Chapter 3 Regression

Lecturer: Zhihua Jiang

Ex. of Regression Problems

- Stock Market Forecast

$f($  $) =$ Stock price at tomorrow

- Self-driving

$f($  $) =$ Steering wheel angle

- Recommendation system

$f($ User, Item $) =$ Buying possibility

Step 1: Model

$y = b + w \cdot x$

A set of functions

Model

$f_1, f_2 \cdots$

w and *b* are parameters (can be any value)

f_1: $y = 10.0 + 9.0 \cdot x$

f_2: $y = 9.8 + 9.2 \cdot x$

f_3: $y = - 0.8 - 1.2 \cdot x$

…… infinite

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

Step 2: Goodness of Function

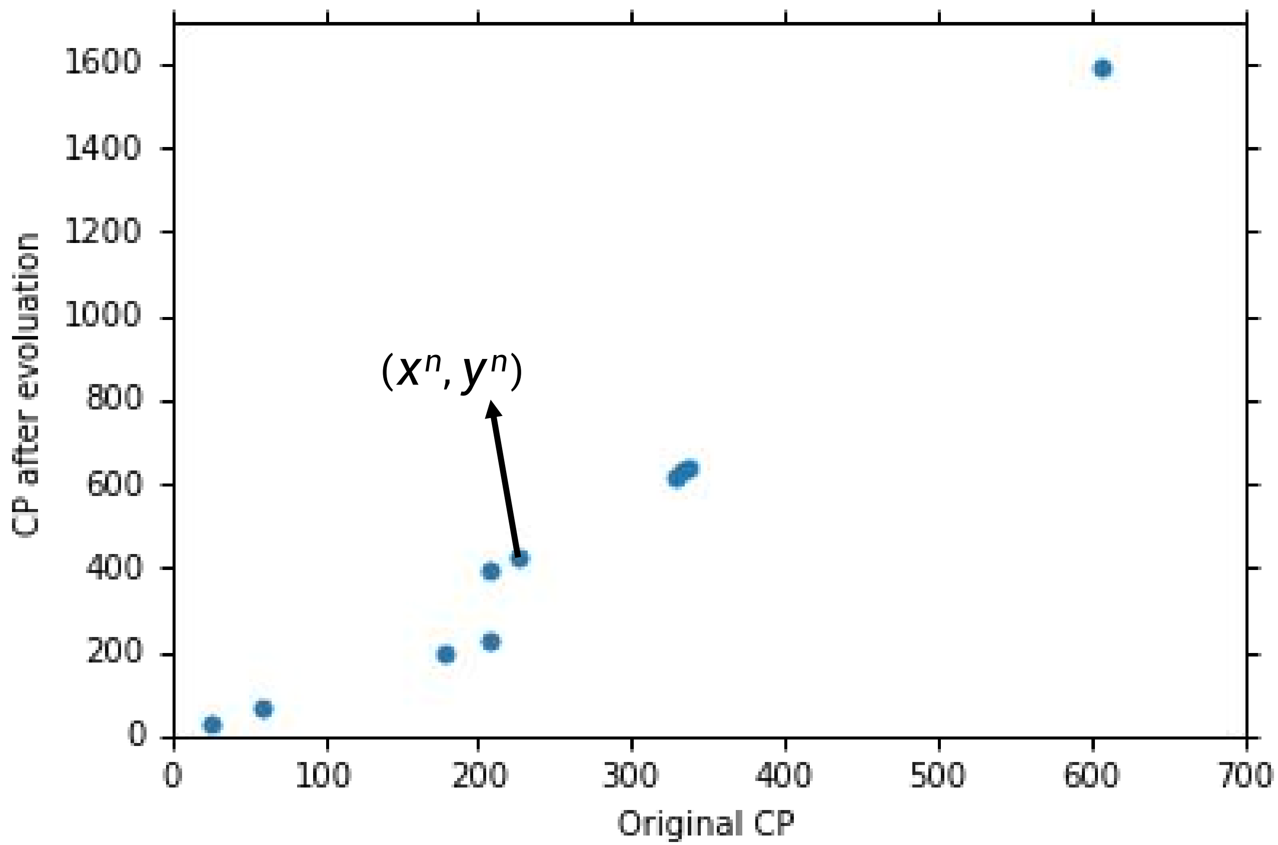
Training Data: 10 examples

(x^1, y^1)

(x^2, y^2)

