

# 第 6 章 线性回归

根据已知训练数据学习/训练，选择合适的模型 $h(\mathbf{X})$ ，  
代价函数越小， $h$ 函数越逼近实际目标函数。

$$\min_{\beta} J(\beta) = \frac{1}{2m} \sum_{i=1}^m (h_{\beta}(X_i) - Y_i)^2$$



找全局最优解

# 回归

- 指研究一组随机变量 ( $Y_1$  ,  $Y_2$  , ...,  $Y_i$ ) 和另一组 ( $X_1$ ,  $X_2$ , ...,  $X_k$ ) 变量之间关系的统计分析方法。回归分析是一种数学模型。
- 是监督学习的一个重要问题，用于预测输入变量和输出变量之间的关系。
- 回归模型是表示输入变量到输出变量之间映射的函数。
- 回归问题等价于函数拟合：使用一条函数曲线使其很好地拟合已知函数，且很好地预测未知数据。

# 回归应用


- 预测空气指数PM2.5

$f$  ( 过去的PM2.5信息 ) = 明天的PM2.5值

- 股票预测

$f$  (  ) = 股票走势等

- 自动驾驶

$f$  (  ) = 方向盘角度、油门刹车幅度

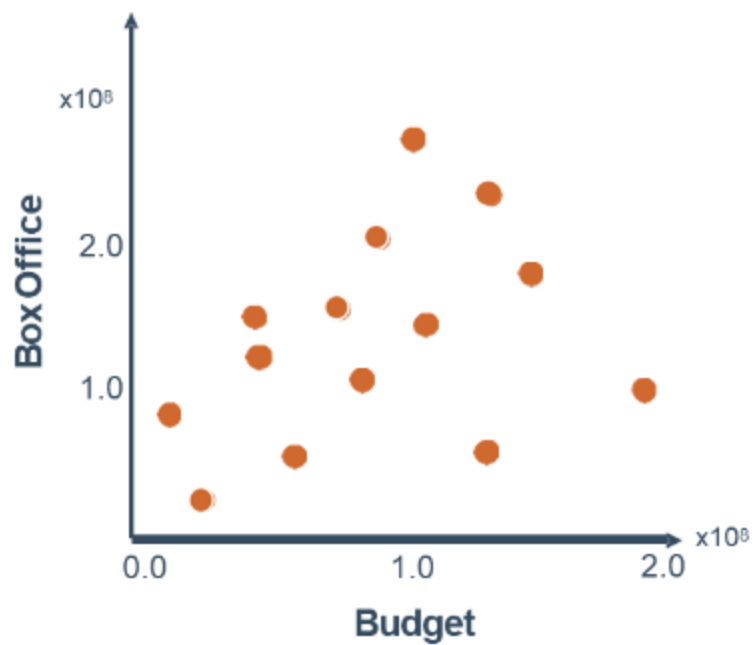
- 推荐系统

$f$  ( 用户A      商品B ) = 购买可能性值

# 线性回归

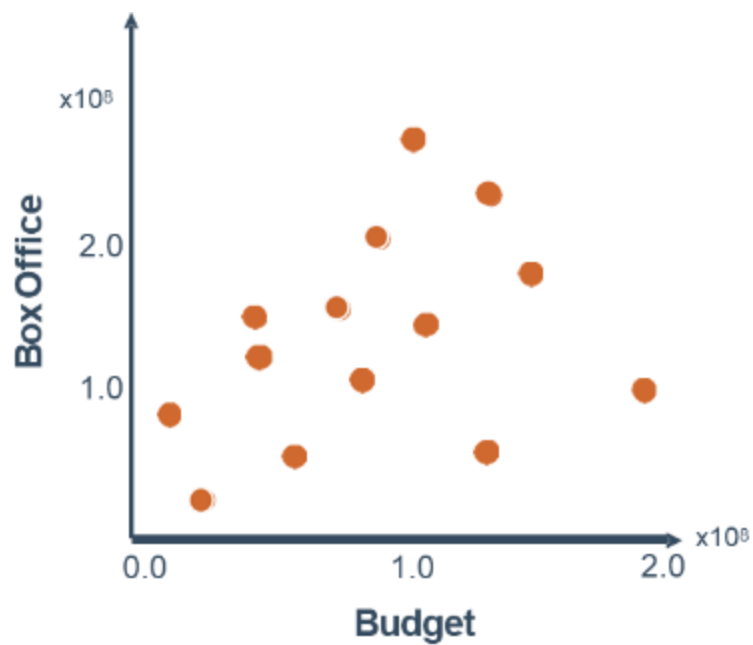
- 线性回归（linear regression）是利用数理统计中的回归分析，来确定两个或两个以上变量间相互依赖的定量关系的一种统计分析方法，运用十分广泛。
- 机器学习中，线性回归假设特征和目标变量之间满足线性关系，根据给定的训练数据建立一个线性模型，并用此模型进行预测。

# 线性回归



$$y_{\beta}(x) = \beta_0 + \beta_1 x$$

# 线性回归

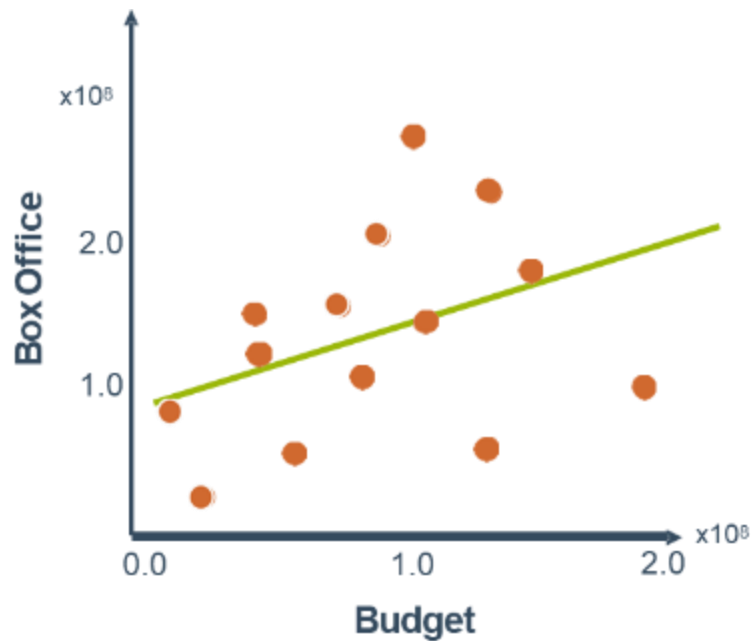


$$y_{\beta}(x) = \beta_0 + \beta_1 x$$

Annotations for the equation:

- $y_{\beta}(x)$ : box office revenue
- $\beta_0$ : coefficient 0
- $\beta_1$ : coefficient 1
- $x$ : movie budget

