# Regression

李宏毅、吳沛遠

# Regression: Output a scalar

- Stock Market Forecast

$f($  $) =$ Dow Jones Industrial Average at tomorrow

- Self-driving Car

$f($  $) =$ 方向盤角度

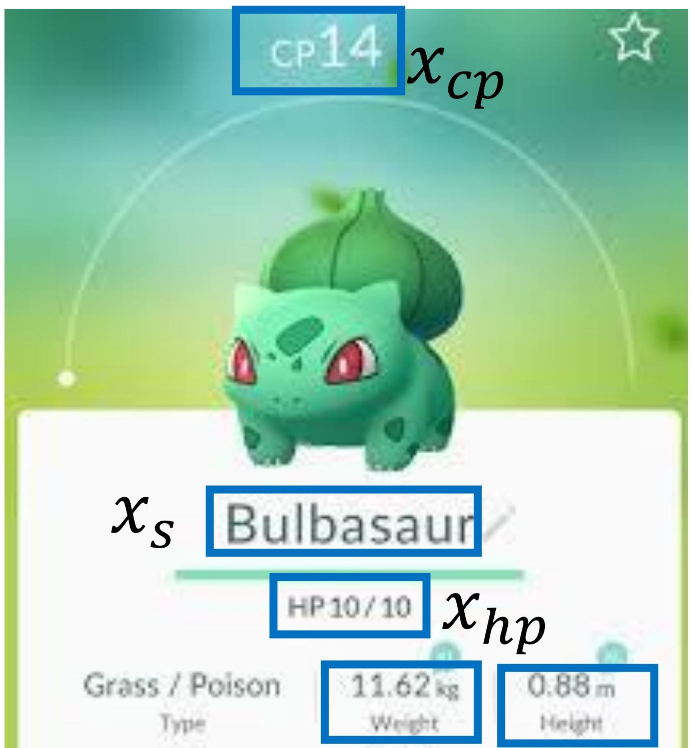
- Recommendation

$f($ 使用者 A 商品 B $) =$ 購買可能性

# Example Application

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

$f($




$x_{cp}$ $x_s$ $x_{hp}$ $x_w$ $x_h$ $x$

$) =$ CP after evolution $y$

# Step 1: Model

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

w and b are parameters (can be any value)

A set of function

Model

$f_1, f_2 \cdots$

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

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

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

…… infinite

$f($ 

$x) =$ CP after evolution $y$

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

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

feature

$w_i$: weight, b: bias

# Step 2: Goodness of Function

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

A set of function

Model

$f_1, f_2 \cdots$

Training Data

function input:

function Output (scalar):

# Step 2: Goodness of Function

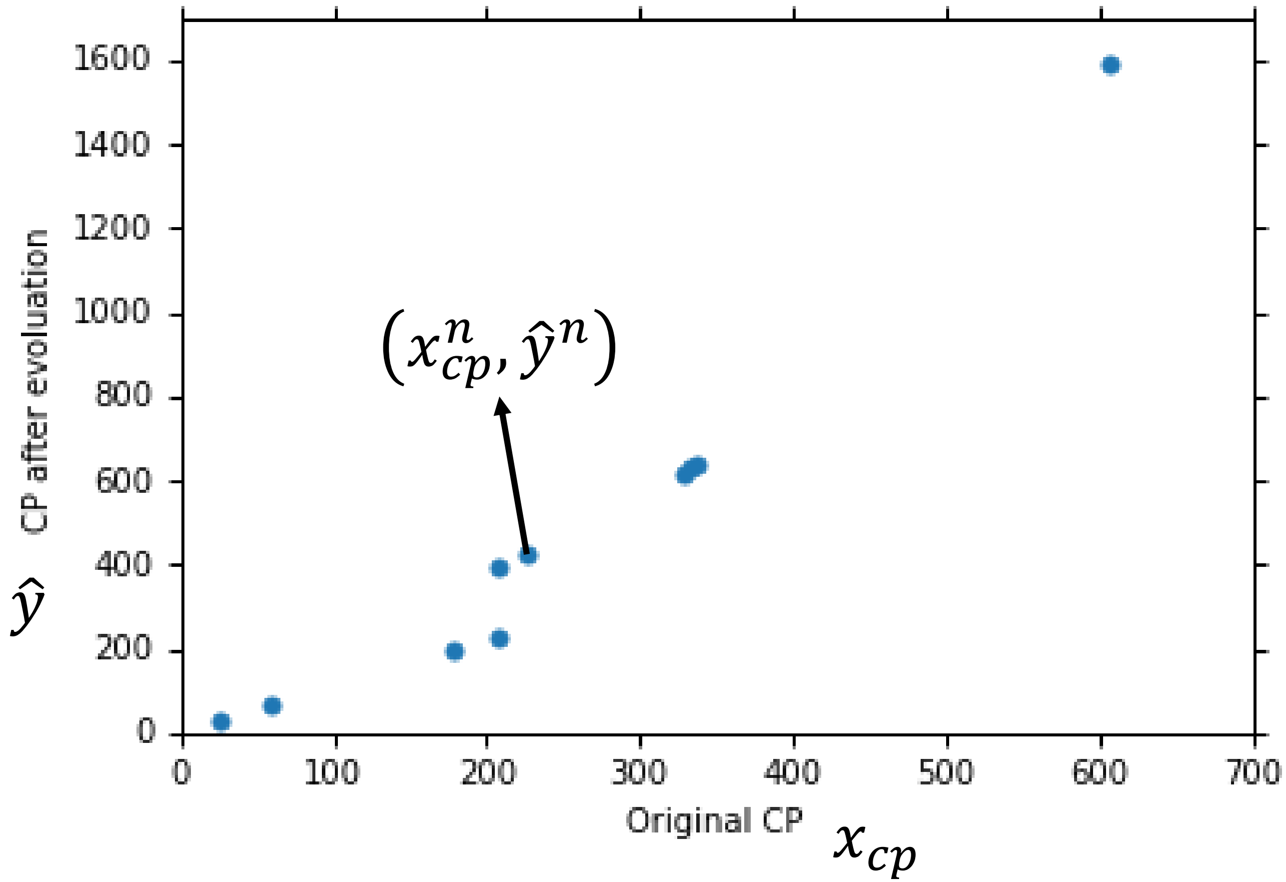
Training Data:
10 pokemons

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

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

$\vdots$

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

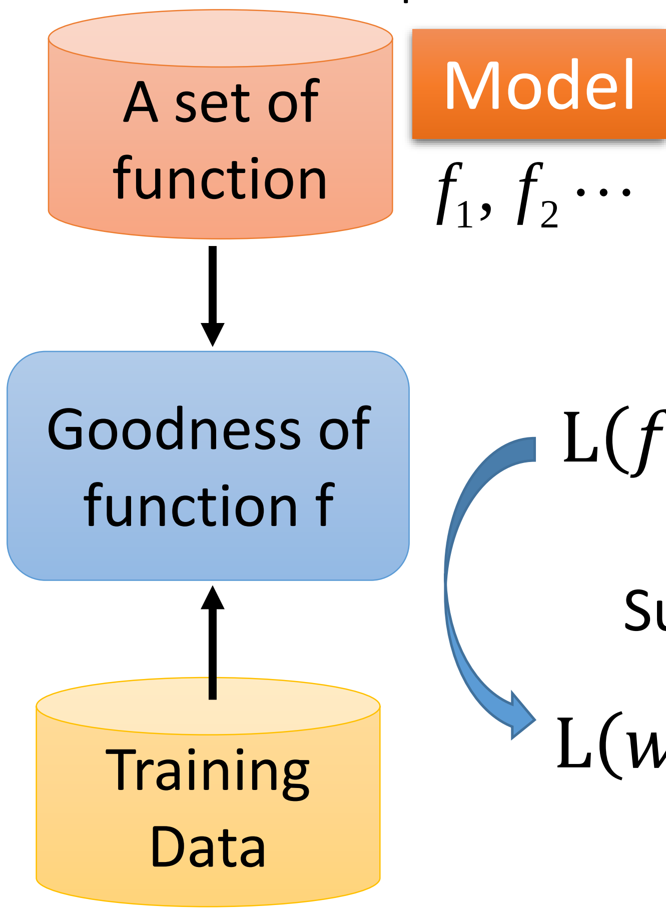



This is real data.

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

# Step 2: Goodness of Function

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




Loss function $L$:

Input: a function, output: how bad it is

Estimation error

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

Sum over examples

Estimated y based on input function

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

# Step 2: Goodness of Function

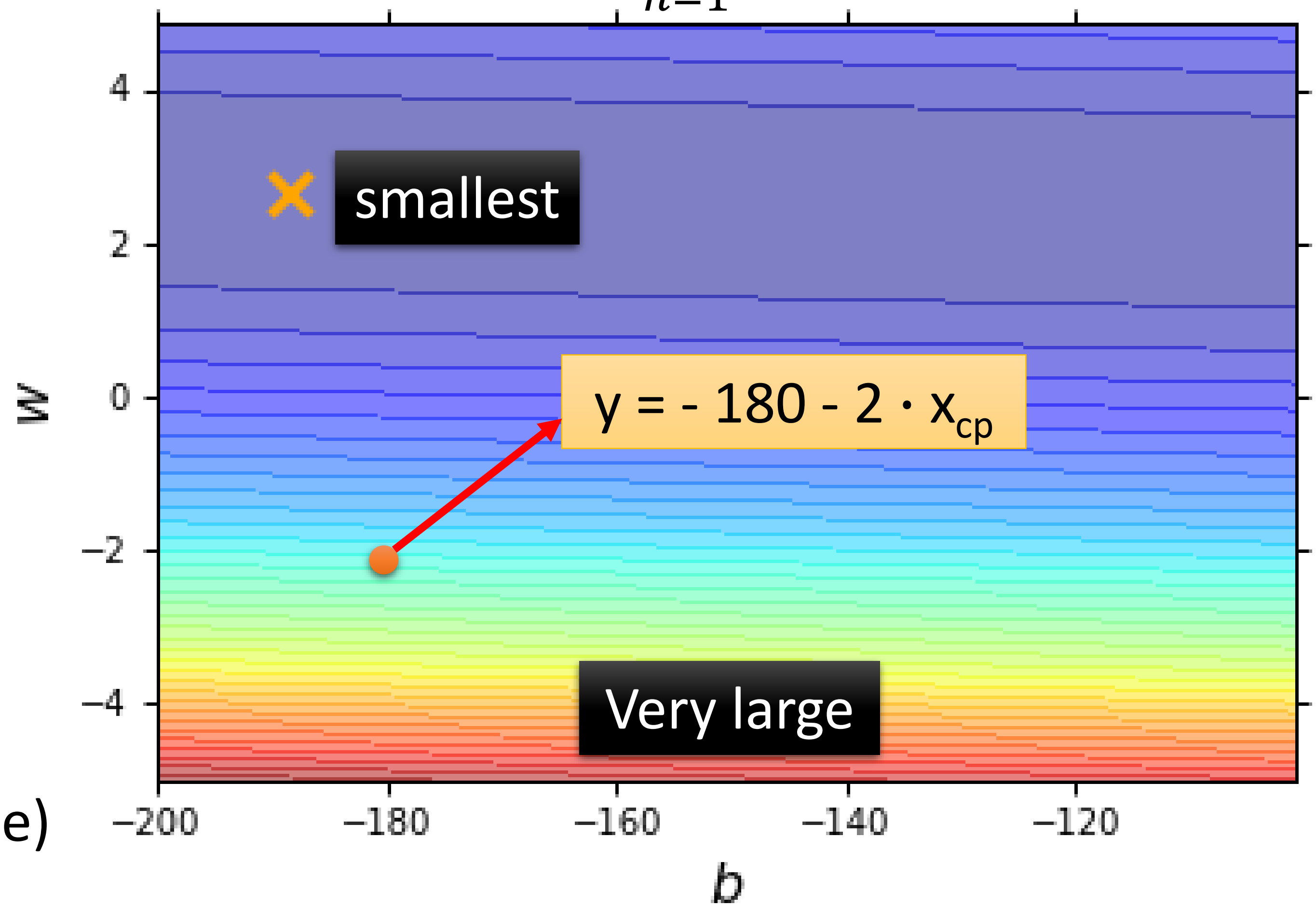
- Loss Function

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

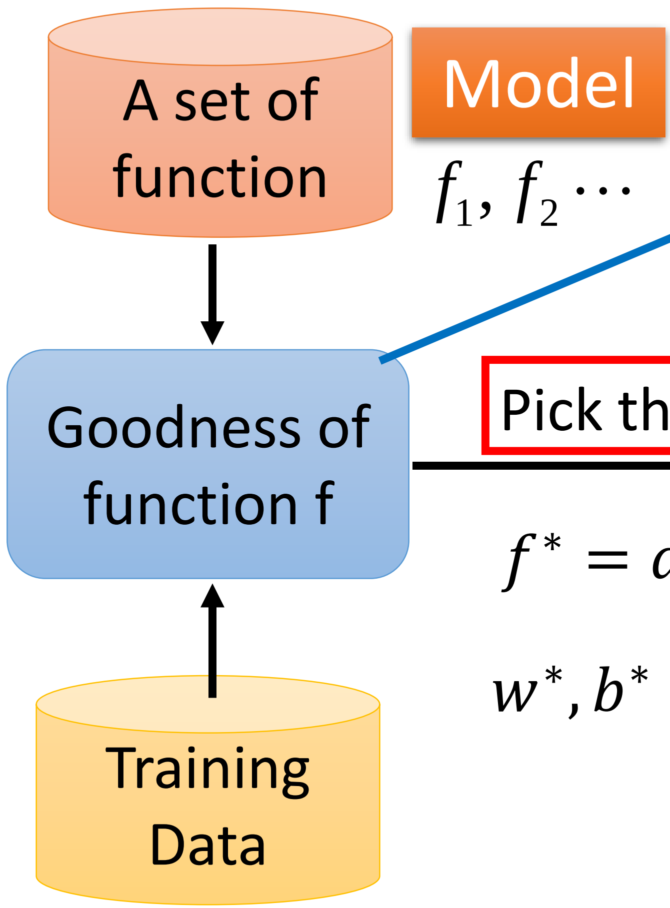
Each point in the figure is a function

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

(true example)