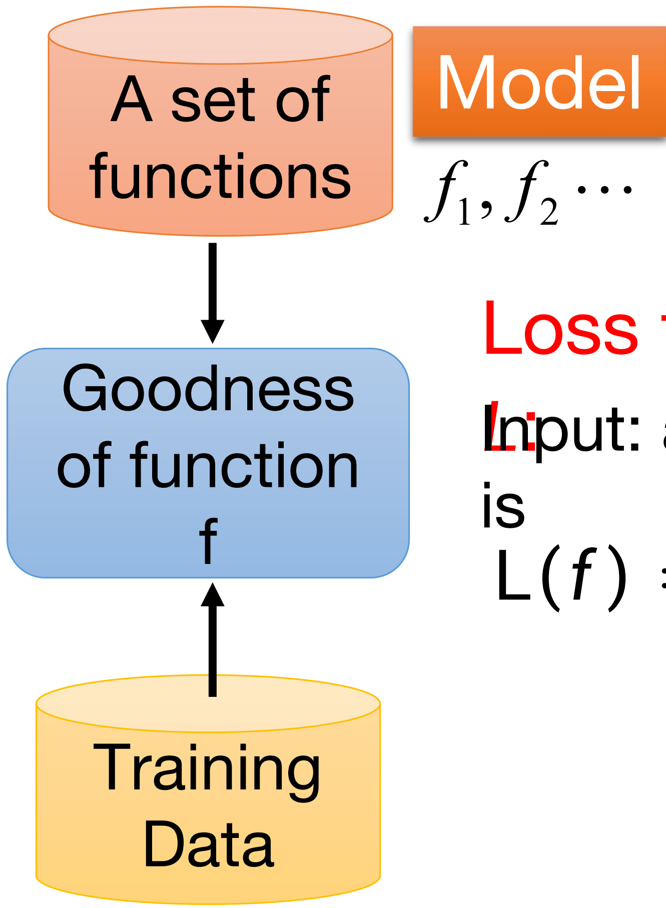
⋮

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



Step 2: Goodness of Function

$y = b + w \cdot x$



Loss function

L: Input: a function, output: how bad it is

$$L(f) = L(w, b)$$

$$= \sum_{n=1}^{10} \boxed{\left(y^n - (b + w \cdot x n)\right)^2}$$

Step 3: Best Function

A set of functions

Model

$f_1, f_2 \cdots$

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

Goodness of function f

Training Data

Pick the "Best" Function

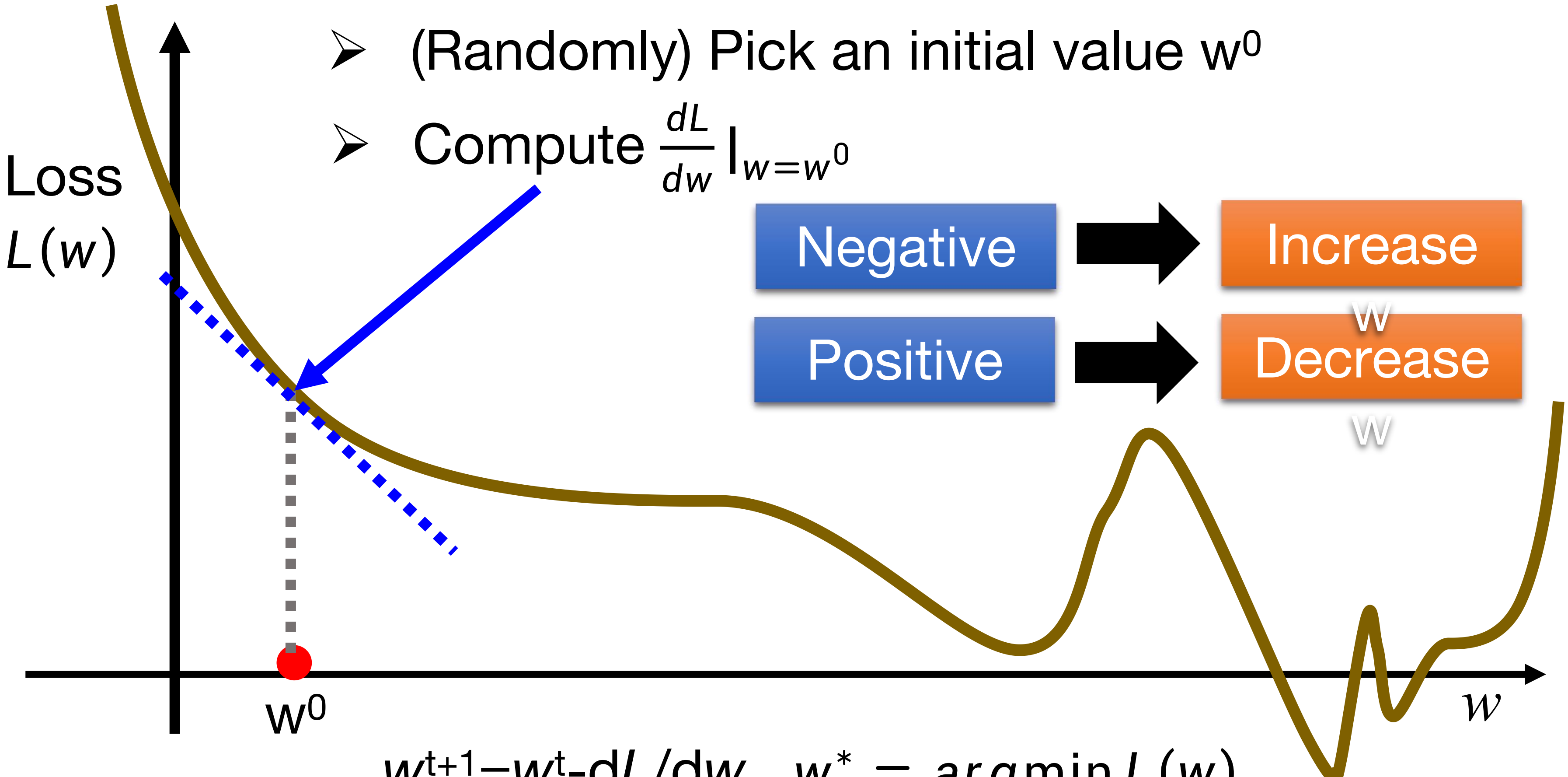
$$f^* = arg \min_f L(f)$$

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

Gradient Descent

Gradient Descent

- Consider loss function $L(w)$ with one parameter w:
  - ➢ (Randomly) Pick an initial value w^0
  - ➢ Compute $\frac{dL}{dw}|_{w=w^0}$



$w^{t+1}=w^t-dL/dw$ $\quad w^* = \arg\min_w L(w)$

Review: Gradient Descent

- In step 3, we have to solve the following optimization problem:

$$\theta^* = \arg\min_{\theta} L(\theta)$$ L: loss function θ: parameters

Suppose that θ has two variables $\{\theta_1, \theta_2\}$

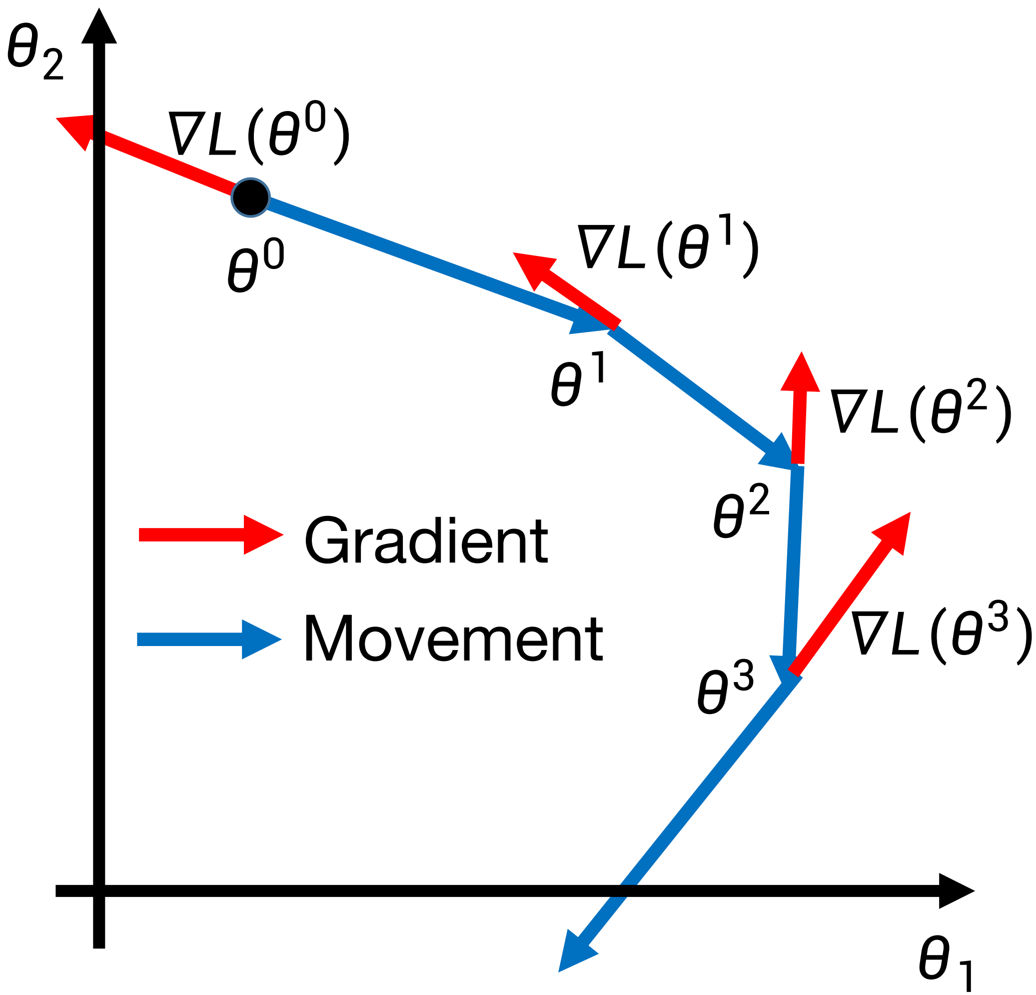
Randomly start at $\theta^0 = \begin{bmatrix} \theta_1^0 \\ \theta_2^0 \end{bmatrix}$