# 线性回归

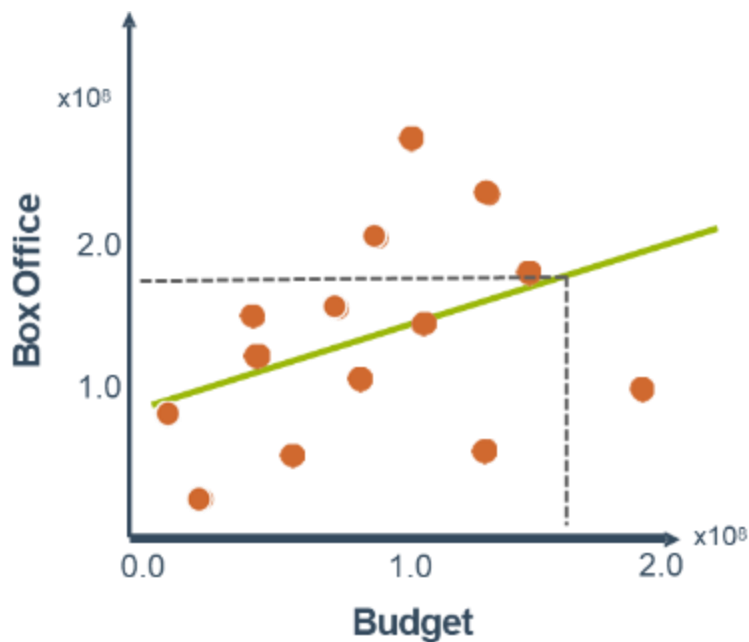


$$y_{\beta}(x) = \beta_0 + \beta_1 x$$

$$\beta_0 = 80 \text{ million}, \beta_1 = 0.6$$



# 使用线性回归预测

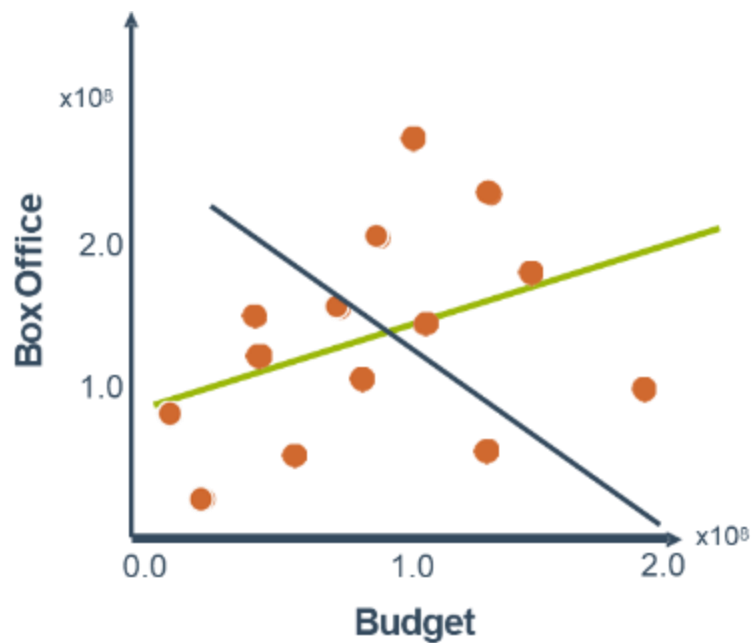


$$y_{\beta}(x) = \beta_0 + \beta_1 x$$

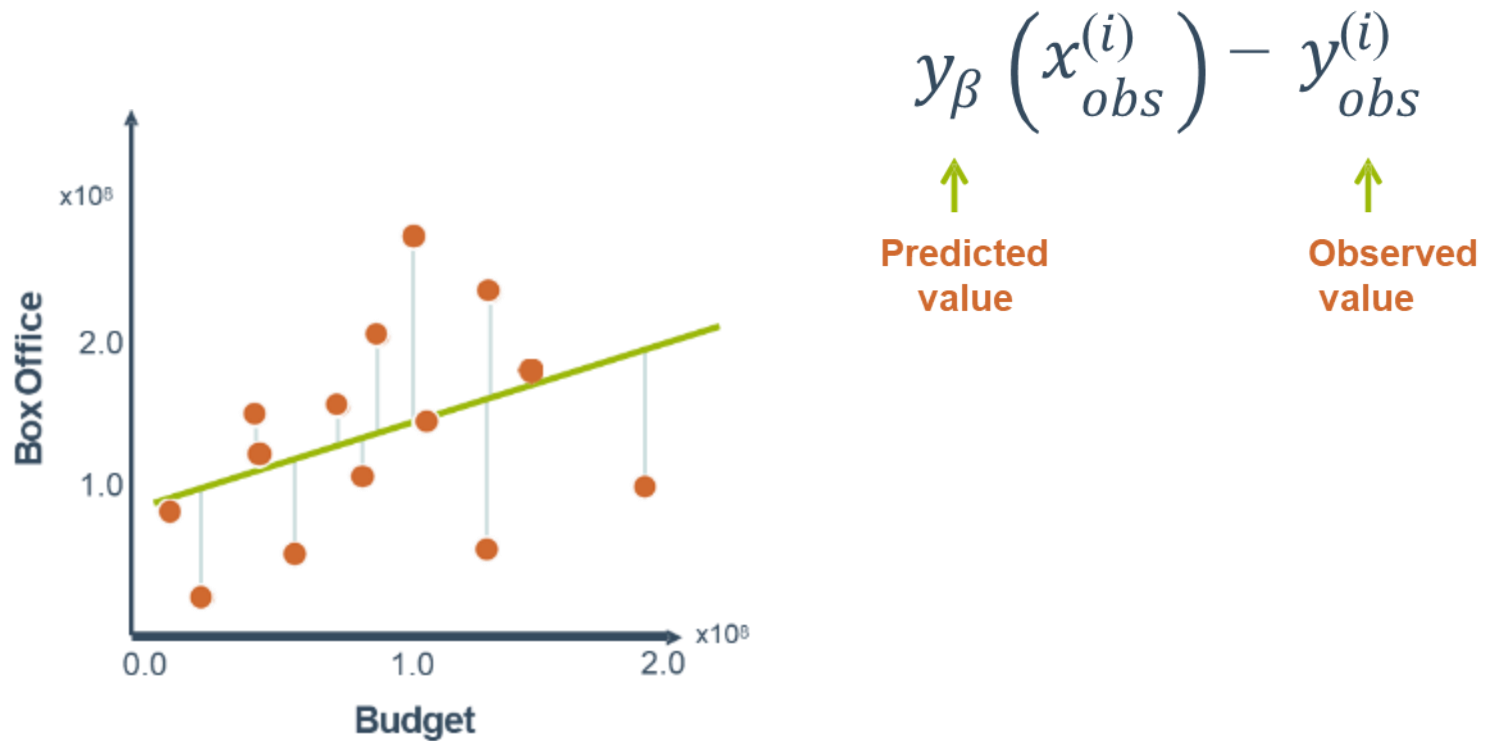
$$\beta_0 = 80 \text{ million}, \beta_1 = 0.6$$