# Step 3: Best Function




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

Pick the "Best" Function

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

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

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

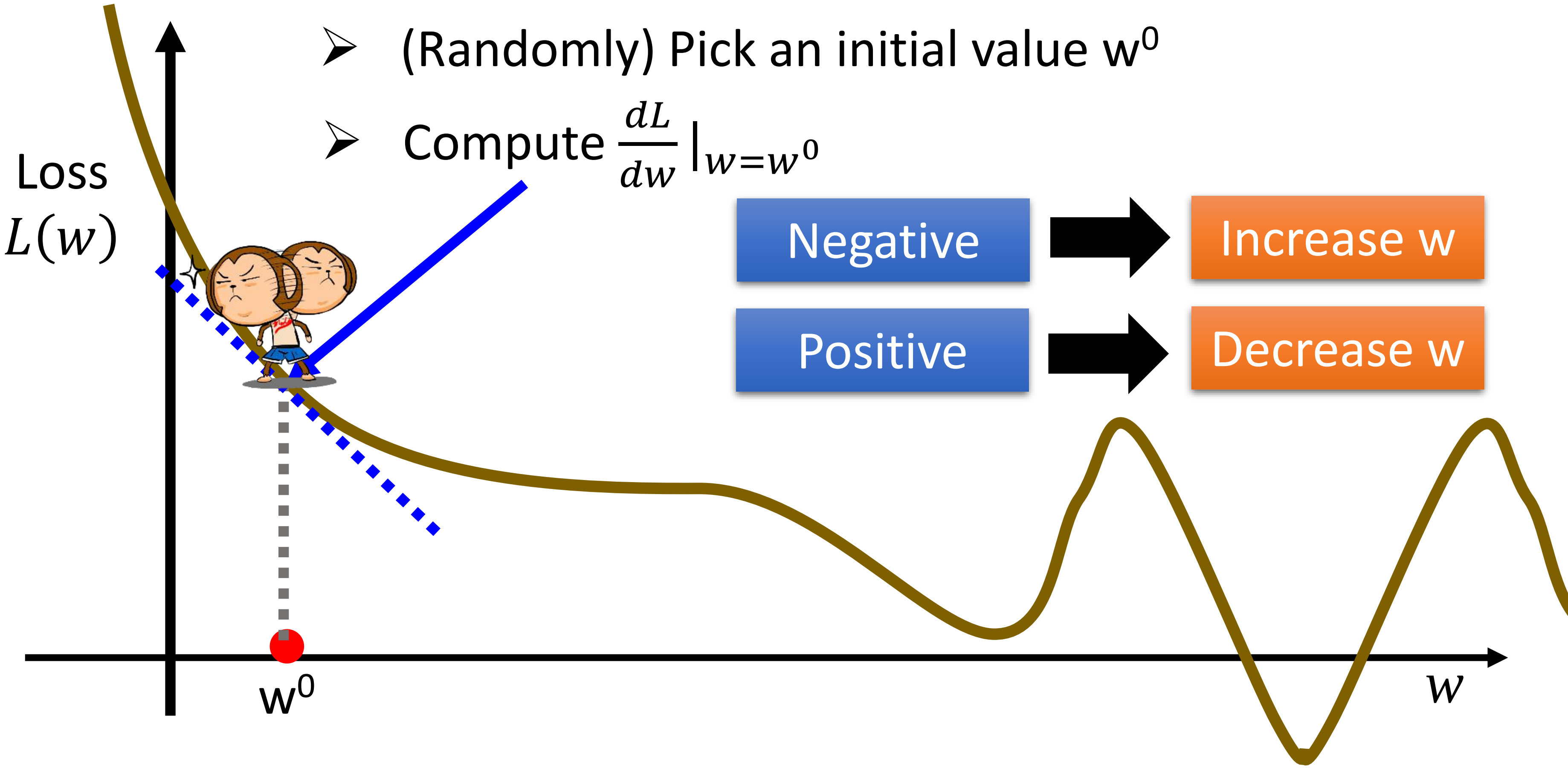
Gradient Descent

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

# 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 $\frac{dL}{dw}|_{w=w^0}$



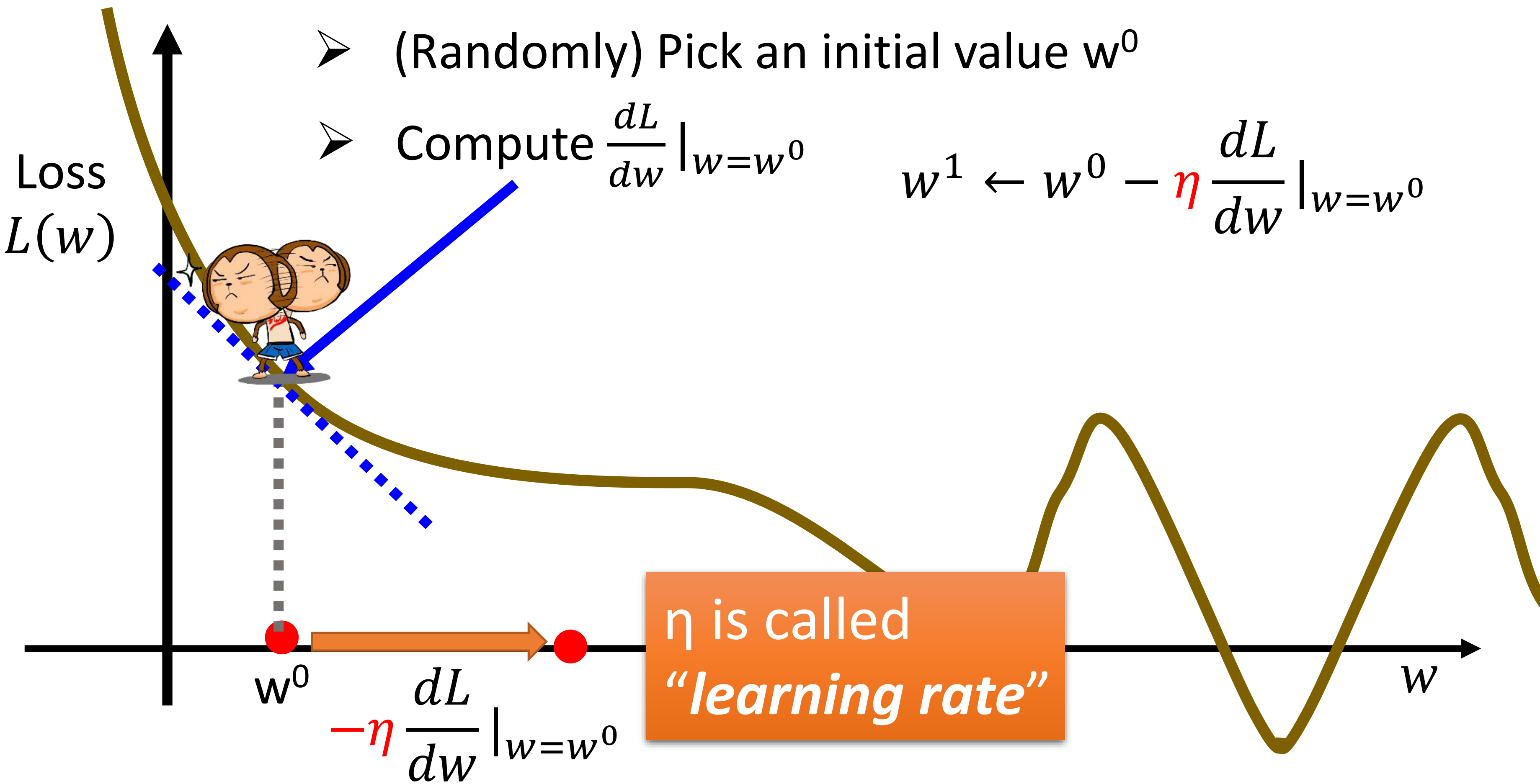
http://chico386.pixnet.net/album/photo/171572850