$$\nabla L(\theta) = \begin{bmatrix} \partial L(\theta_1)/\partial\theta_1 \\ \partial L(\theta_2)/\partial\theta_2 \end{bmatrix}$$

$$\begin{bmatrix} \theta_1^1 \\ \theta_2^1 \end{bmatrix} = \begin{bmatrix} \theta_1^0 \\ \theta_2^0 \end{bmatrix} - \eta \begin{bmatrix} \partial L(\theta_1^0)/\partial\theta_1 \\ \partial L(\theta_2^0)/\partial\theta_2 \end{bmatrix} \Rightarrow \theta^1 = \theta^0 - \eta \nabla L(\theta^0)$$

$$\begin{bmatrix} \theta_1^2 \\ \theta_2^2 \end{bmatrix} = \begin{bmatrix} \theta_1^1 \\ \theta_2^1 \end{bmatrix} - \eta \begin{bmatrix} \partial L(\theta_1^1)/\partial\theta_1 \\ \partial L(\theta_2^1)/\partial\theta_2 \end{bmatrix} \Rightarrow \theta^2 = \theta^1 - \eta \nabla L(\theta^1)$$

Review: Gradient Descent



Start at position θ^0

Compute gradient at θ^0

Move to $\theta^1 = \theta^0 - \eta \nabla L(\theta^0)$

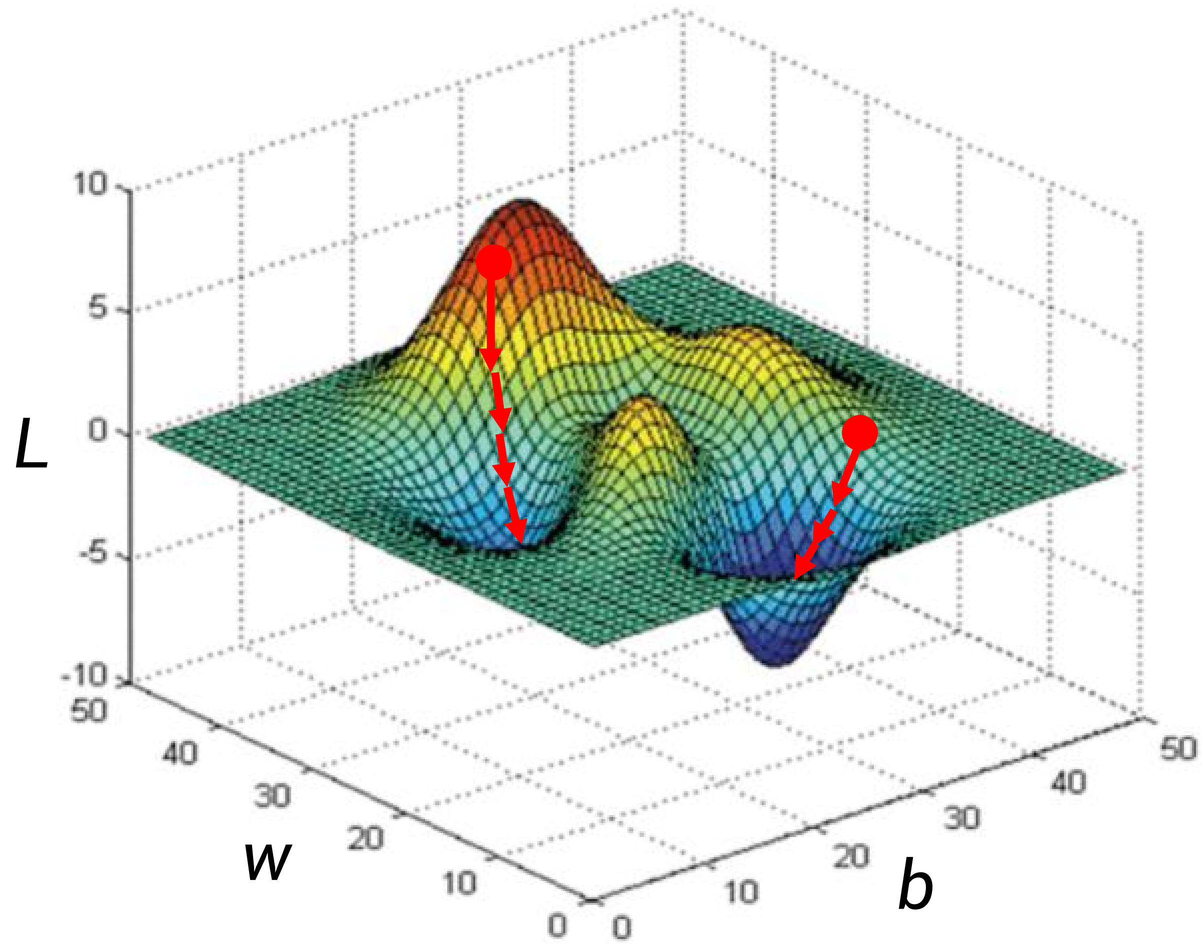
Compute gradient at θ^1

Move to $\theta^2 = \theta^1 - \eta \nabla L(\theta^1)$

⋮ ⋮

Step 3: Gradient Descent

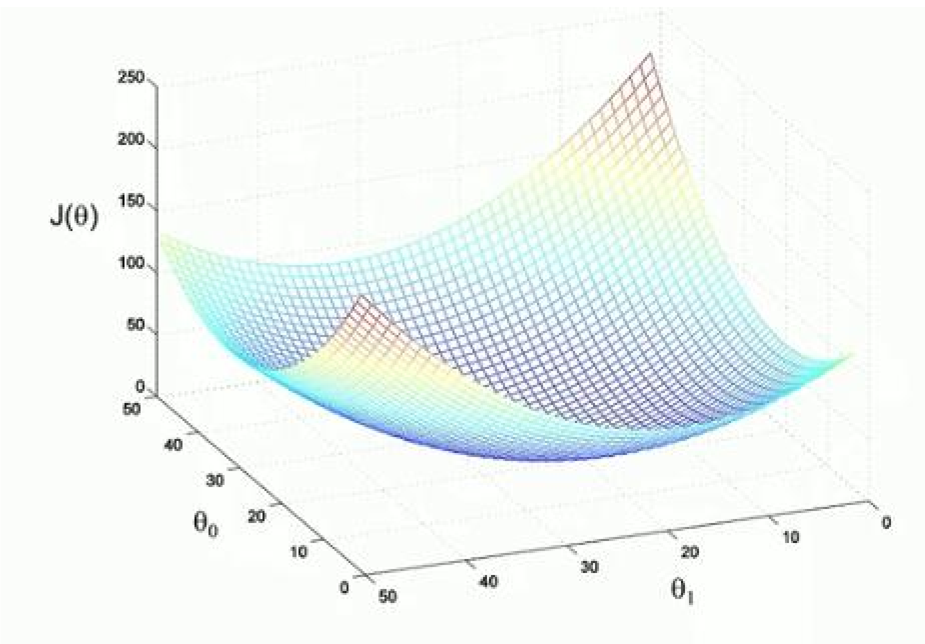
- Worry for local optimum?