给定1.6亿的预算，预测票房收益为1.75亿

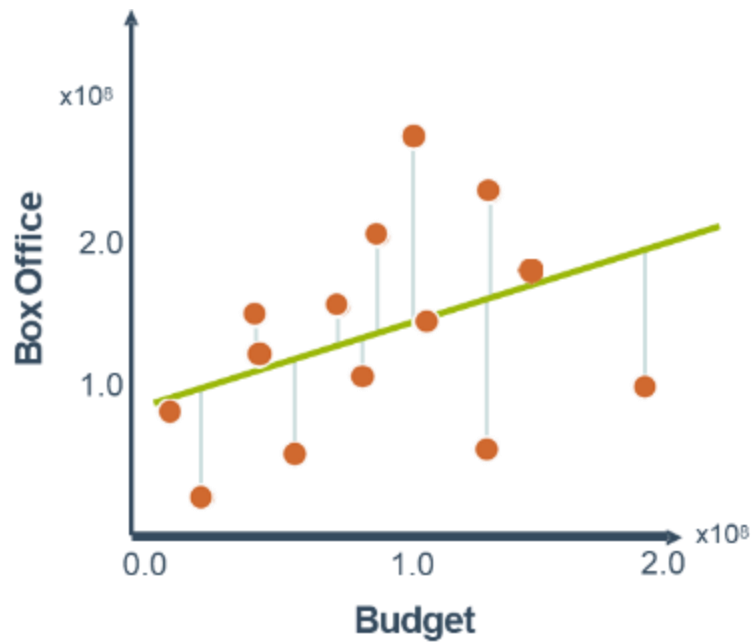
# 哪个模型拟合得更好？



# 计算残差

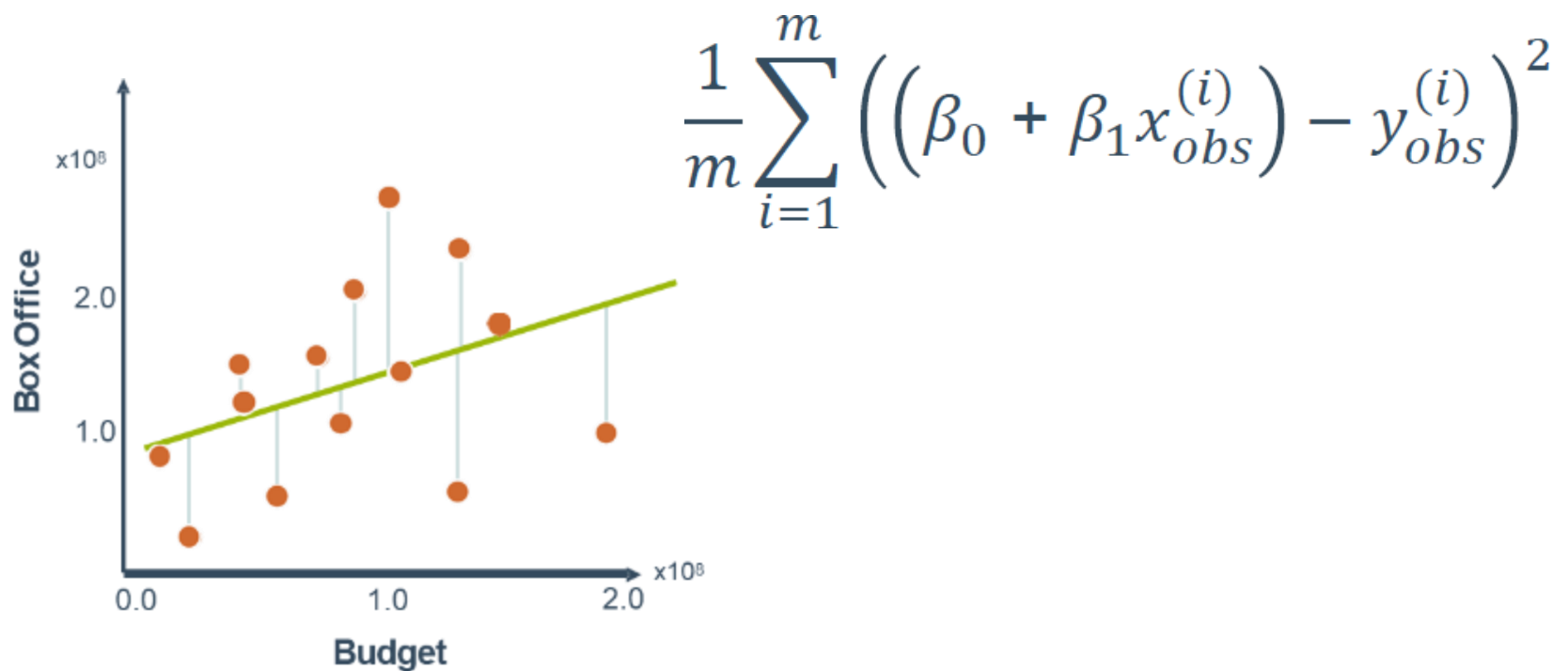


# 计算残差

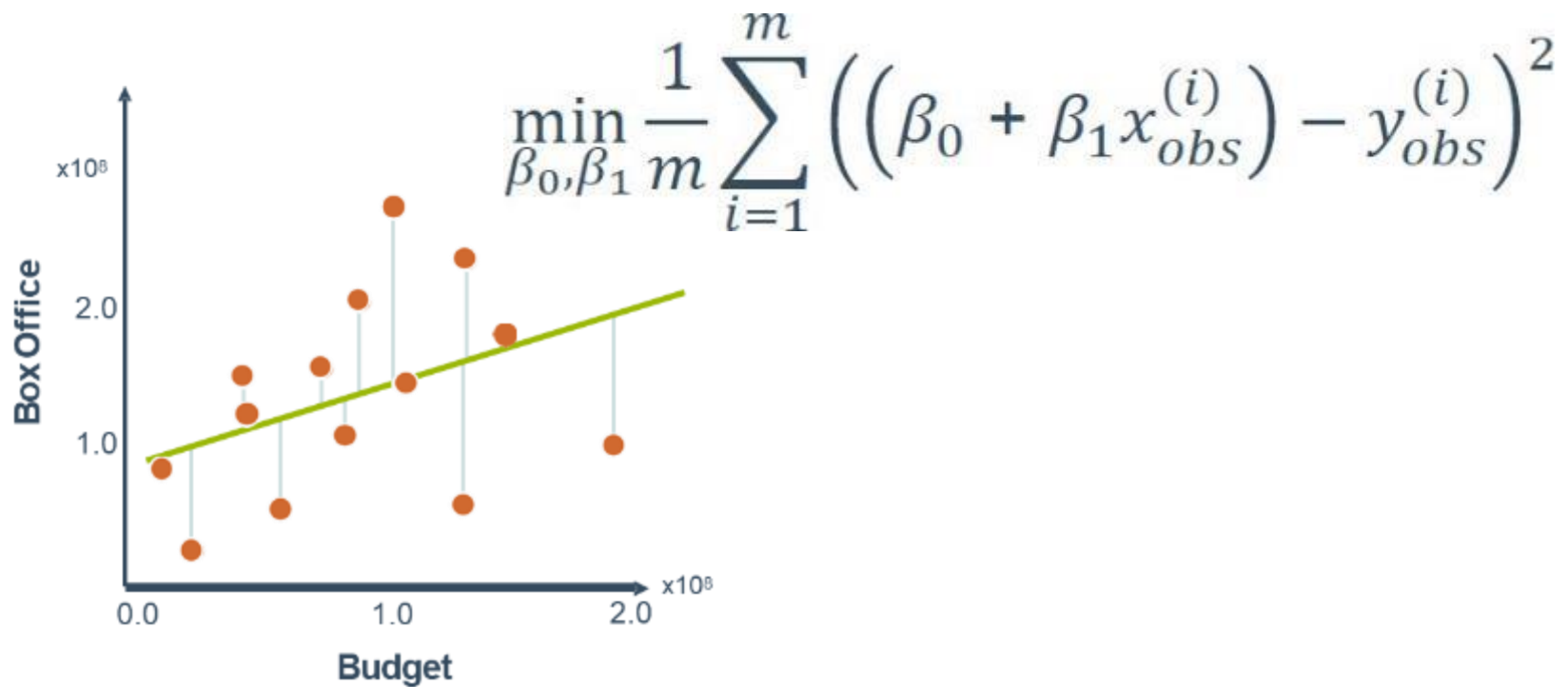


$$\left( \beta_0 + \beta_1 x_{obs}^{(i)} \right) - y_{obs}^{(i)}$$

# 均方误差 (Mean Squared Error, MSE)



# 最小均方误差



# 其他评价指标

平均绝对误差（**Mean Absolute Error, MAE**）：

$$\frac{1}{m} \sum_{i=1}^m |y_{\beta}(x^{(i)}) - y_{obs}^{(i)}|$$

均方根误差（**Root Mean Squared Error, RMSE**）：

$$\sqrt{\frac{1}{m} \sum_{i=1}^m (y_{\beta}(x^{(i)}) - y_{obs}^{(i)})^2}$$

**MAE** is the easiest to understand, because it's the average error.

**MSE** is more popular than MAE, because MSE "punishes" larger errors.