# 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 $\frac{dL}{dw}|_{w=w^0}$

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



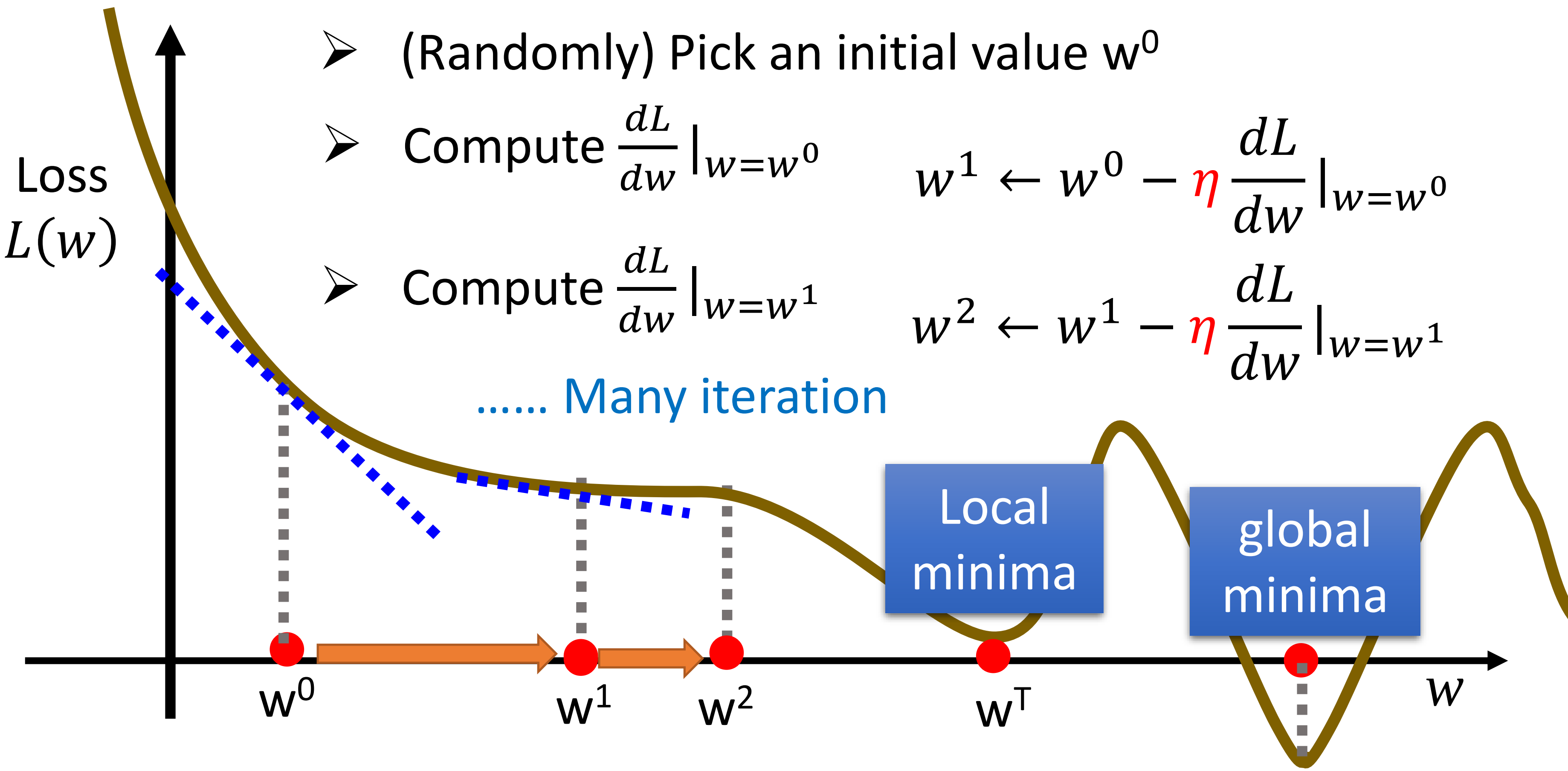
# Step 3: Gradient Descent

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

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

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

…… Many iteration



# Step 3: Gradient Descent

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

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

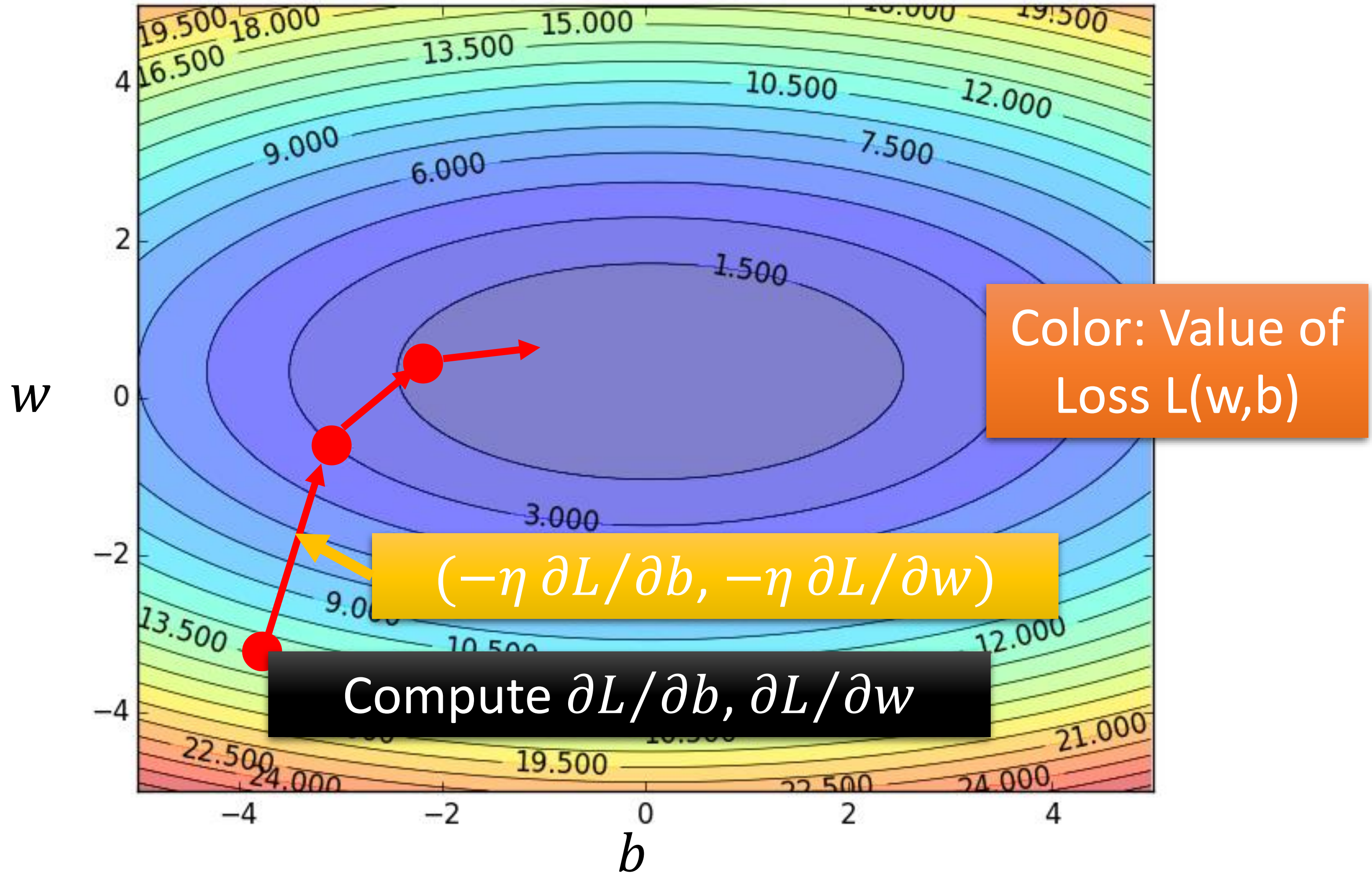
  - (Randomly) Pick an initial value $w^0$, $b^0$
  - Compute $\frac{\partial L}{\partial w}|_{w=w^0,b=b^0}$, $\frac{\partial L}{\partial b}|_{w=w^0,b=b^0}$

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

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

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

# Step 3: Gradient Descent



# Step 3: Gradient Descent

- When solving:

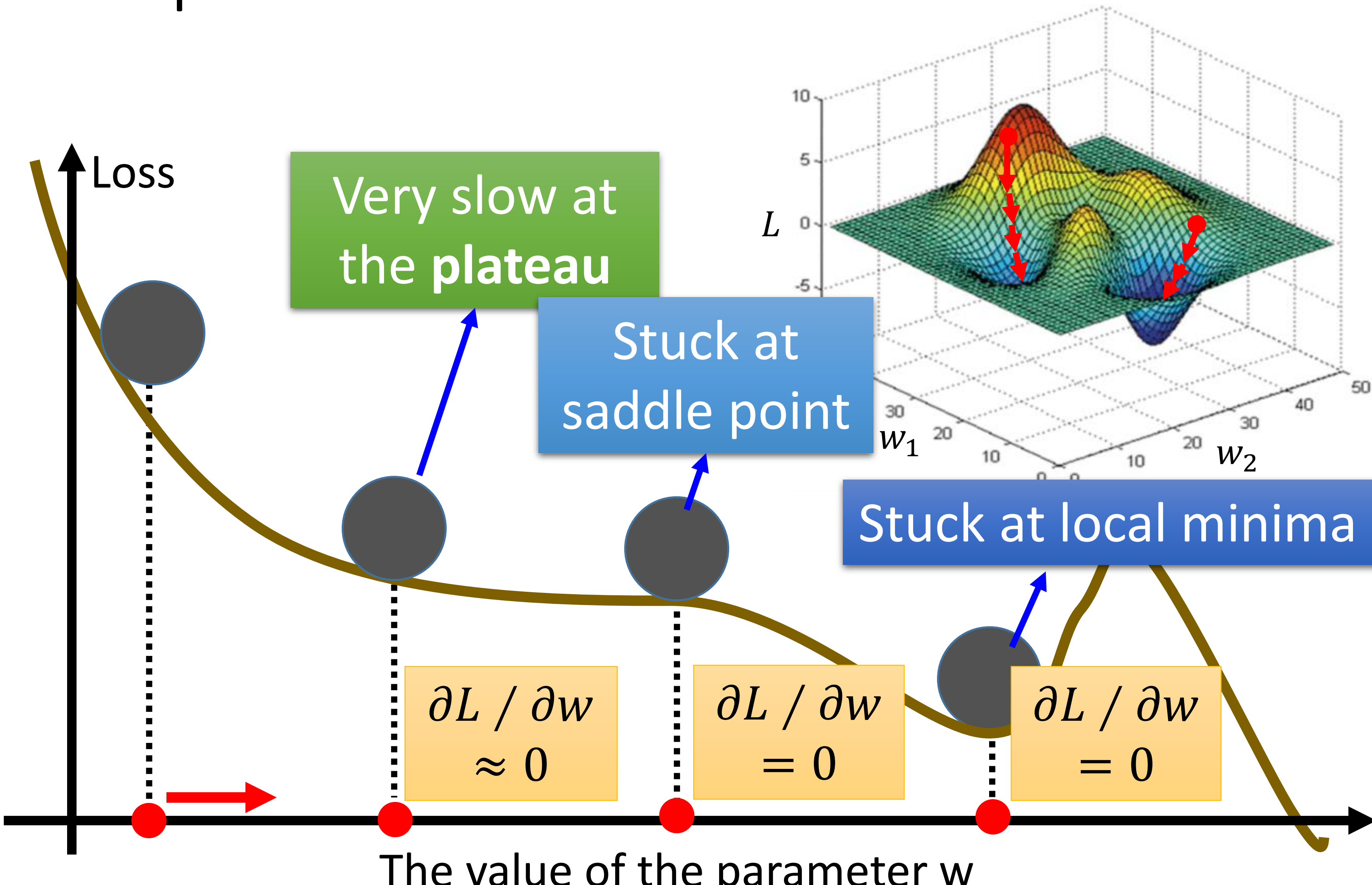
$$\theta^* = \arg\max_{\theta} L(\theta) \quad \text{by gradient descent}$$

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

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

Is this statement correct?

# Step 3: Gradient Descent



# Step 3: Gradient Descent

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

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

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

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

# Step 3: Gradient Descent

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

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

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

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

# How's the results?

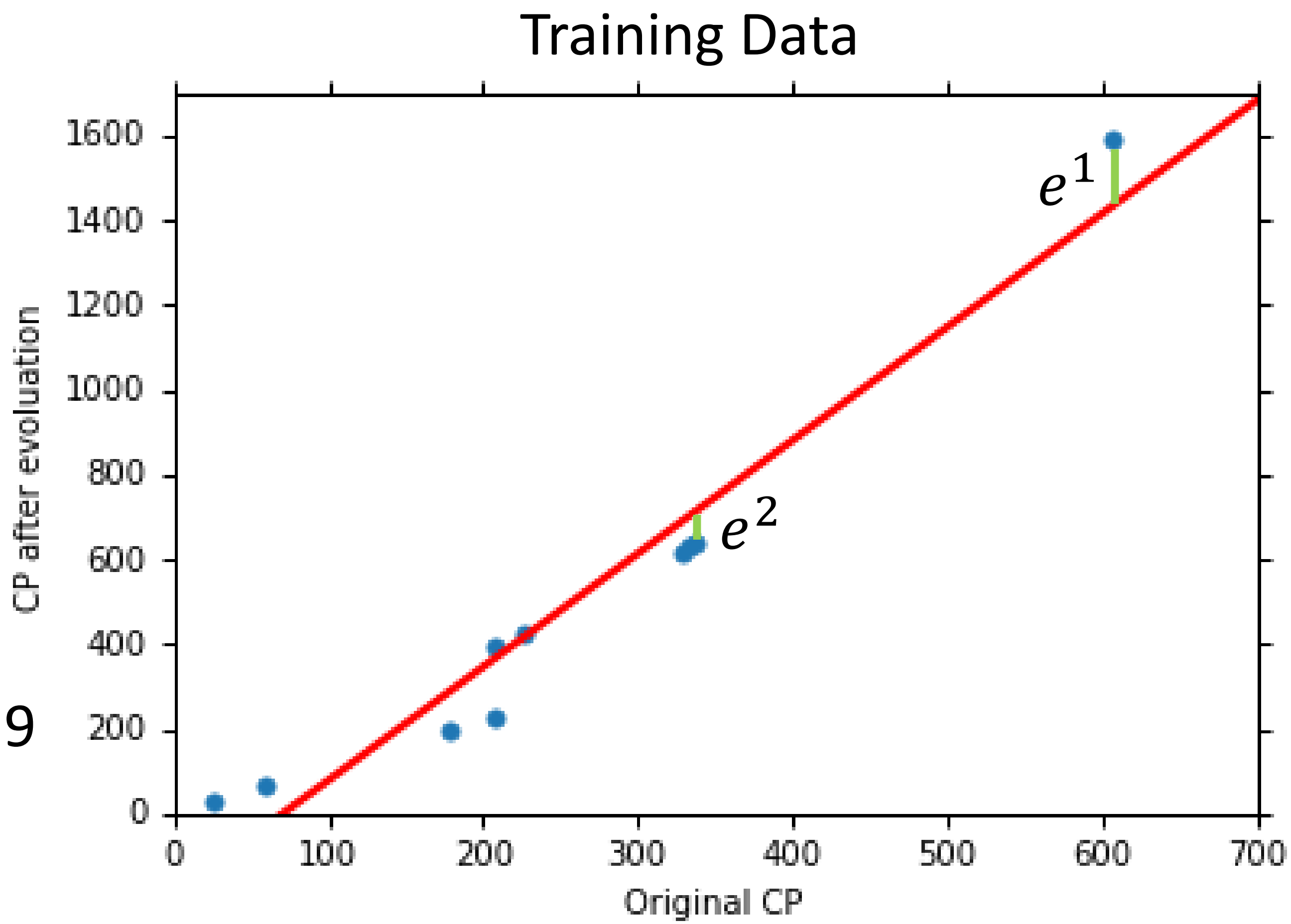
$y = b + w \cdot x_{cp}$

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)