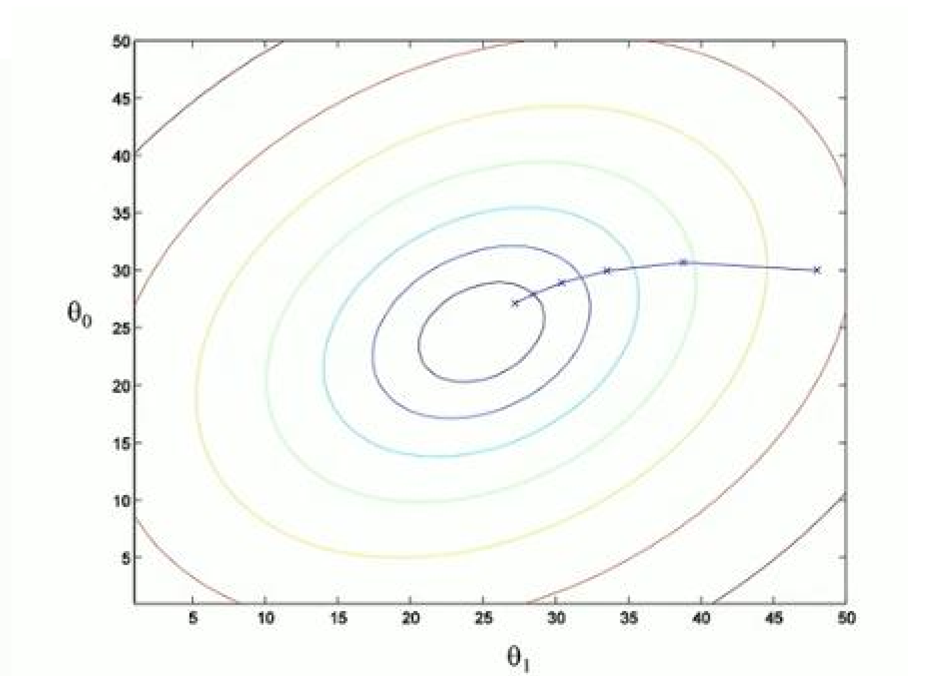


Step 3: Gradient Descent

Don’t worry. In linear regression, the loss function *L* is convex.

No local optimal.





Step 3: Gradient Descent

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

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

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

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

How’s the results?

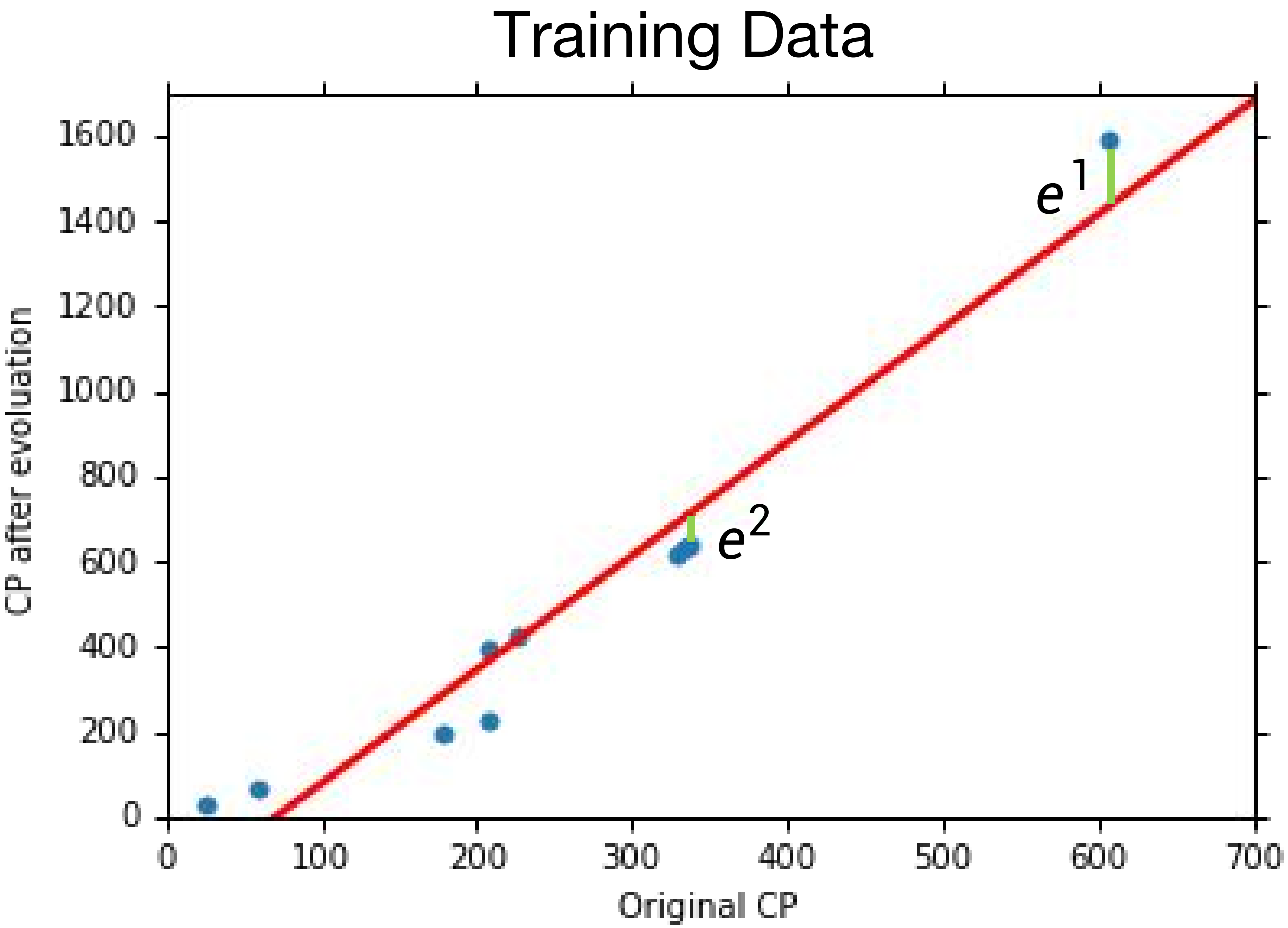
$y = b + w \cdot x$

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)