**RMSE** is even more popular than MSE, because RMSE is interpretable in the "y" units.

# 其他评价指标

残差平方和（SSE）：

$$\sum_{i=1}^m \left( y_{\beta}(x^{(i)}) - y_{obs}^{(i)} \right)^2$$

总离差平方和（TSS）：

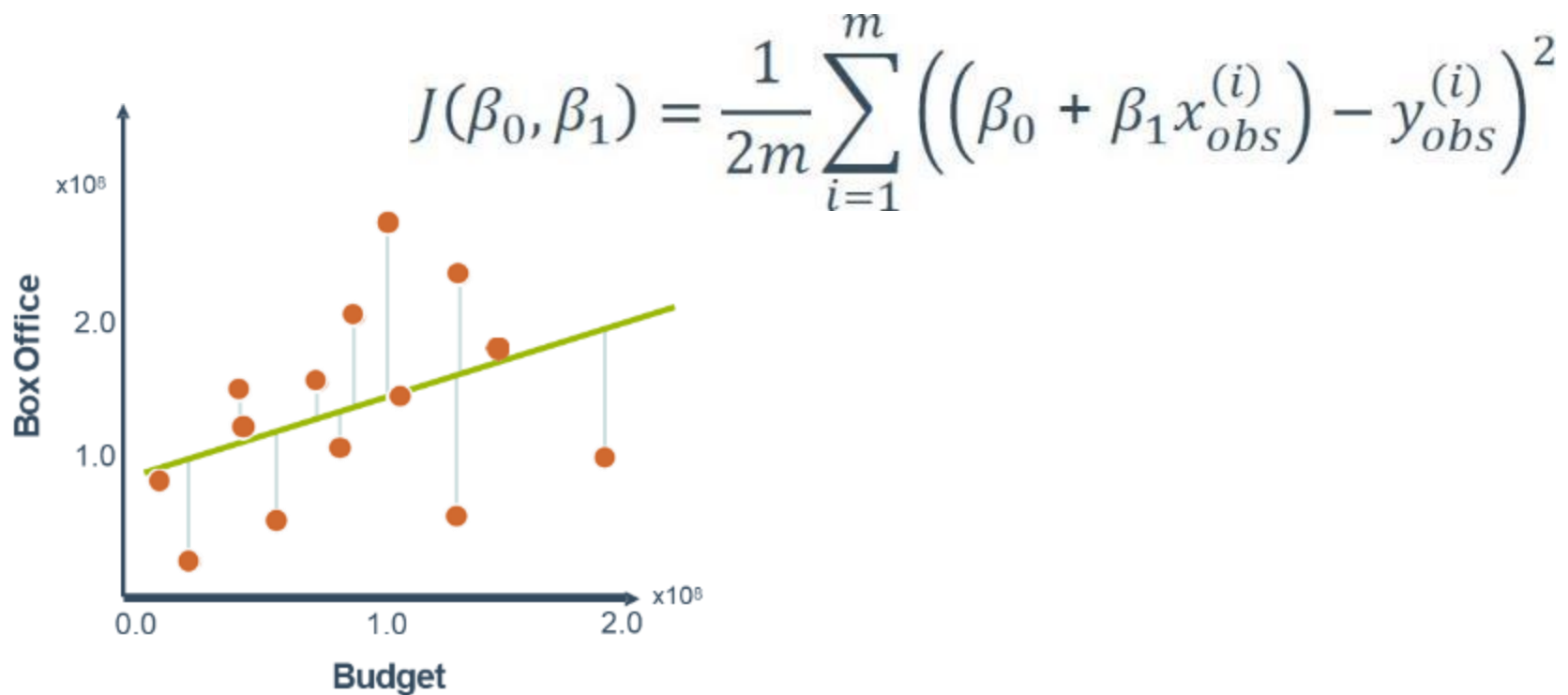
$$\sum_{i=1}^m \left( \overline{y_{obs}} - y_{obs}^{(i)} \right)^2$$

决定系数（R<sup>2</sup>）：

$$1 - \frac{SSE}{TSS}$$



# 代价函数



# 最小二乘法

$$y_{\beta}(x) = \beta_0 + \beta_1 x$$

$$\widehat{\beta}_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

$$\widehat{\beta}_0 = \bar{y} - \widehat{\beta}_1 \bar{x}$$

# 最小二乘法

	工资	额度
1	4000	20000
2	5000	30000
3	8000	50000
4	10000	70000
5	12000	60000
6	15000	?