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

b = -188.4

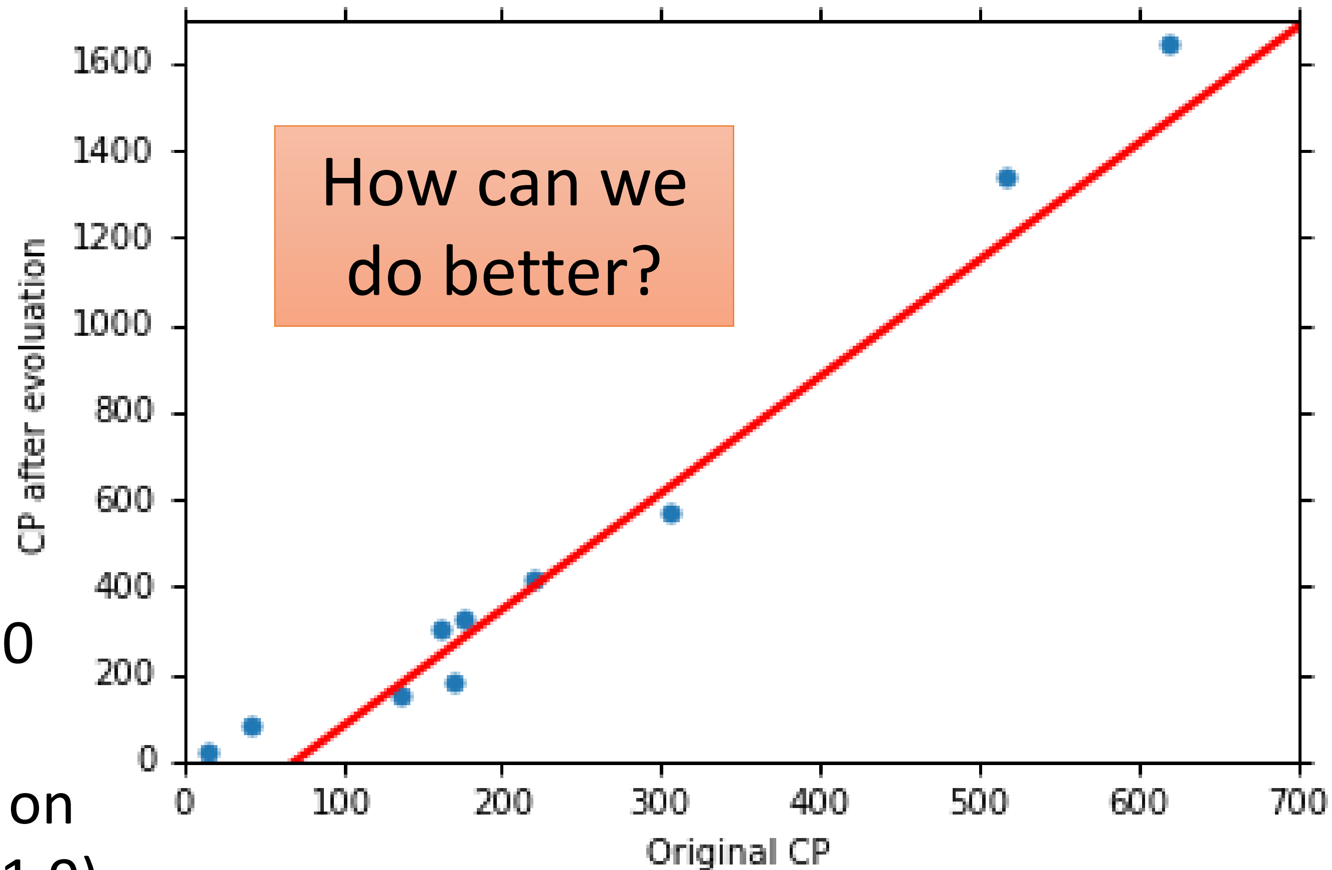
w = 2.7

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*

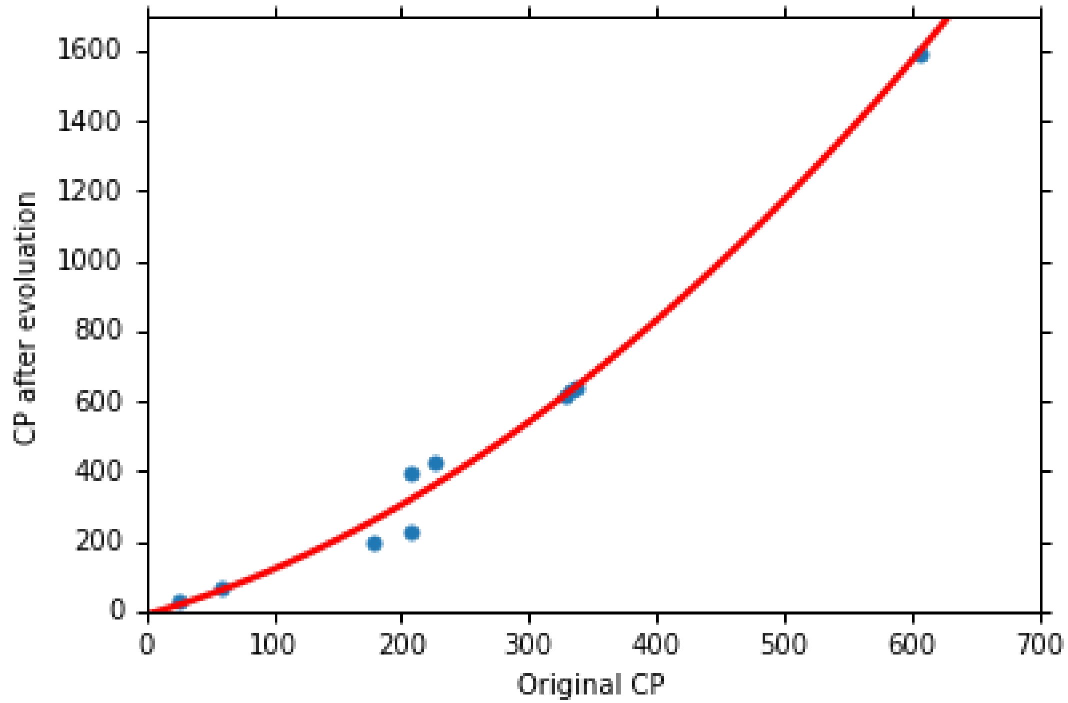
$$y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2$$