$y = b + w \cdot x$

b = -188.4

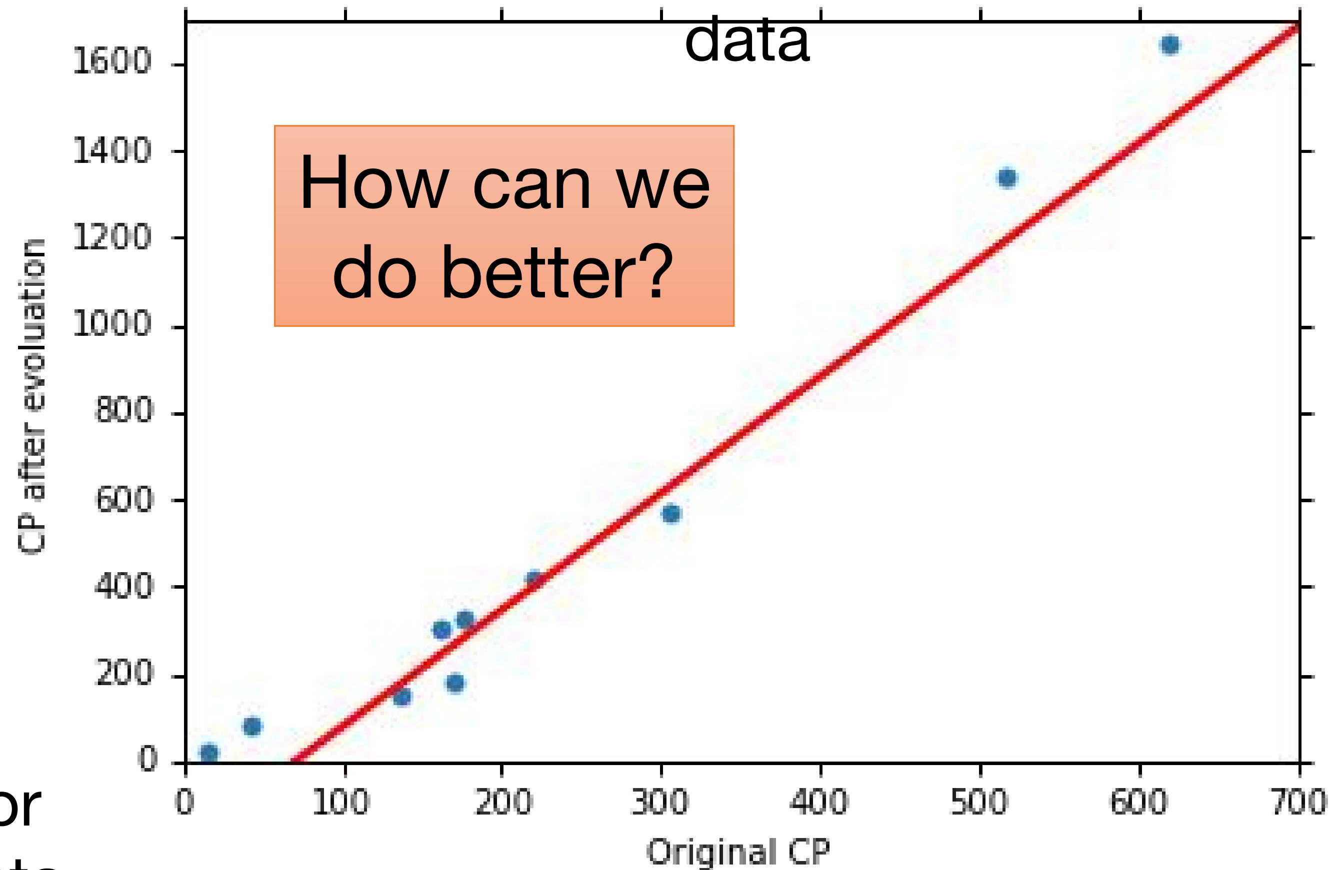
w = 2.7

Average Error on Testing Data

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

> Average Error on Training Data

Another 10 examples as testing data



How can we do better?