- (1) 求贷款额度  $y$  关于月收入  $x$  的线性回归方程；
- (2) 利用 (1) 中的回归方程，预测张三（月工资 15000）的贷款额度。

# 最小二乘法

某地区2007年至2013年农村居民家庭人均纯收入  $y$ （单位：千元）的数据如下表：

年份	2007	2008	2009	2010	2011	2012	2013
年份代号 $x$	1	2	3	4	5	6	7
人均收入 $y$	2.9	3.3	3.6	4.4	4.8	5.2	5.9

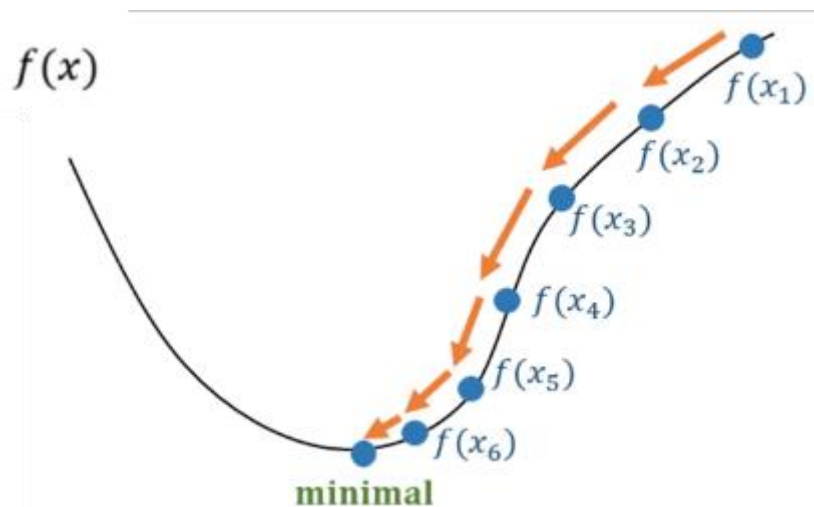
- （1）求  $y$  关于  $x$  的线性回归方程；
- （2）利用（1）中的回归方程，预测该地区2015年农村居民家庭人均纯收入。

# 梯度下降法

$$y_{\beta}(x) = \beta_0 + \beta_1 x$$

$$J(\beta_0, \beta_1) = \frac{1}{2m} \sum_{i=1}^m \left( (\beta_0 + \beta_1 x_{obs}^{(i)}) - y_{obs}^{(i)} \right)^2$$

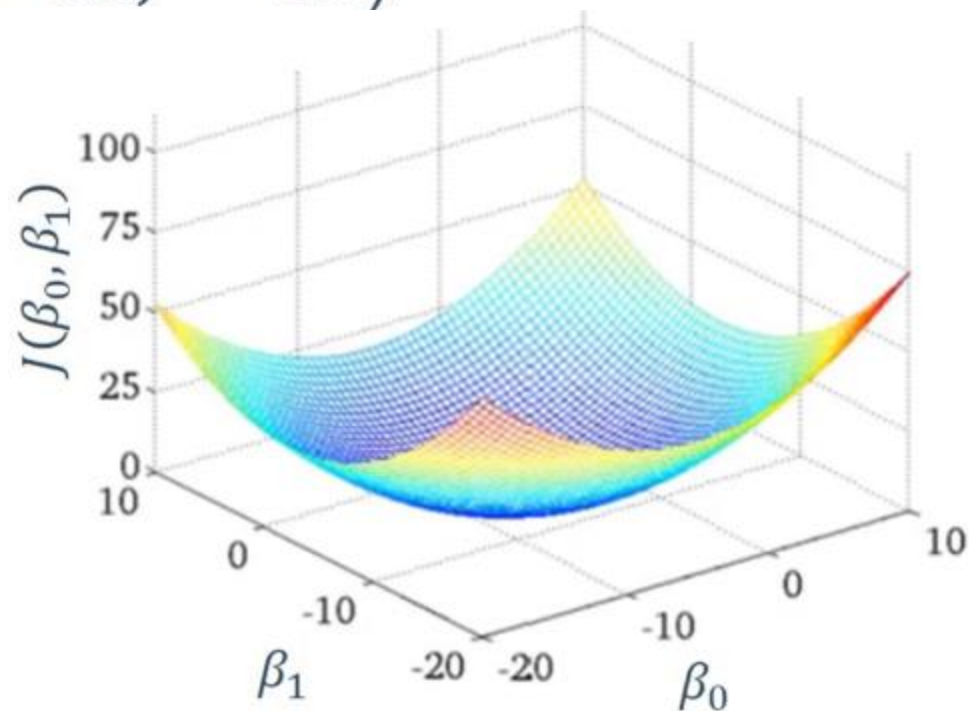
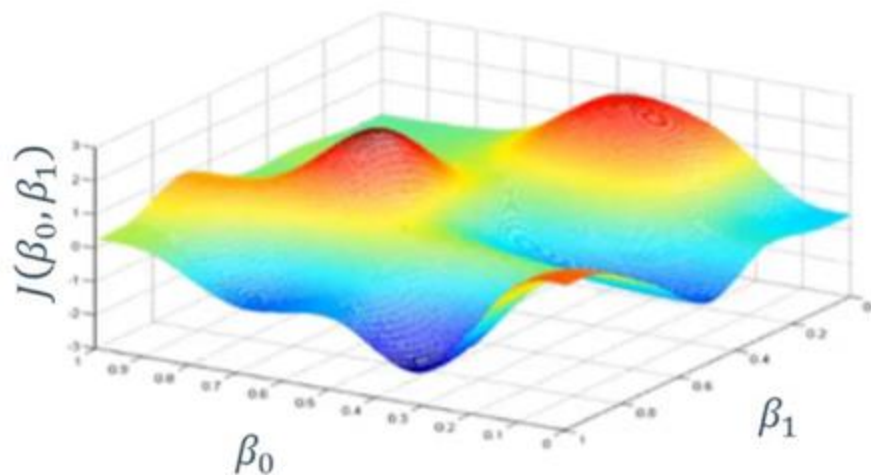
一种常用的一阶优化方法。该算法从任一点开始，沿该点**梯度**的反方向运动一段距离，再沿新位置的梯度反方向运行一段距离，如此迭代，解一直朝下坡最陡的方向运动，希望能运动到函数的全局最小点。