### *Best Function*

$b = -10.3$

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

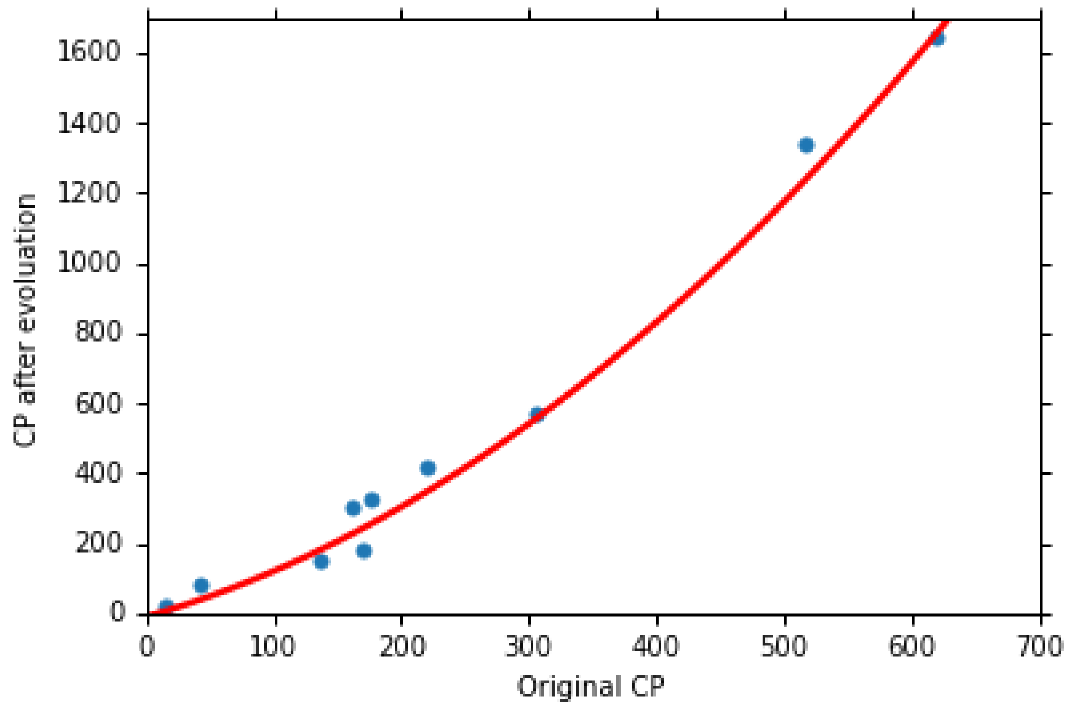
Average Error = 15.4




### *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*

$b = 6.4$, $w_1 = 0.66$

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

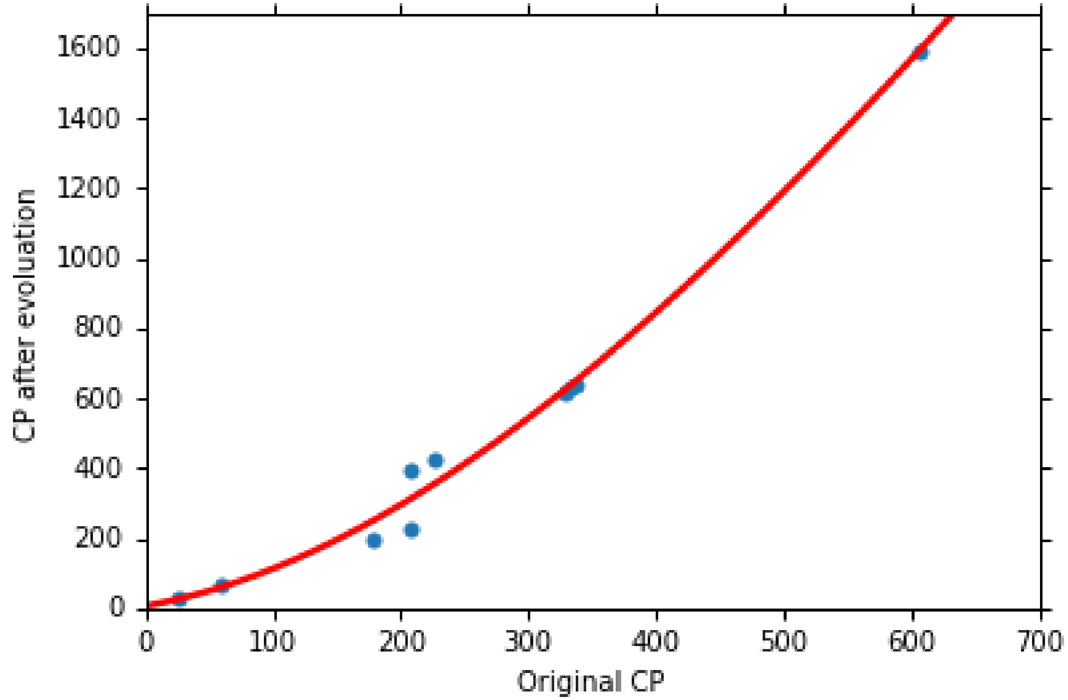
$w_3 = -1.8 \times 10^{-6}$

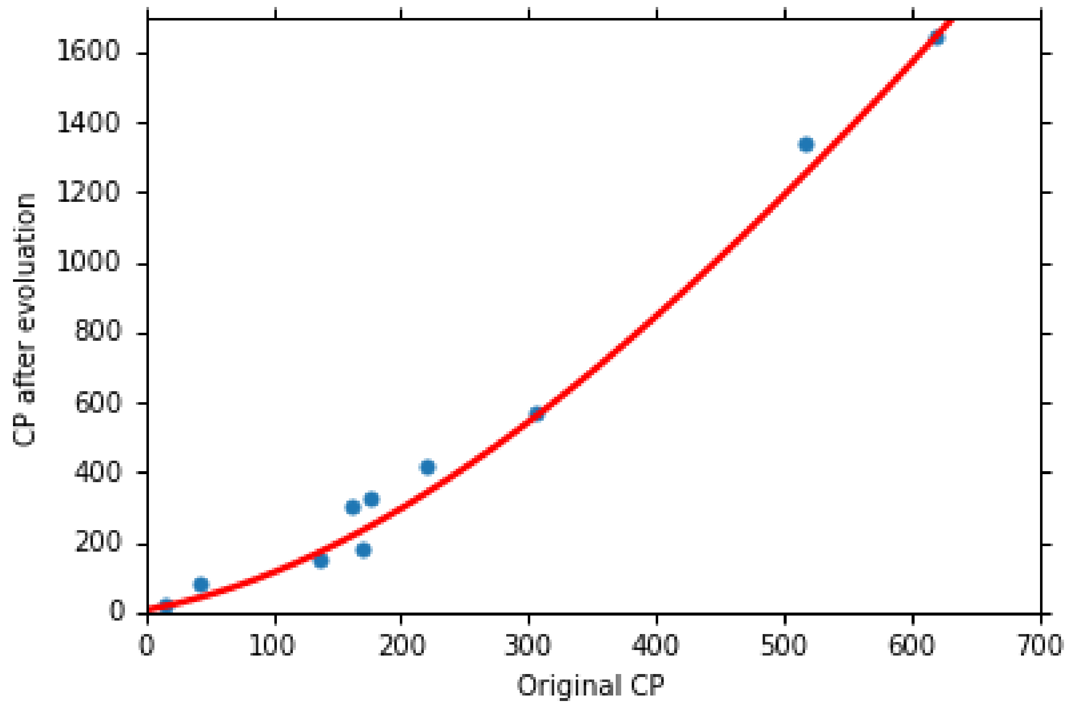
Average Error = 15.3

## *Testing:*

Average Error = 18.1

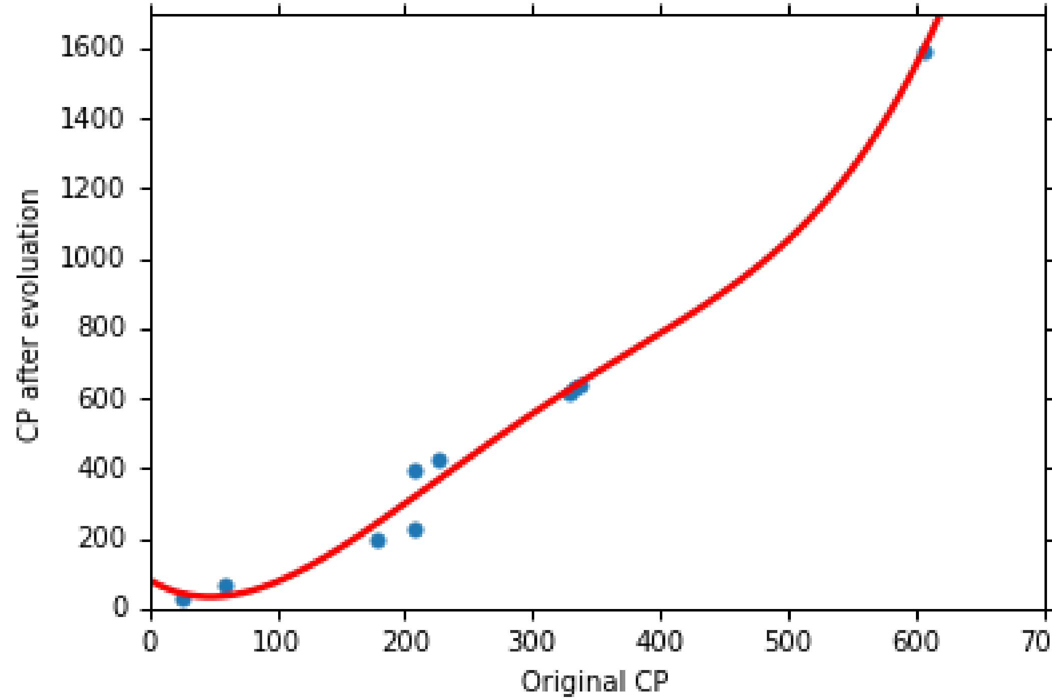
Slightly better.
How about more complex model?



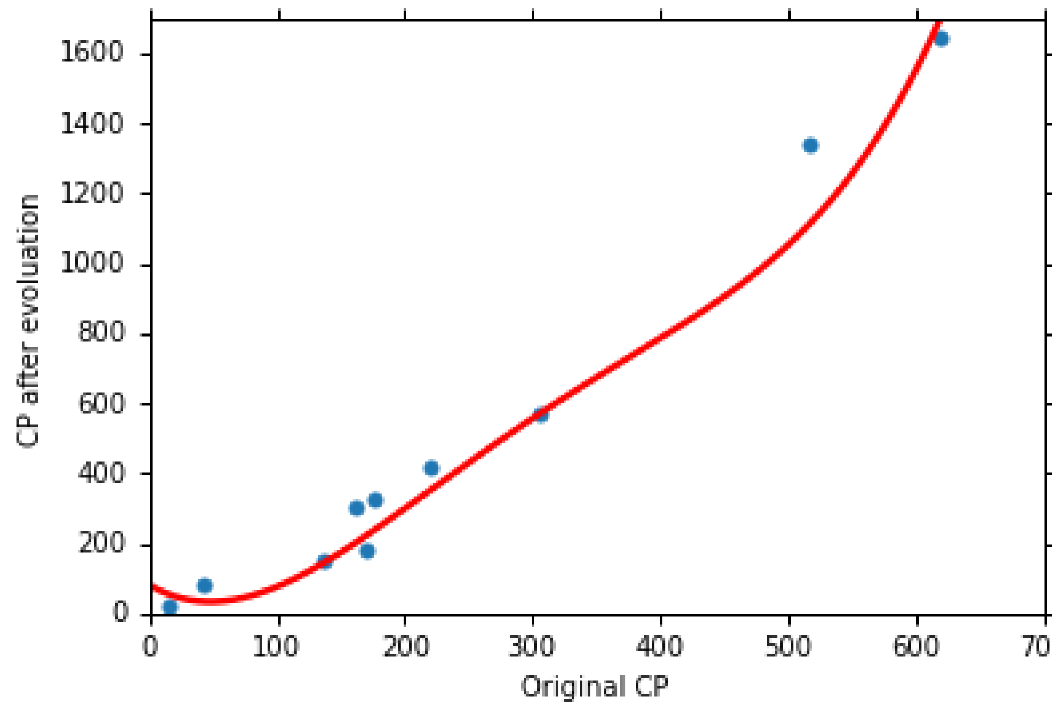
## ***Selecting another Model***

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




## ***Best Function***

Average Error = 14.9



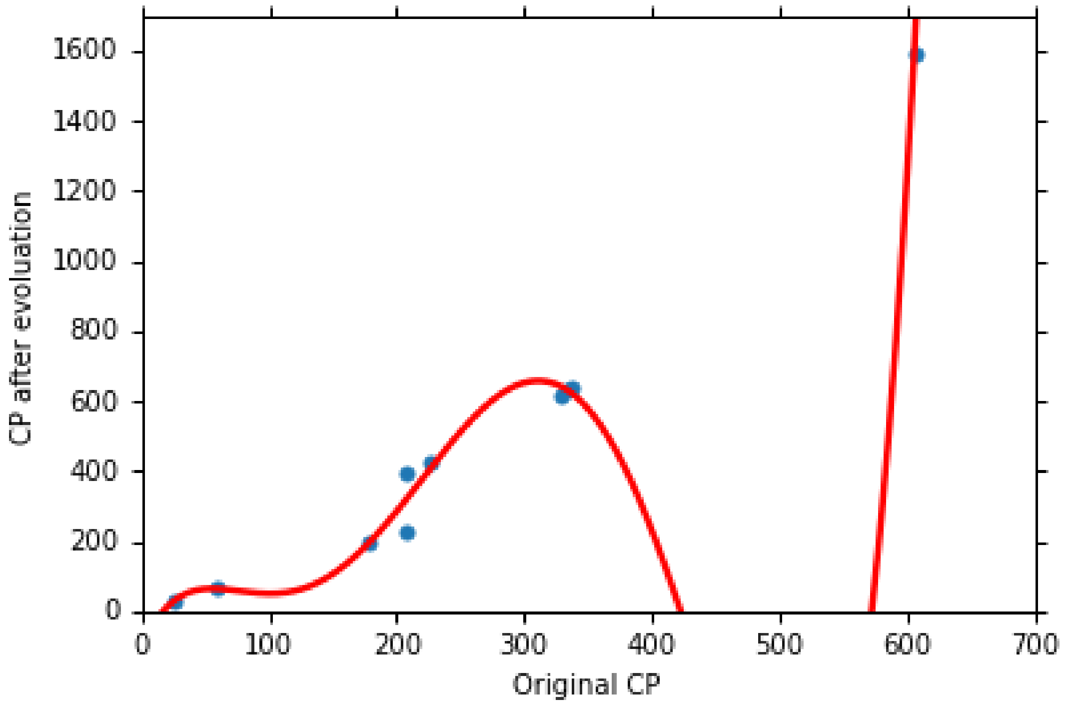

## ***Testing:***

Average Error = 28.8

The results become worse ...

## ***Selecting another Model***

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



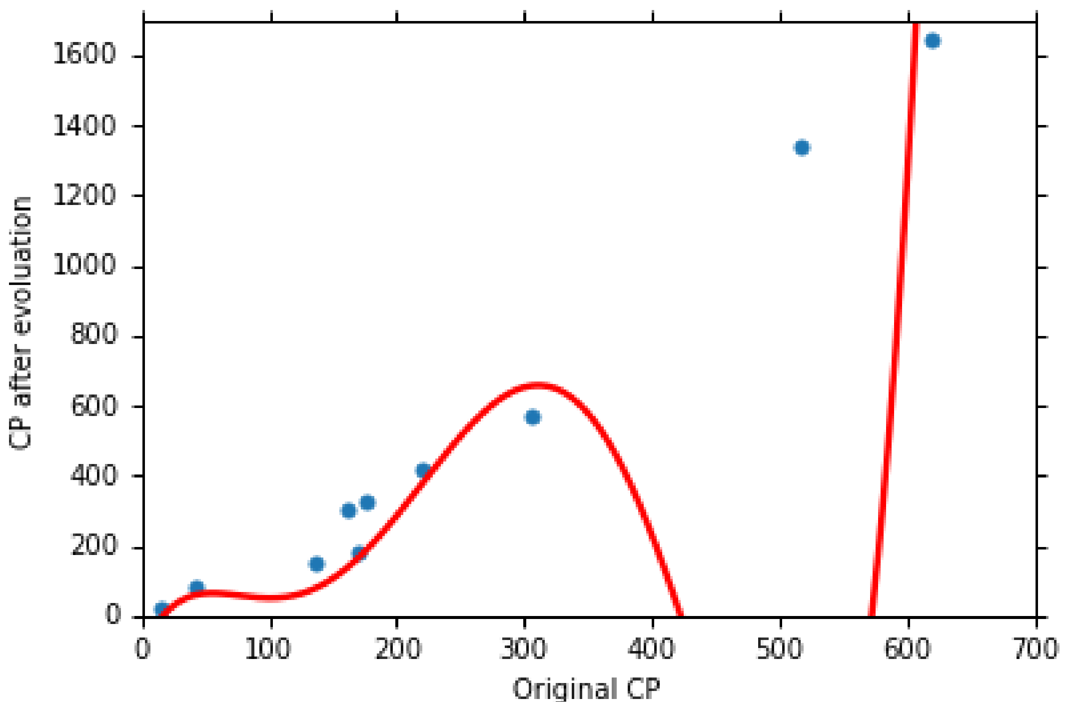

## ***Best Function***

Average Error = 12.8

## ***Testing:***

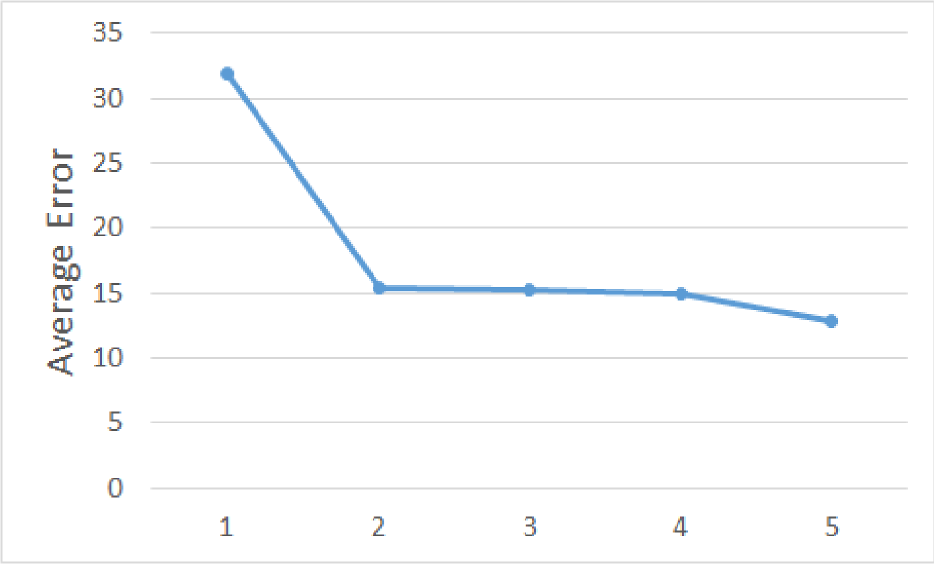
Average Error = 232.1

The results are so bad.