Selecting another Model

$$y = b + w_1 \cdot x + w_2 \cdot x^2$$

Best Function

$b = -10.3$

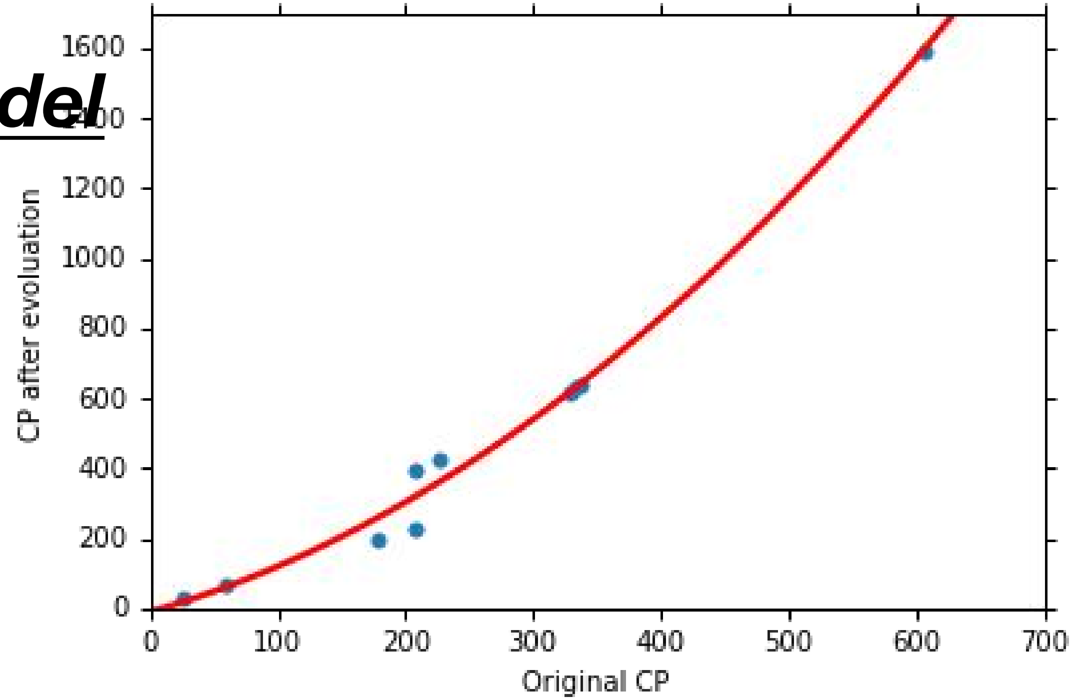
$w_1 = 1.0$, $w_2 = 2.7 \times 10^{-3}$

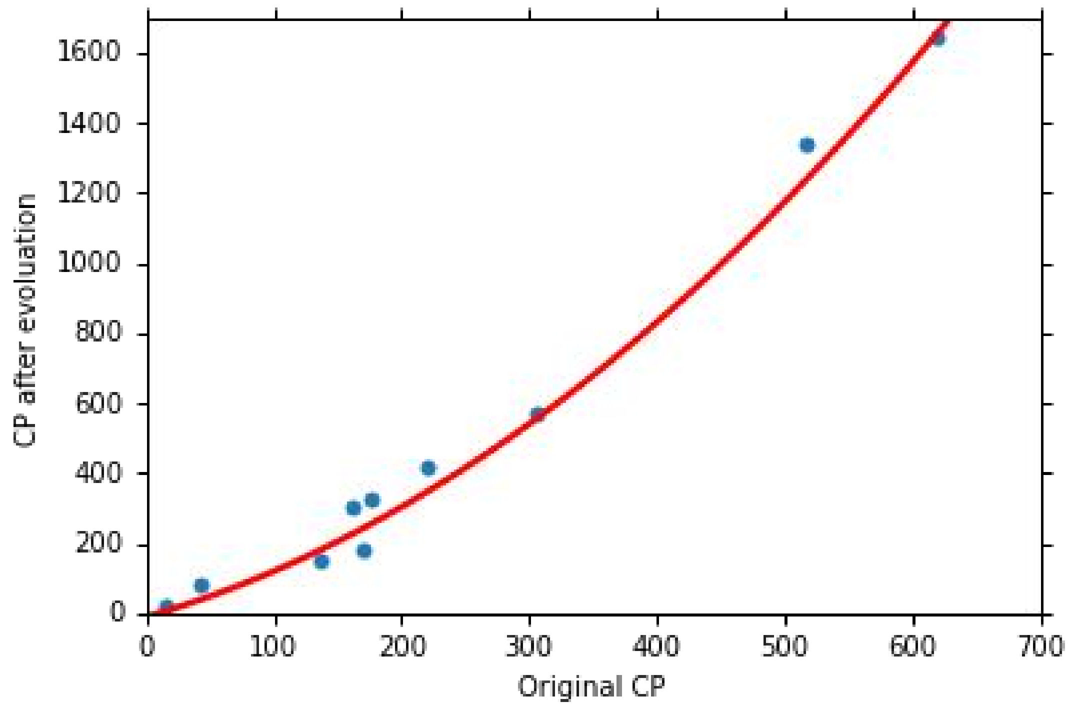
Average Error = 15.4

Testing:

Average Error = 18.4

Better! Could it be even better?





Selecting another Model

$$y = b + w_1 \cdot x + w_2 \cdot x^2 + w_3 \cdot x^3$$

Best Function

$b = 6.4$, $w_1 = 0.66$

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

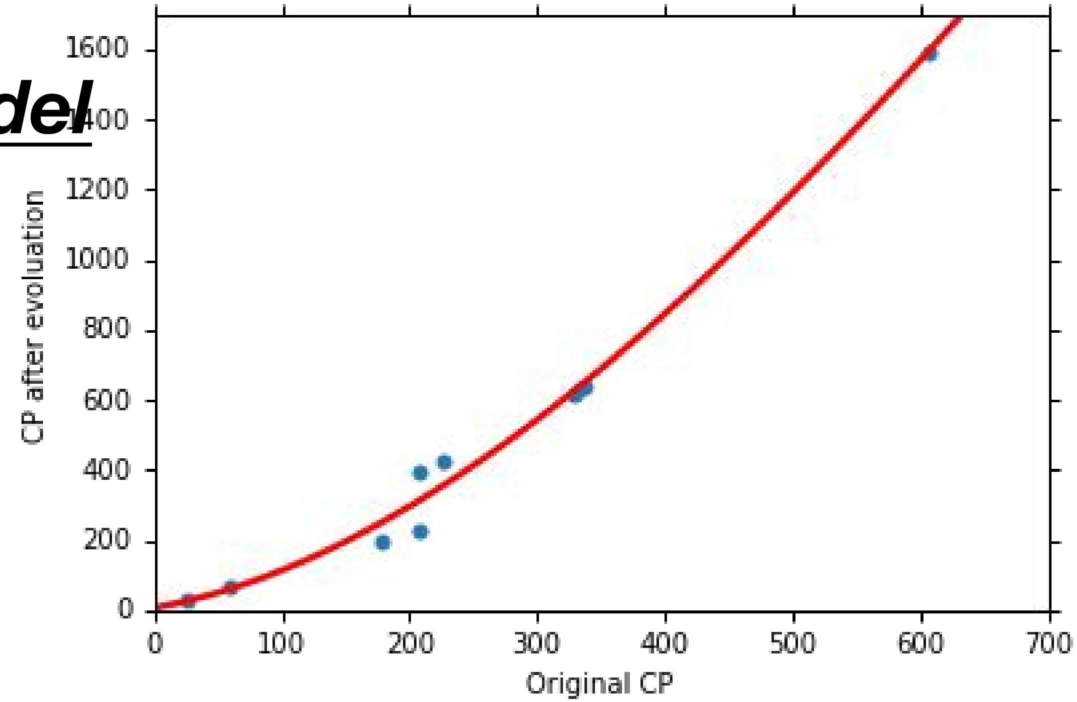
$w_3 = -1.8 \times 10^{-6}$

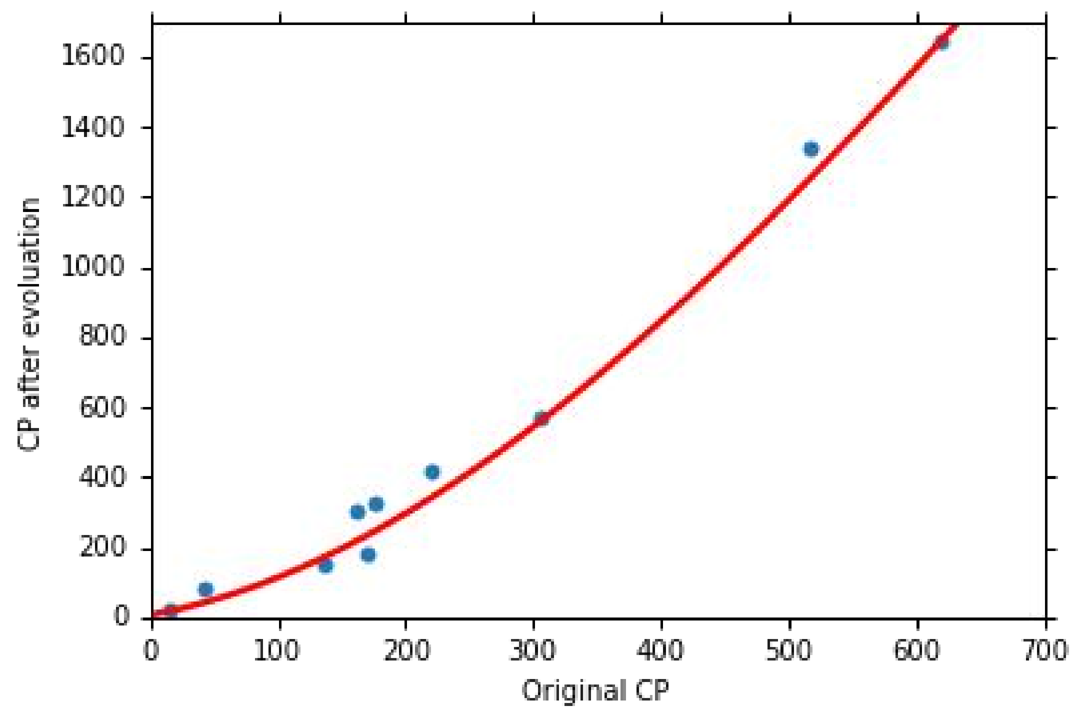
Average Error = 15.3

Testing:

Average Error = 18.1

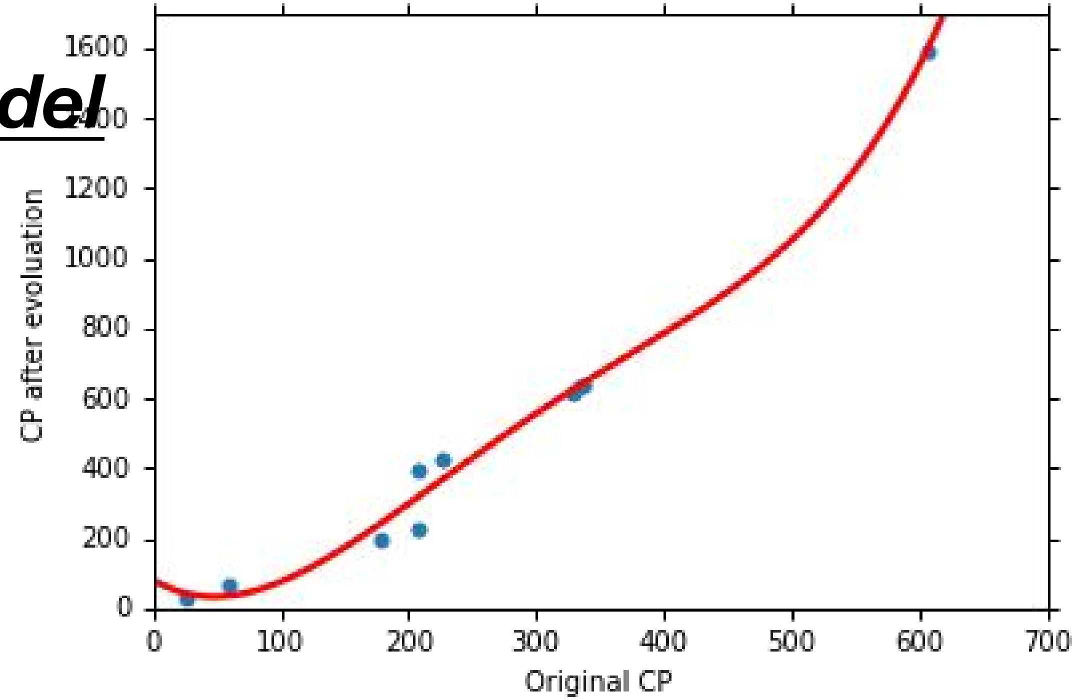
Slightly better.
How about more complex model?





Selecting another Model

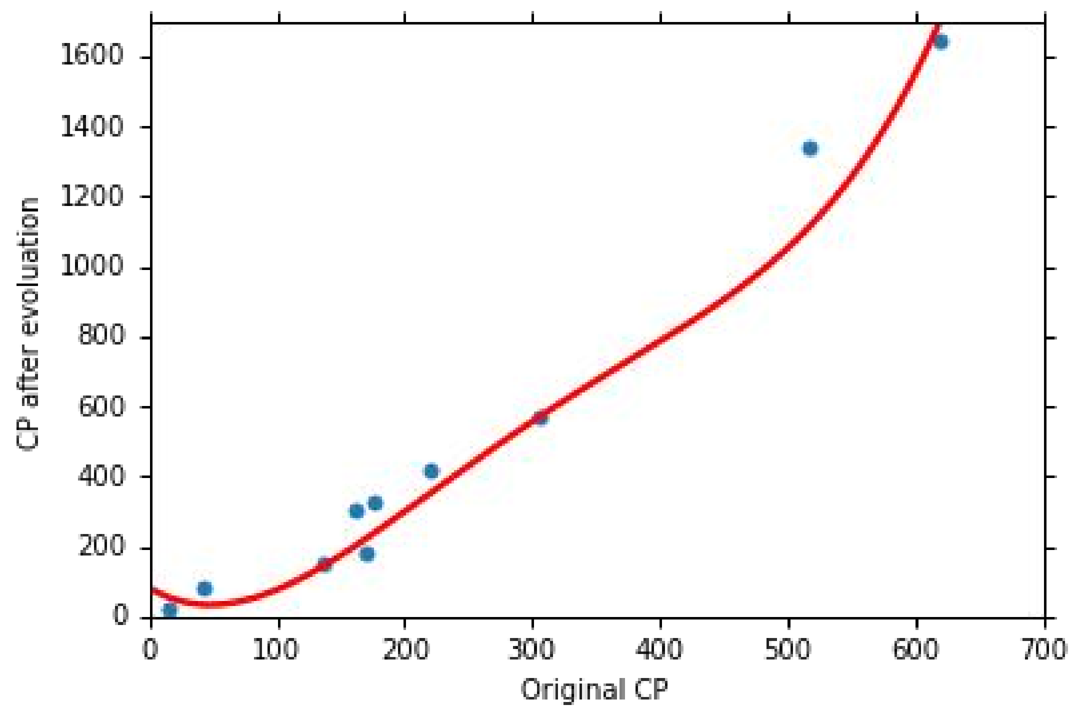
$$y = b + w_1 \cdot x + w_2 \cdot x^2 + w_3 \cdot x^3 + w_4 \cdot x^4$$



Best Function

Average Error = 14.9

Testing:

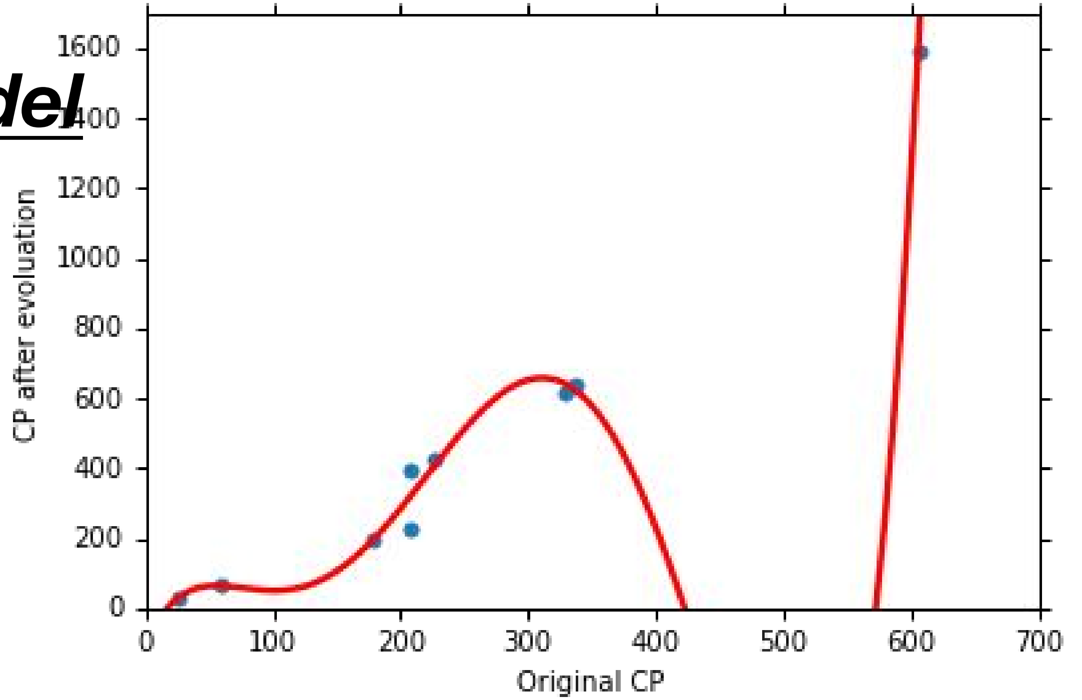


Average Error = 28.8

The results become worse ...

Selecting another Model

$$y = b + w_1 \cdot x + w_2 \cdot x^2 + w_3 \cdot x^3 + w_4 \cdot x^4 + w_5 \cdot x^5$$



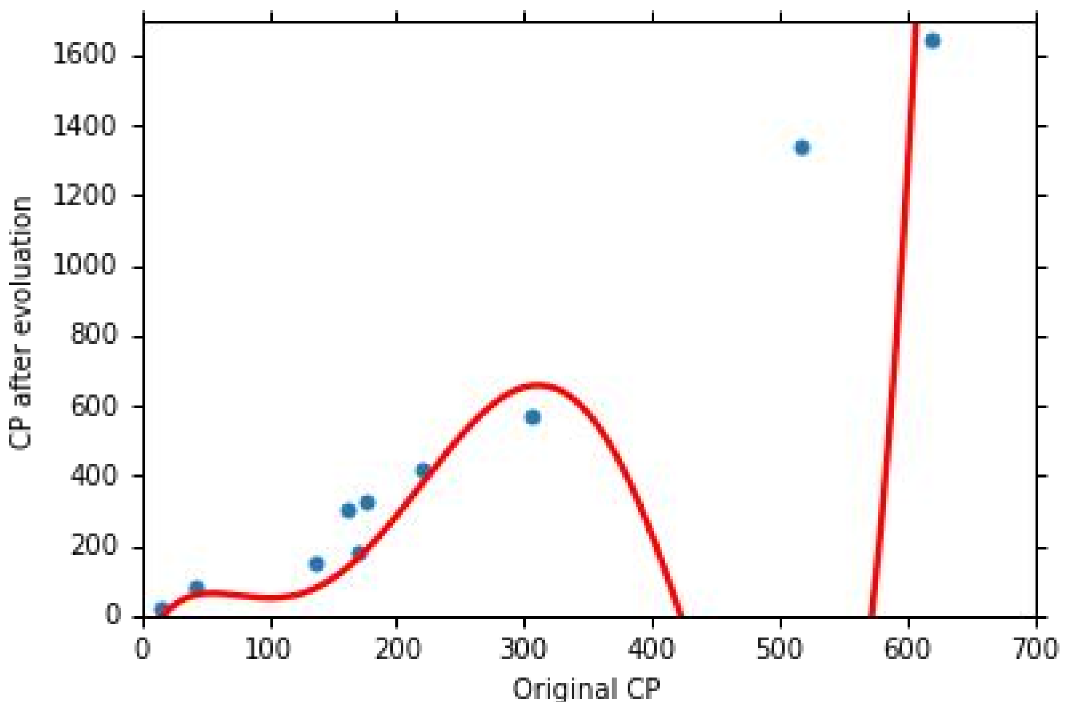
Best Function

Average Error = 12.8

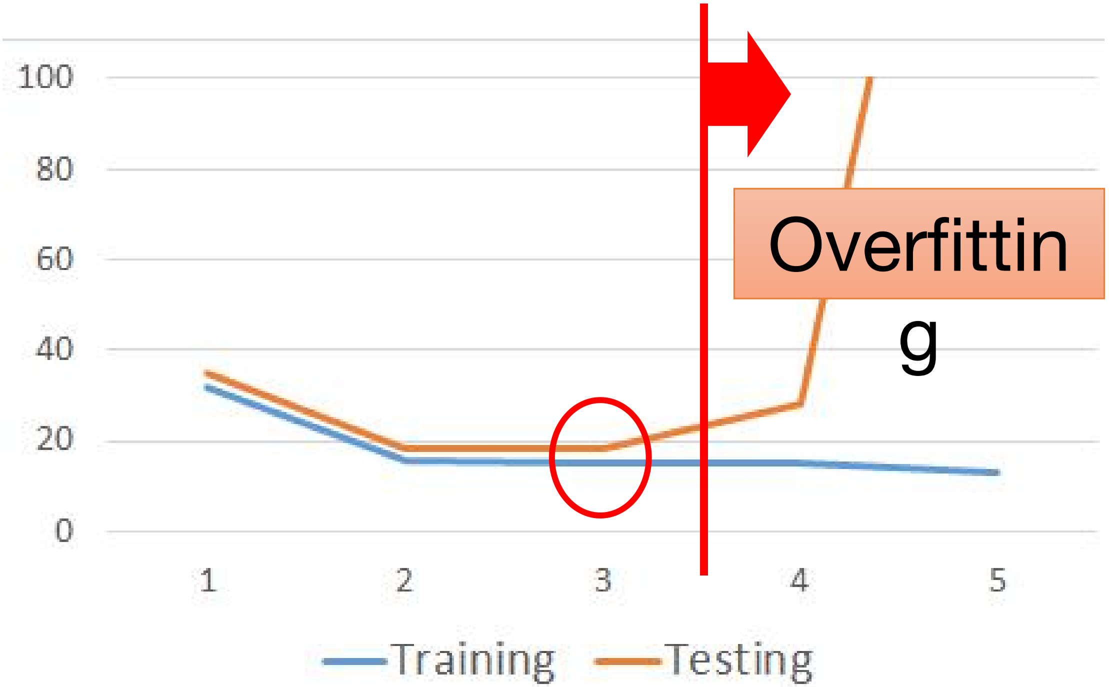
Testing:

Average Error = 232.1

The results are so bad.



Model Selection



	Trainin g	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