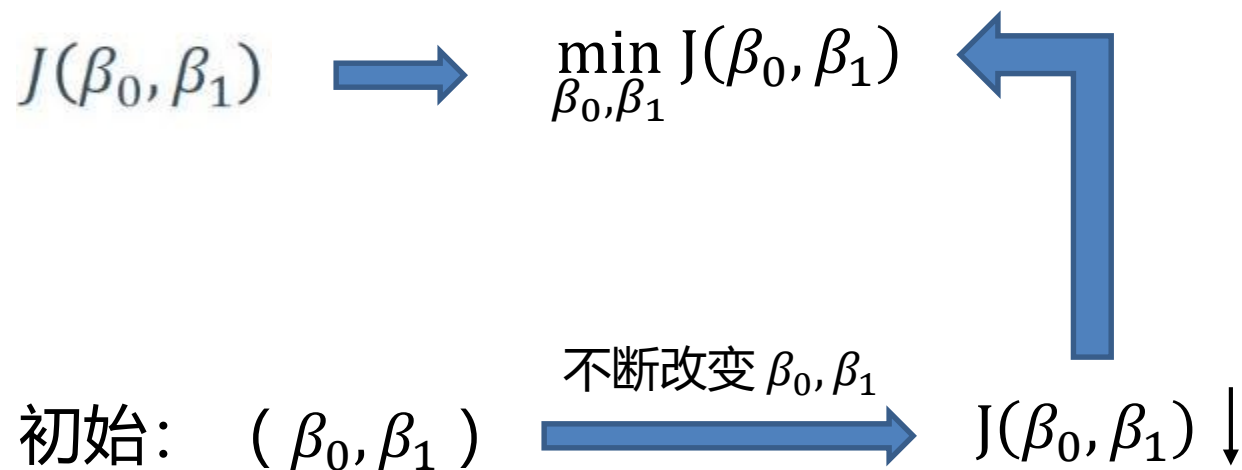
# 梯度下降法

$$y_{\beta}(x) = \beta_0 + \beta_1 x$$

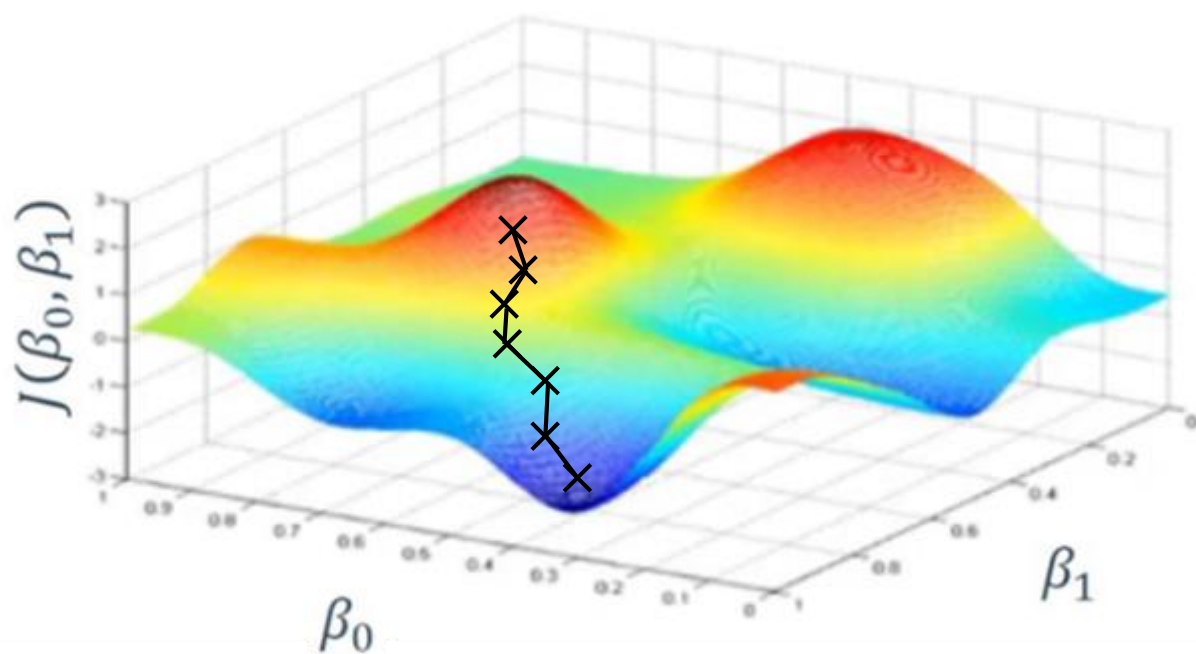
$$J(\beta_0, \beta_1) = \frac{1}{2m} \sum_{i=1}^m \left( (\beta_0 + \beta_1 x_{obs}^{(i)}) - y_{obs}^{(i)} \right)^2$$