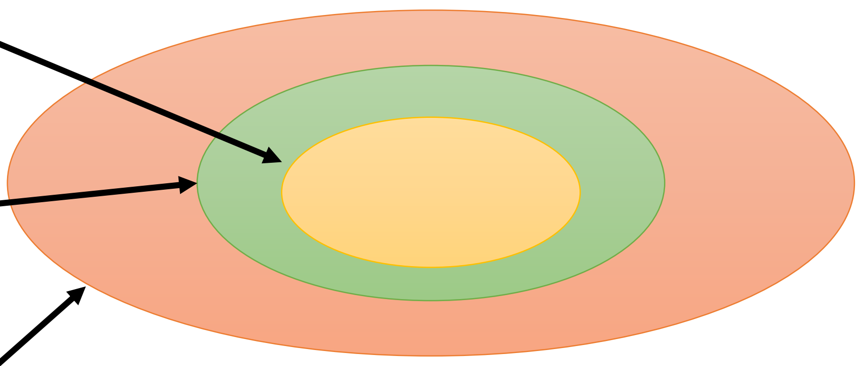


# Model Selection

Training Data



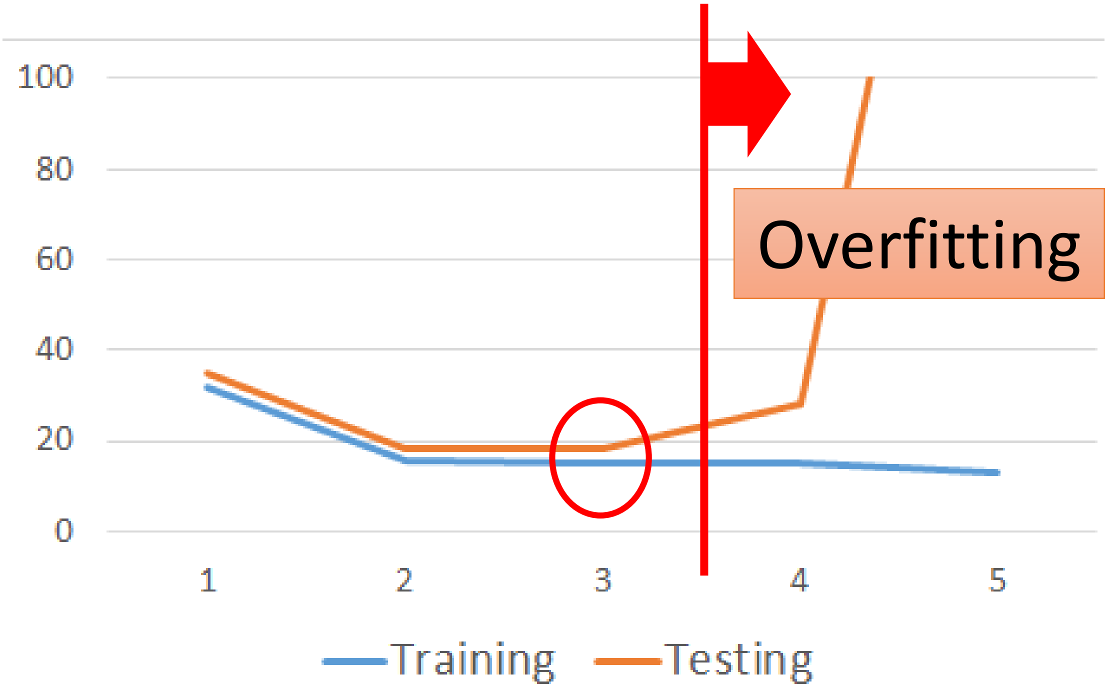

1. $y = b + w \cdot x_{cp}$
2. $y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2$
3. $y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2 + w_3 \cdot (x_{cp})^3$
4. $y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2 + w_3 \cdot (x_{cp})^3 + w_4 \cdot (x_{cp})^4$
5. $y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2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



A more complex model yields lower error on training data.

If we can truly find the best function

# Model Selection

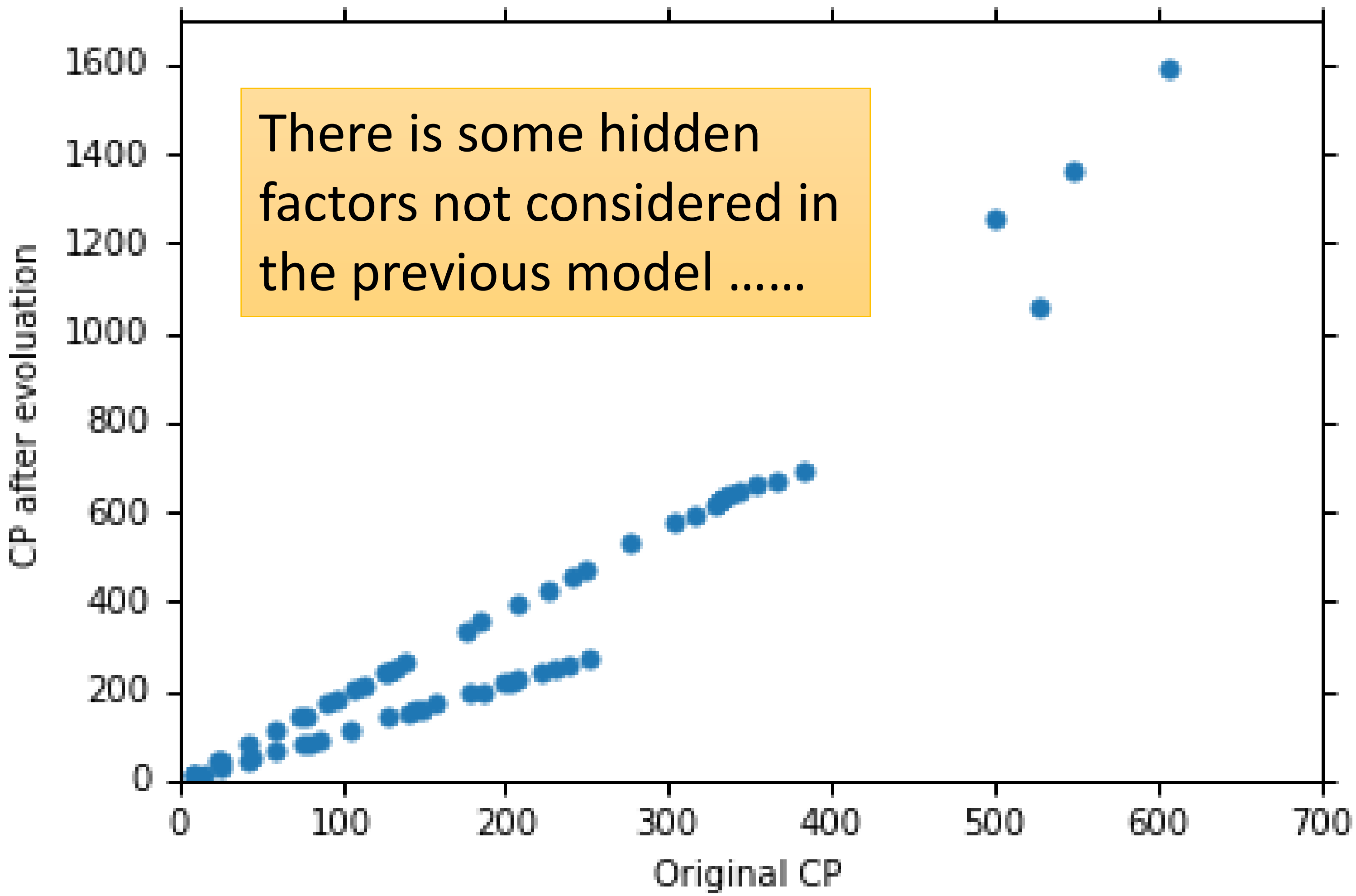



| | Training | Testing |
|---|---|---|
| 1 | 31.9 | 35.0 |
| 2 | 15.4 | 18.4 |
| 3 | 15.3 | 18.1 |
| 4 | 14.9 | 28.2 |
| 5 | 12.8 | 232.1 |

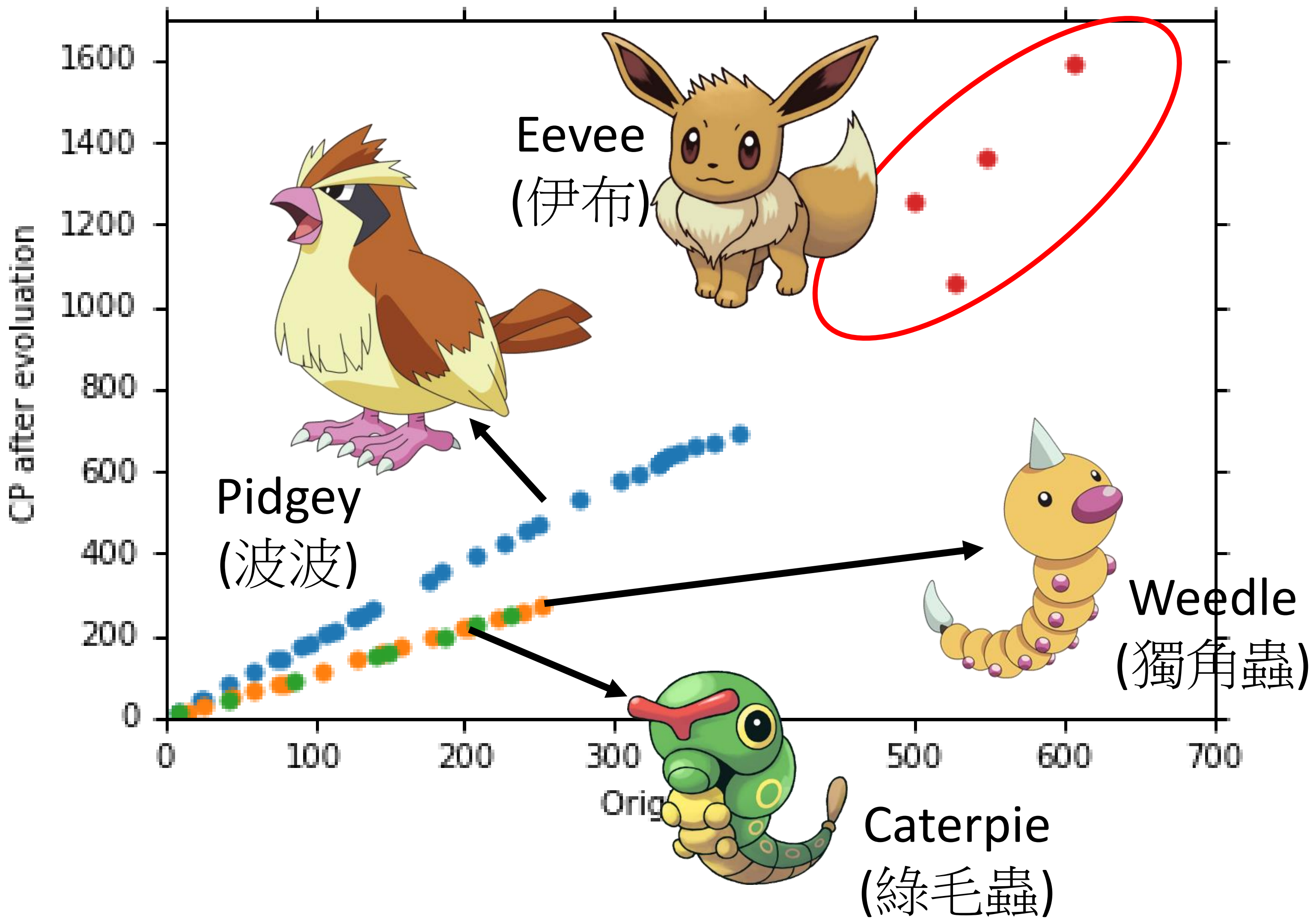
A more complex model does not always lead to better performance on ***testing data***.

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

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

Linear model?

