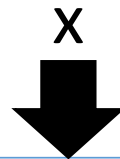
Average error
= 14.3



Are there any other hidden factors?



Back to step 1: Redesign the Model Again



Another more complex model.

$$\begin{aligned} \text{If } x_s = \text{Pidgey:} \quad & y' = b_1 + w_1 \cdot x_{cp} + w_5 \cdot (x_{cp})^2 \\ \text{If } x_s = \text{Weedle:} \quad & y' = b_2 + w_2 \cdot x_{cp} + w_6 \cdot (x_{cp})^2 \\ \text{If } x_s = \text{Caterpie:} \quad & y' = b_3 + w_3 \cdot x_{cp} + w_7 \cdot (x_{cp})^2 \\ \text{If } x_s = \text{Eevee:} \quad & y' = b_4 + w_4 \cdot x_{cp} + w_8 \cdot (x_{cp})^2 \\ & y = y' + w_9 \cdot x_{hp} + w_{10} \cdot (x_{hp})^2 \\ & + w_{11} \cdot x_h + w_{12} \cdot (x_h)^2 + w_{13} \cdot x_w + w_{14} \cdot (x_w)^2 \end{aligned}$$



Training Error
= 1.9



Testing Error
= 102.3

Overfitting!

We want the model to be more general.

Back to step 2: Regularization

We want not only the error to be small but also the parameters to be small. \equiv **Weight decay**

Corresponds to the input directly.

The functions with smaller w_i are better

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

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

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

➤ Smaller w_i means ...

smoother

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

$w_i \searrow \Rightarrow \Delta y \searrow$

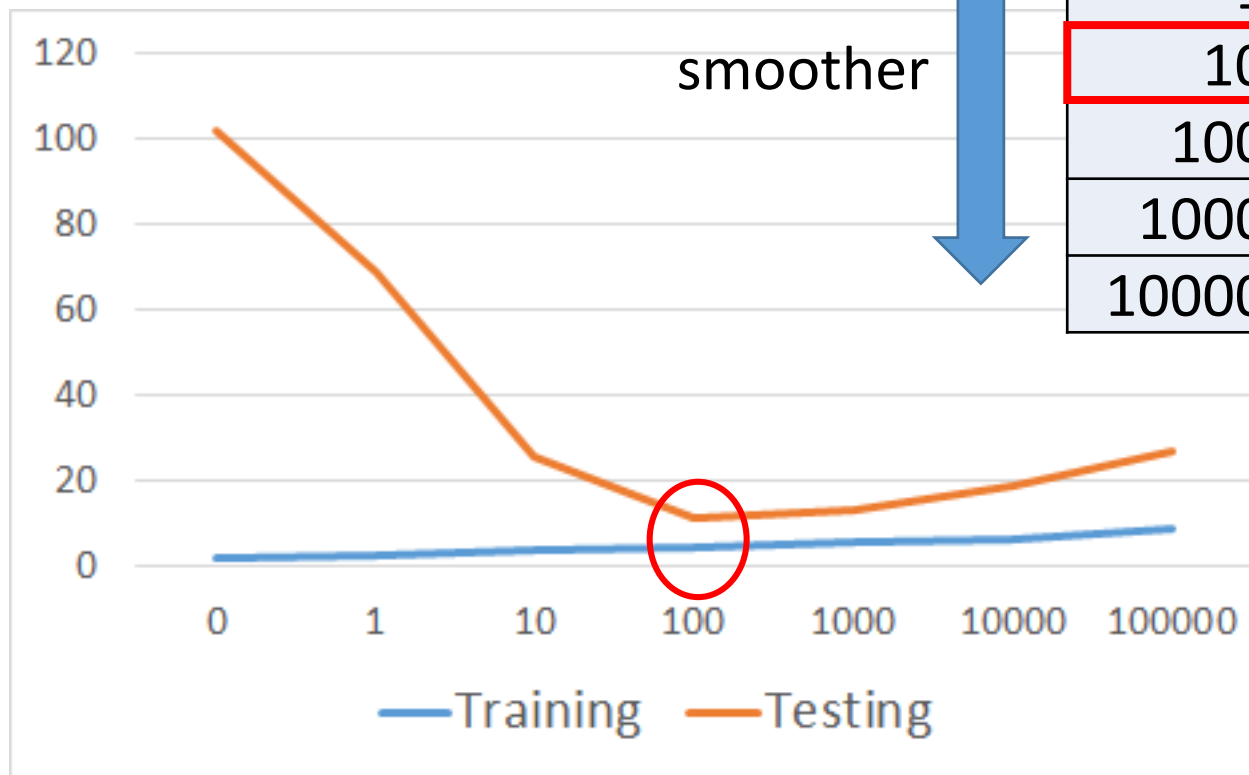
Nothing to do with b .

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

➤ We believe smoother function is more likely to be correct

Do you have to apply regularization on bias? **No**

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 weight more 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