# 梯度下降法

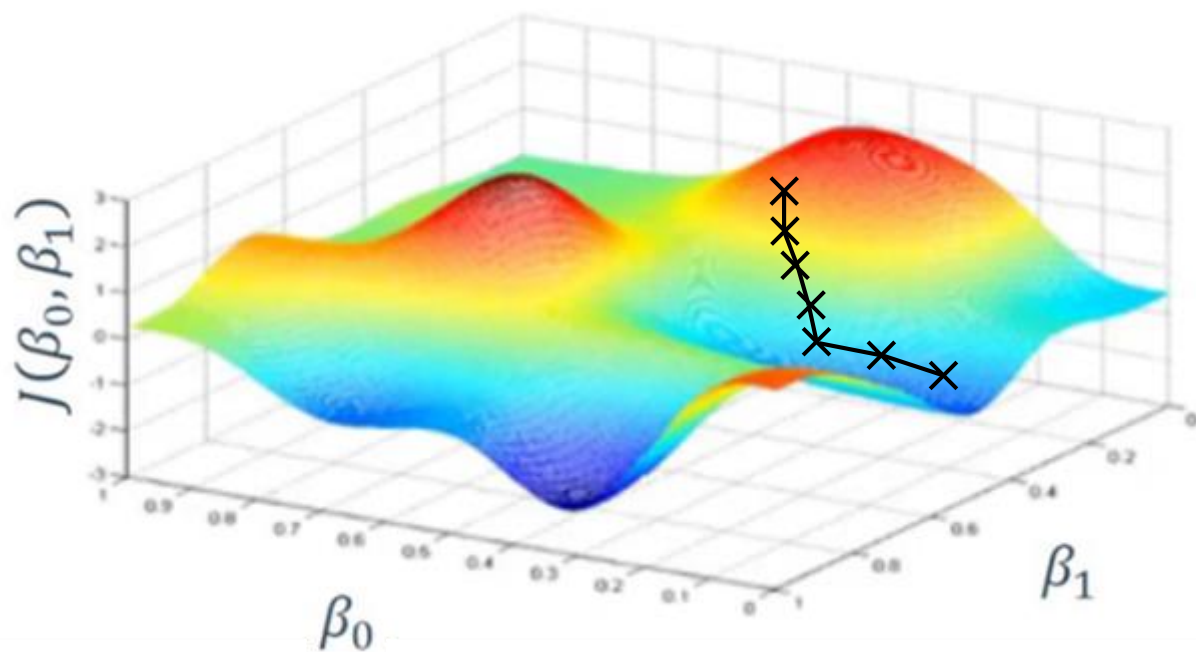


# 梯度下降法





# 梯度下降法



# 梯度下降法

数学定义：

$$\beta_i := \beta_i - \alpha \frac{\partial}{\partial \beta_i} J(\beta_0, \beta_1), \quad i=0,1 \Rightarrow \text{不断迭代直到收敛}$$

正确过程：同时更新

$$\text{temp0} := \beta_0 - \alpha \frac{\partial}{\partial \beta_0} J(\beta_0, \beta_1)$$

$$\text{temp1} := \beta_1 - \alpha \frac{\partial}{\partial \beta_1} J(\beta_0, \beta_1)$$

$$\beta_0 := \text{temp0}$$

$$\beta_1 := \text{temp1}$$

# 梯度下降法

偏导数项意义：

$$\beta_i := \beta_i - \alpha \frac{\partial}{\partial \beta_i} J(\beta_0, \beta_1), \quad i=0,1$$



$\min_{\beta_1} J(\beta_1)$  为例

# 梯度下降法

学习速率 $\alpha$ :

$$\beta_i := \beta_i - \alpha \frac{\partial}{\partial \beta_i} J(\beta_0, \beta_1), \quad i=0,1$$

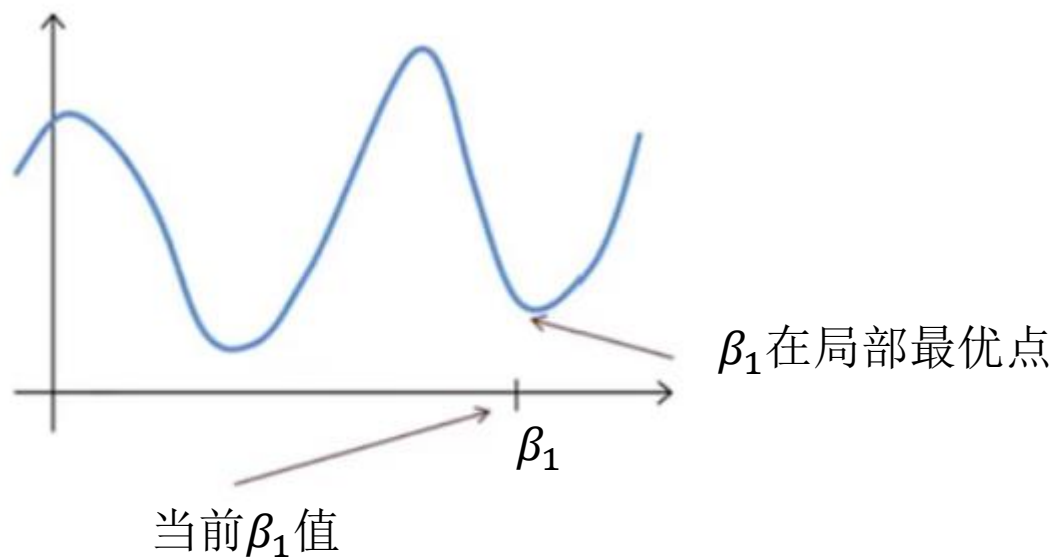


取值大小影响收敛速度

# 梯度下降法

局部最低点:

$$\beta_1 := \beta_1 - \alpha \frac{\partial}{\partial \beta_1} J(\beta_1)$$



接近局部最优点时，梯度下降法会自动减小步长，所以没必要另外减小 $\alpha$ 。

# 梯度下降法

线性回归模型：

$$h_{\beta}(x) = \beta_0 + \beta_1 x$$

$$J(\beta_0, \beta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\beta}(x^{(i)}) - y^{(i)})^2$$



$$\beta_i := \beta_i - \alpha \frac{\partial}{\partial \beta_i} J(\beta_0, \beta_1), \quad i=0,1$$