$y = b_1 \cdot$ 1

$+w_1 \cdot$ 1 $x_{cp}$

$+b_2 \cdot$ 0

$+w_2 \cdot$ 0

$+b_3 \cdot$ 0

$+w_3 \cdot$ 0

$+b_4 \cdot$ 0

$+w_4 \cdot$ 0

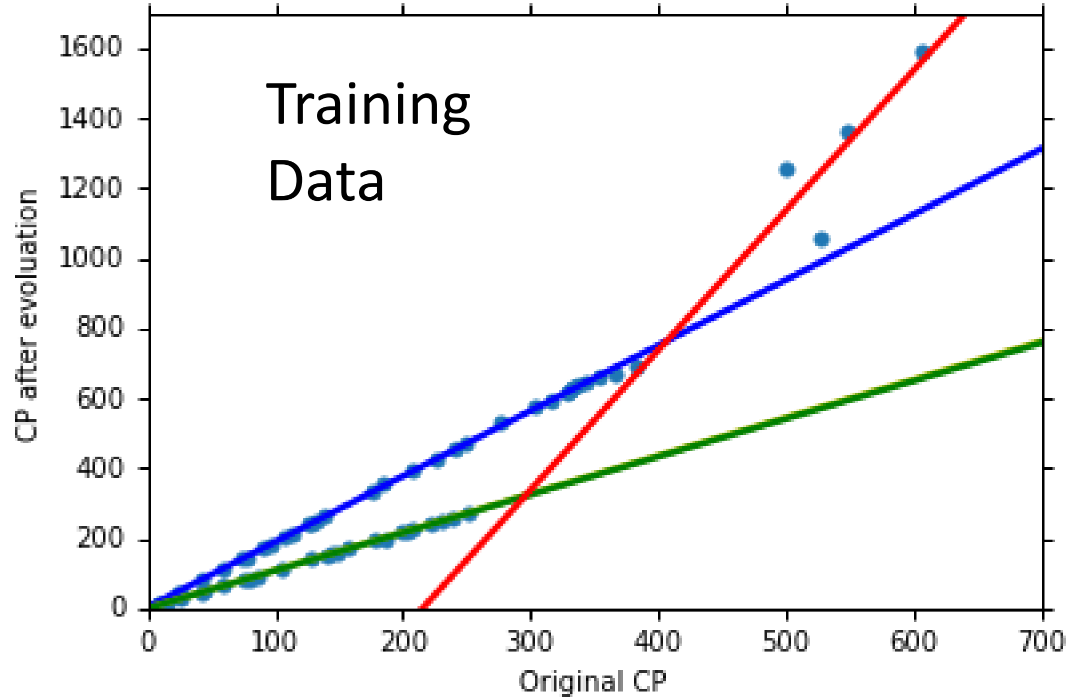
$\delta(x_s =$ Pidgey$)$

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

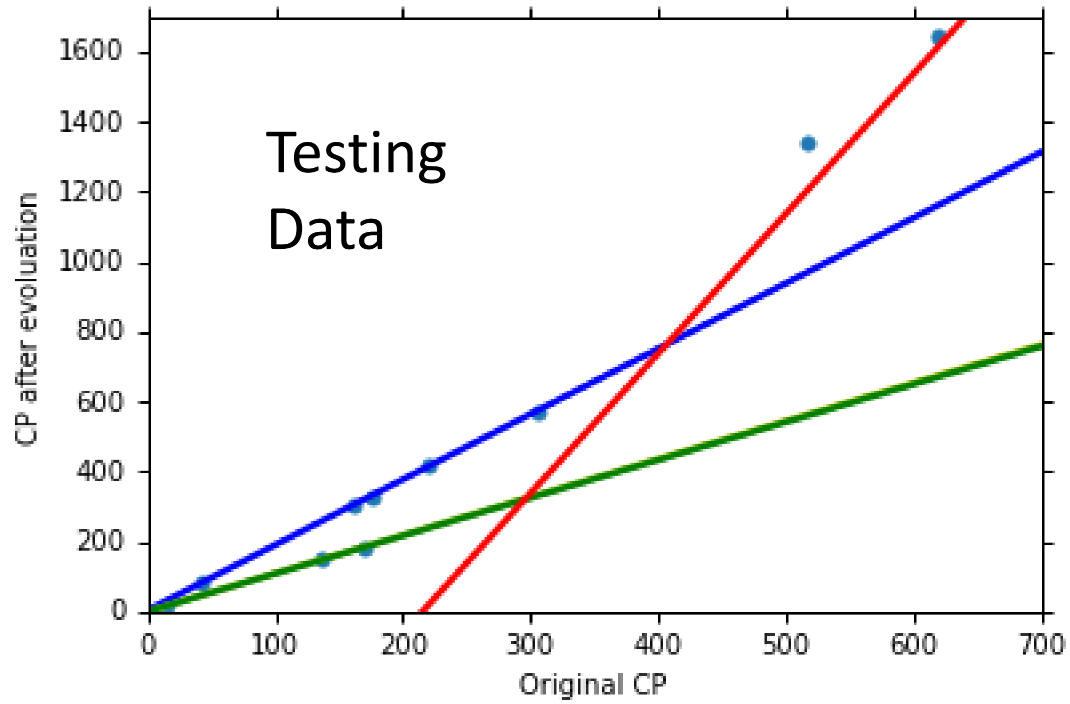
If $x_s =$ Pidgey

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

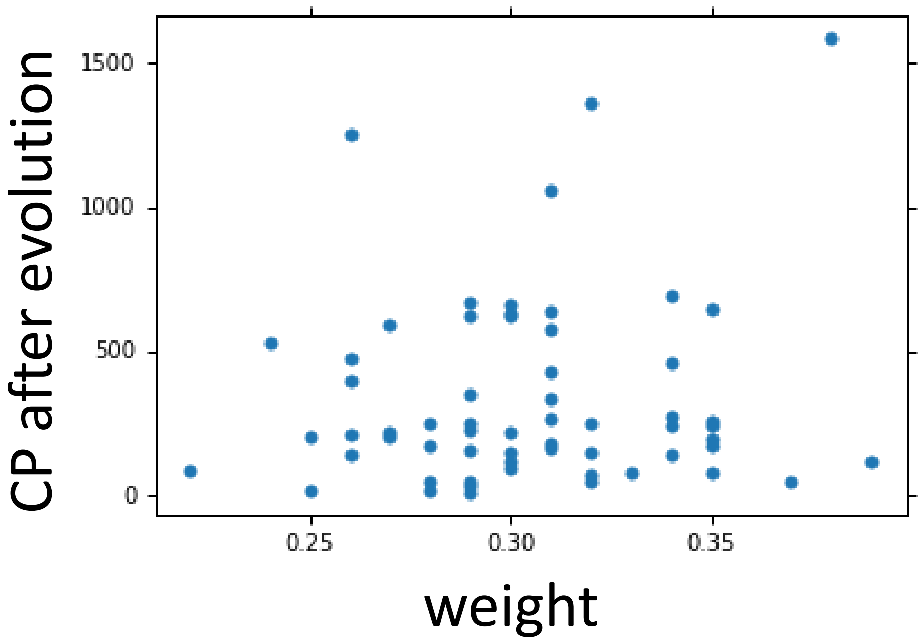
Average error
= 3.8

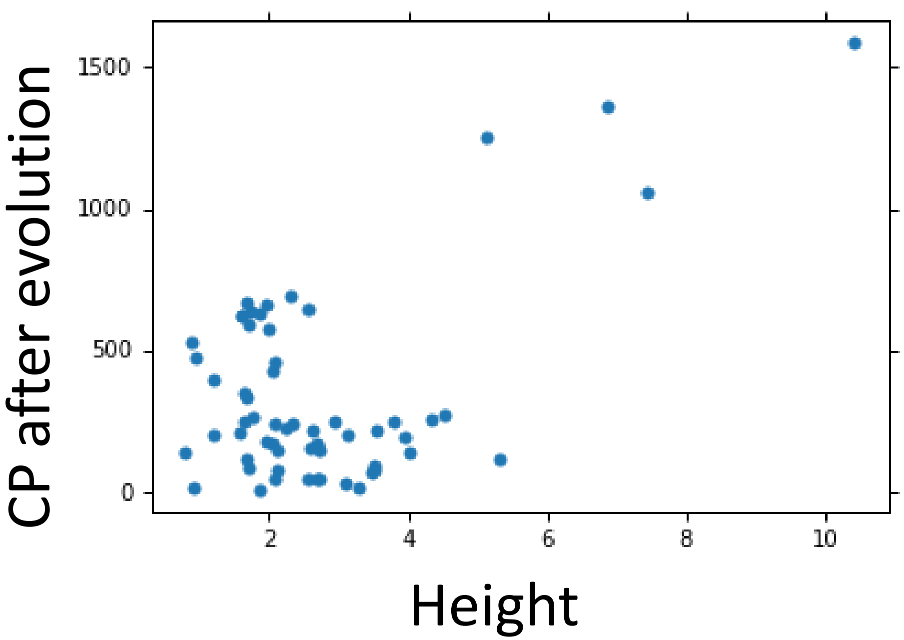


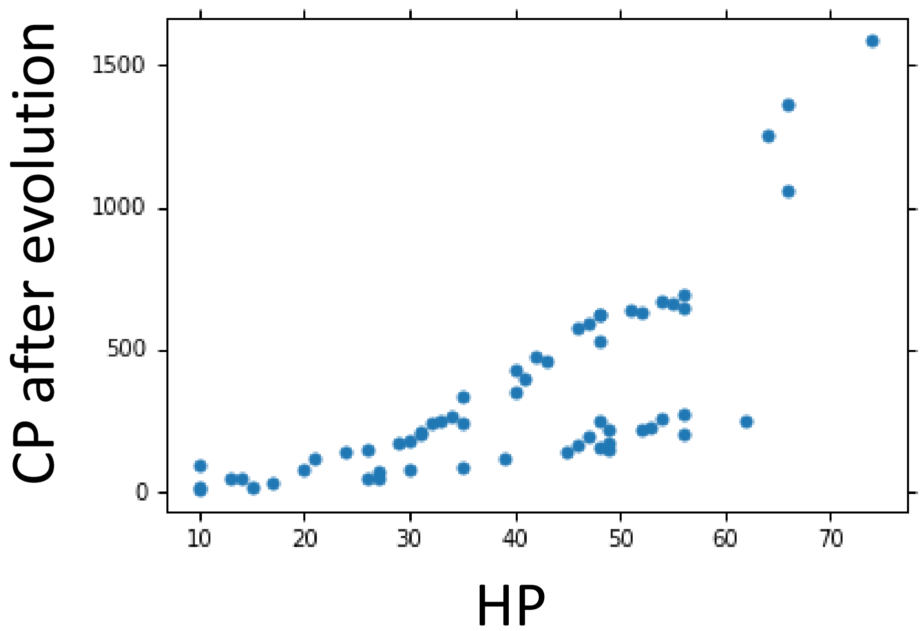

Average error
= 14.3



Are there any other hidden factors?



# Back to step 1: Redesign the Model Again

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

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

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

The functions with smaller $w_i$ are better

- Smaller $w_i$ means … smoother

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

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

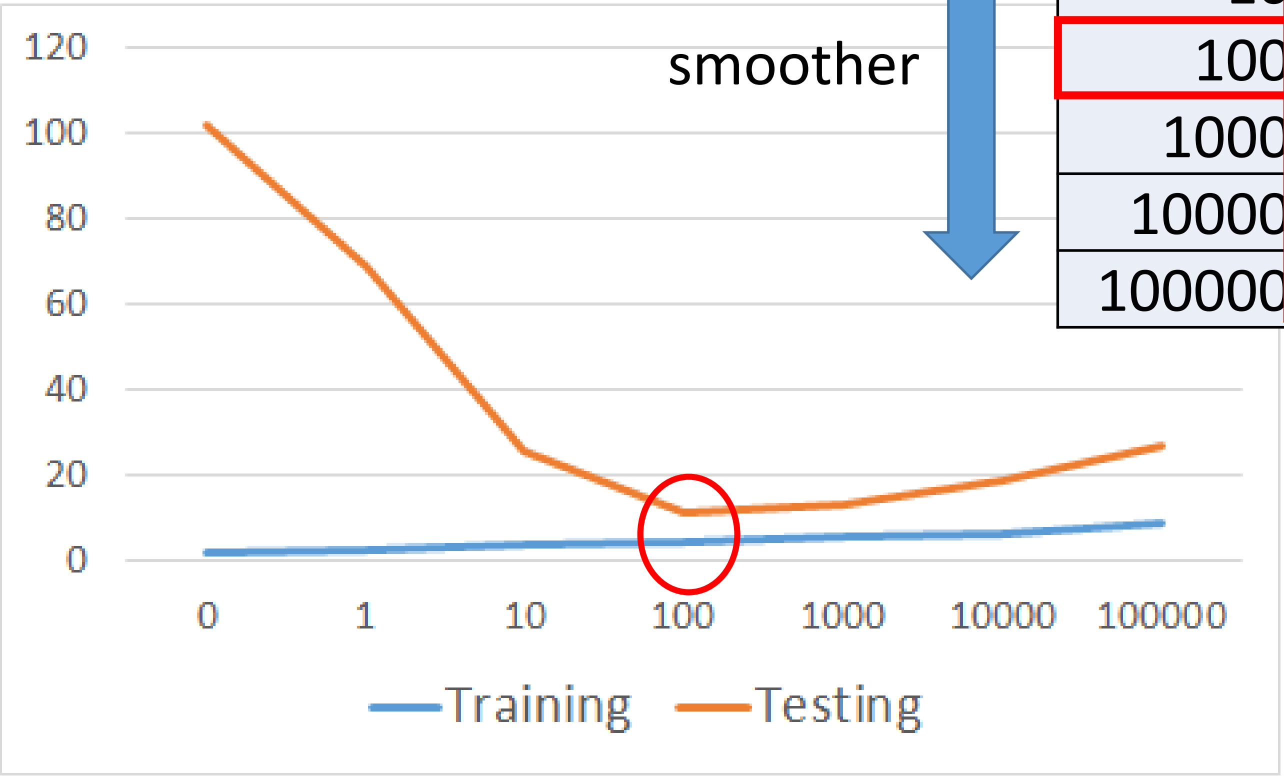
- We believe smoother function is more likely to be correct

Do you have to apply regularization on bias?

# Regularization

| $\lambda$ | 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 |

smoother

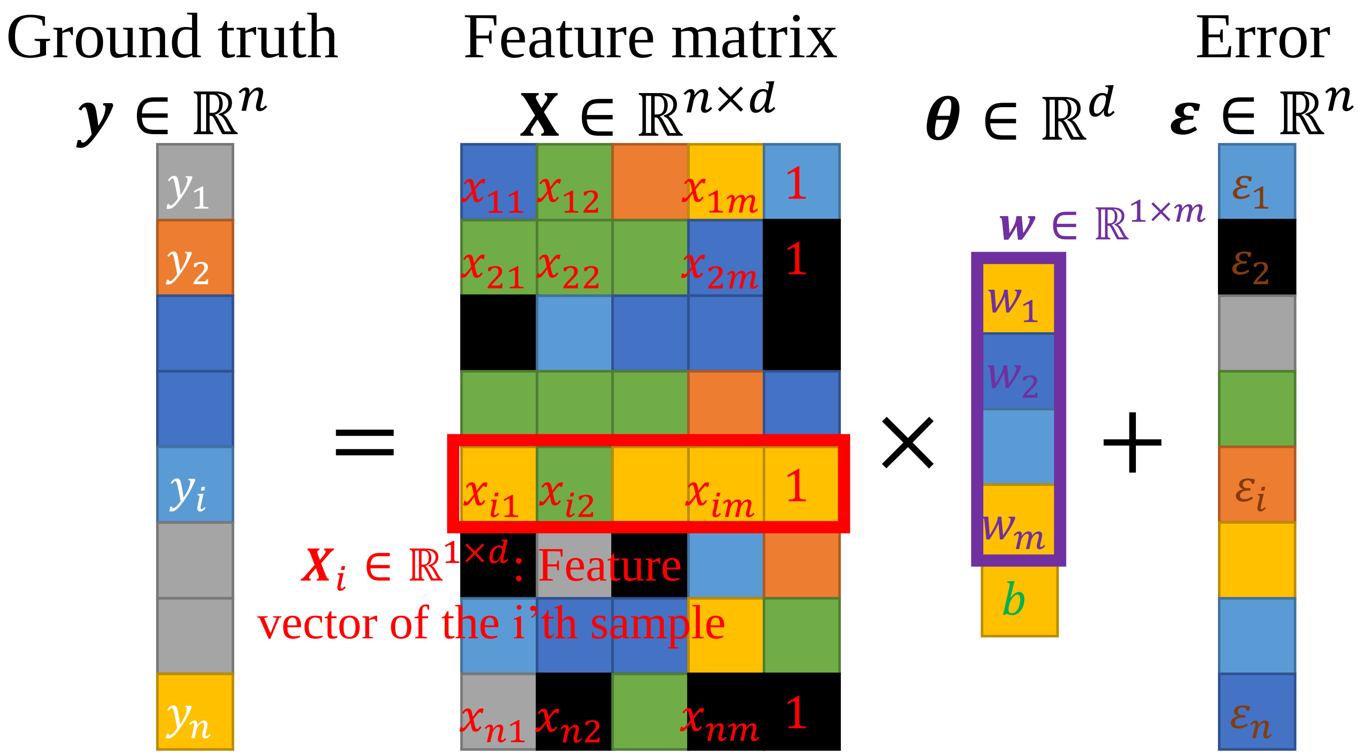



How smooth?

Select $\lambda$ obtaining the best model

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

# Matrix Form $\boldsymbol{y} = \mathbf{X}\boldsymbol{\theta} + \boldsymbol{\varepsilon}$




With bias term: $d = m + 1$ (denote $x_{i\, m+1} = 1$)
No bias term: $d = m$

Sum of Squared Error:
$$\varepsilon_1^2 + \cdots + \varepsilon_n^2 = \|\boldsymbol{\varepsilon}\|_2^2$$

Sum of Absolute Error:
$$|\varepsilon_1| + \cdots + |\varepsilon_n| = \|\boldsymbol{\varepsilon}\|_1$$

# Famous Regression Methods

$$\boldsymbol{y} = \mathbf{X}\boldsymbol{\theta} + \boldsymbol{\varepsilon} \Rightarrow \boldsymbol{\varepsilon} = \boldsymbol{y} - \mathbf{X}\boldsymbol{\theta}$$

$$\boldsymbol{\Omega} = diag(\omega_1, \dots, \omega_n) = \begin{bmatrix} \omega_1 & 0 & \cdots & 0 \\ 0 & \omega_2 & & \\ \vdots & & \ddots & \vdots \\ 0 & & \cdots & \omega_n \end{bmatrix}$$

| Regression methods | $L(\boldsymbol{\theta})$ | Matrix form |
|---|---|---|
| Least-squares linear regr. | $\sum_i (y_i - \boldsymbol{X}_\mathrm{i}\boldsymbol{\theta})^2$ | $\lVert\boldsymbol{y} - \mathbf{X}\boldsymbol{\theta}\rVert_2^2$ |
| Weighted least squares | $\sum_i \omega_i (y_i - \boldsymbol{X}_\mathrm{i}\boldsymbol{\theta})^2$ | $(\boldsymbol{y} - \boldsymbol{X\theta})^T \boldsymbol{\Omega} (\boldsymbol{y} - \boldsymbol{X\theta})$ |
| Ridge regr. | $\sum_i (y_i - \boldsymbol{X}_\mathrm{i}\boldsymbol{\theta})^2 + \lambda \sum_i w_i^2$ | $\lVert\boldsymbol{y} - \mathbf{X}\boldsymbol{\theta}\rVert_2^2 + \lambda \lVert\boldsymbol{w}\rVert_2^2$ |
| LASSO | $\sum_i (y_i - \boldsymbol{X}_\mathrm{i}\boldsymbol{\theta})^2 + \lambda \sum_i \lvert w_i \rvert$ | $\lVert\boldsymbol{y} - \mathbf{X}\boldsymbol{\theta}\rVert_2^2 + \lambda \lVert\boldsymbol{w}\rVert_1$ |
| Least absolute deviations | $\sum_i \lvert y_i - \boldsymbol{X}_\mathrm{i}\boldsymbol{\theta} \rvert$ | $\lVert\boldsymbol{y} - \mathbf{X}\boldsymbol{\theta}\rVert_1$ |
| Chebyshev criterion | $\max_i \lvert y_i - \boldsymbol{X}_\mathrm{i}\boldsymbol{\theta} \rvert$ | $\lVert\boldsymbol{y} - \mathbf{X}\boldsymbol{\theta}\rVert_\infty$ |

Quadratic cost; minimize w/calculus

Quadratic programming

Linear programming

We usually do not pose regularization on the bias term $b$
(a larger bias does not make the model more “non-smooth”)

# Minimize Quadratic Cost with Calculus

- For least-squares linear regr., write

$$L(\boldsymbol{\theta}) = \|\boldsymbol{y} - \mathbf{X}\boldsymbol{\theta}\|_2^2 = (\boldsymbol{y} - \mathbf{X}\boldsymbol{\theta})^T(\boldsymbol{y} - \mathbf{X}\boldsymbol{\theta})$$
$$= \boldsymbol{y}^T\boldsymbol{y} - \boldsymbol{\theta}^T\boldsymbol{X}^T\boldsymbol{y} - \boldsymbol{y}^T\mathbf{X}\boldsymbol{\theta} + \boldsymbol{\theta}^T\boldsymbol{X}^T\mathbf{X}\boldsymbol{\theta}$$

- Taking gradient

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = 2\boldsymbol{X}^T\mathbf{X}\boldsymbol{\theta} - 2\boldsymbol{X}^T\boldsymbol{y}$$

- Optimal solution occurs when gradient vanishes, namely

$$\boldsymbol{\theta}^* = (\boldsymbol{X}^T\mathbf{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

## For students obsessed with rigorous math:

**Q:** Does gradient equals zero implies minimal solution?

**A:** Counter-examples exist (think of functions like $\theta^3$).
A more rigorous proof is to write
$L(\boldsymbol{\theta}) = (\boldsymbol{\theta} - \boldsymbol{\theta}^*)^T(\boldsymbol{X}^T\mathbf{X})(\boldsymbol{\theta} - \boldsymbol{\theta}^*) + (\boldsymbol{y}^T\boldsymbol{y} - \boldsymbol{y}^T\mathbf{X}(\boldsymbol{X}^T\mathbf{X})^{-1}\boldsymbol{X}^T\boldsymbol{y})$
and recognize the fact that $\boldsymbol{X}^T\mathbf{X}$ is positive semi-definite.

**Q:** What if $\boldsymbol{X}^T\mathbf{X}$ is not invertible?

**A:** The optimal solution is not unique. The optimal solution with the minimum 2-norm is given by $\boldsymbol{\theta}^* = \boldsymbol{X}^\dagger\boldsymbol{y}$, where $\boldsymbol{X}^\dagger$ is the pseudo-inverse of $\boldsymbol{X}$. (Rather technically involved)

# Minimize Quadratic Cost with Calculus

| Regression methods | $L(\boldsymbol{\theta})$ | Matrix form | Optimal solution (For simplicity assume no bias, namely $\theta = w$) |
|---|---|---|---|
| Least-squares linear regr. | $\sum_i (y_i - \boldsymbol{X}_i\boldsymbol{\theta})^2$ | $\|\boldsymbol{y} - \mathbf{X}\boldsymbol{\theta}\|_2^2$ | $\boldsymbol{\theta}^* = (\boldsymbol{X}^T\mathbf{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$ (assume $\boldsymbol{X}^T\mathbf{X}$ is invertible) |
| Weighted least squares | $\sum_i \omega_i(y_i - \boldsymbol{X}_i\boldsymbol{\theta})^2$ | $(y_i - \boldsymbol{X}_i\boldsymbol{\theta})^T\boldsymbol{\Omega}(y_i - \boldsymbol{X}_i\boldsymbol{\theta})$ | $\boldsymbol{\theta}^* = (\boldsymbol{X}^T\boldsymbol{\Omega}\mathbf{X})^{-1}\boldsymbol{X}^T\boldsymbol{\Omega}\boldsymbol{y}$ (assume $\boldsymbol{X}^T\boldsymbol{\Omega}\mathbf{X}$ is invertible) |
| Ridge regr. | $\sum_i (y_i - \boldsymbol{X}_i\boldsymbol{\theta})^2 + \lambda\sum_i w_i^2$ | $\|\boldsymbol{y} - \mathbf{X}\boldsymbol{\theta}\|_2^2 + \lambda\|\boldsymbol{w}\|_2^2$ | $\boldsymbol{\theta}^* = (\boldsymbol{X}^T\mathbf{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y}$ |

Homework questions ☺

$$\boldsymbol{\Omega} = diag(\omega_1, \dots, \omega_n) = \begin{bmatrix} \omega_1 & 0 & \cdots & 0 \\ 0 & \omega_2 & & \\ \vdots & & \ddots & \vdots \\ 0 & & \cdots & \omega_n \end{bmatrix}$$

Besides intuitive "proof" (like setting derivatives to zero), you may challenge yourself to give rigorous proofs that really convince you, and/or cover the more general cases (say when $\boldsymbol{X}^T\boldsymbol{\Omega}\mathbf{X}$ is not invertible, considering the bias term $b$, etc.).