$$\min_{\beta_0, \beta_1} J(\beta_0, \beta_1)$$

# 梯度下降法

线性回归模型：

$$\frac{\partial}{\partial \beta_i} J(\beta_0, \beta_1) = ?$$

# 梯度下降法

梯度下降算法：

Repeat until convergence {

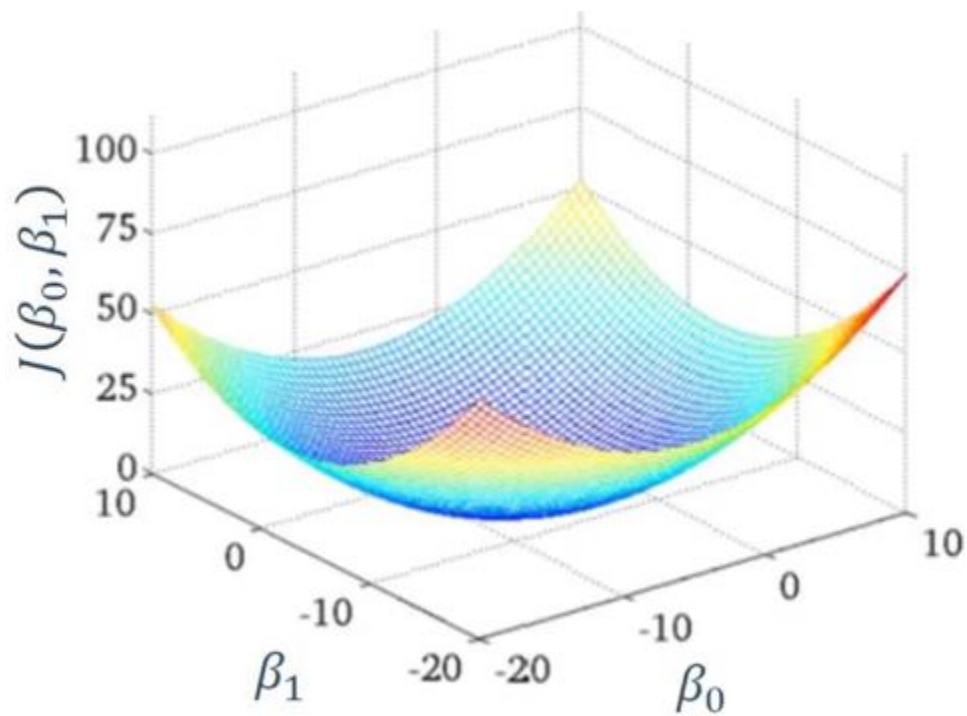
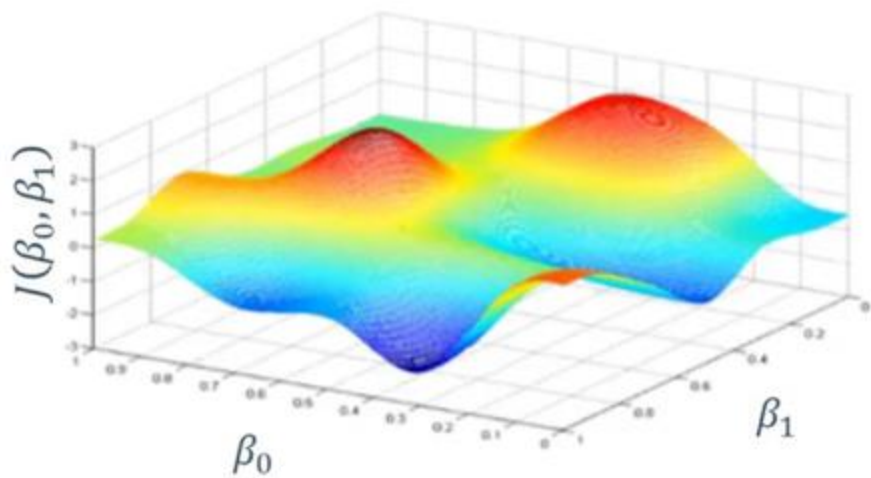
$$\beta_0 := \beta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\beta}(x^{(i)}) - y^{(i)})$$

$$\beta_1 := \beta_1 - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\beta}(x^{(i)}) - y^{(i)}) \cdot x^{(i)}$$

}



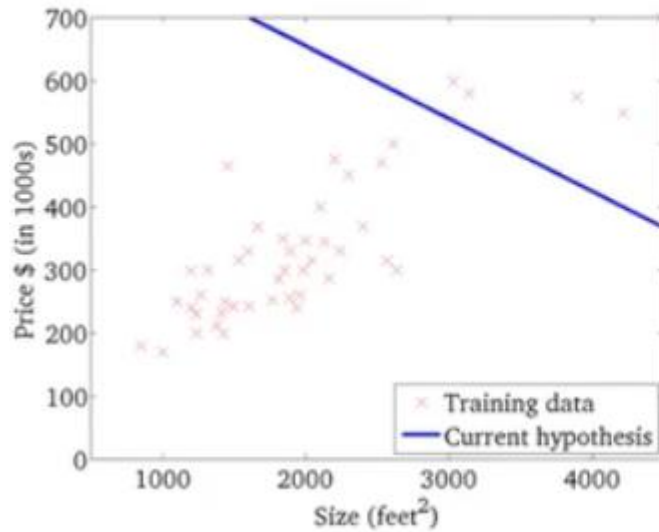
# 梯度下降法



# 梯度下降法

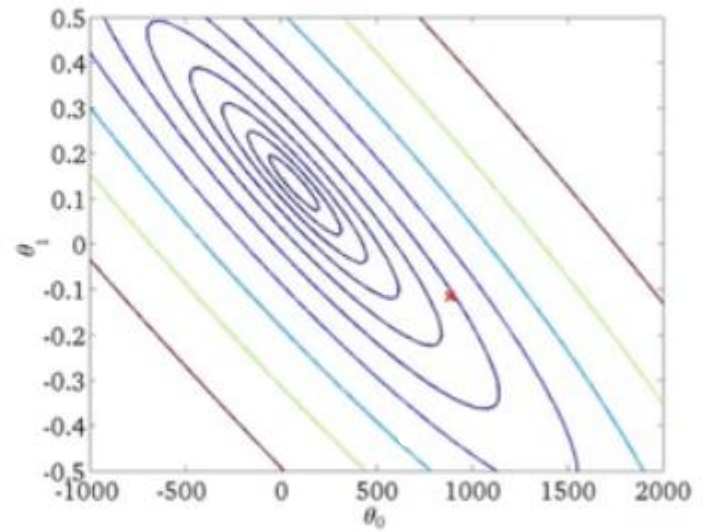
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



$$J(\theta_0, \theta_1)$$

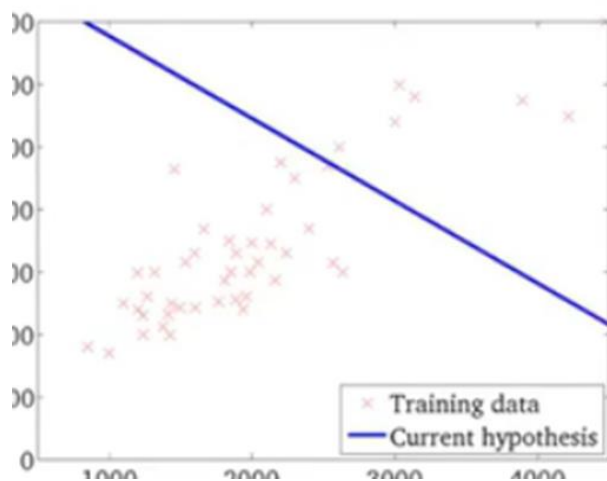
(function of the parameters  $\theta_0, \theta_1$ )



# 梯度下降法

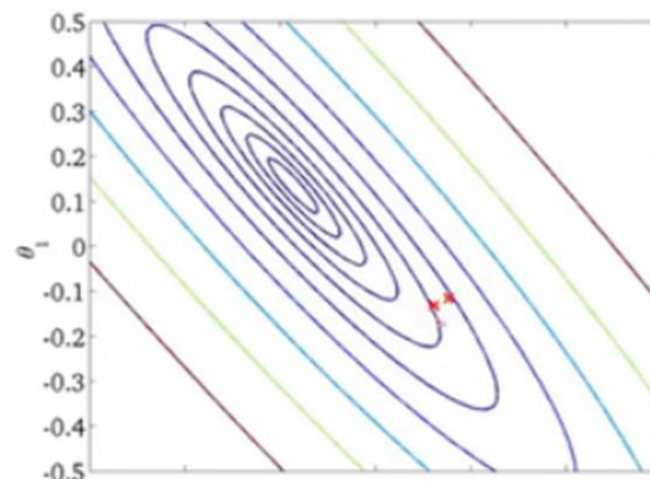
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



$$J(\theta_0, \theta_1)$$

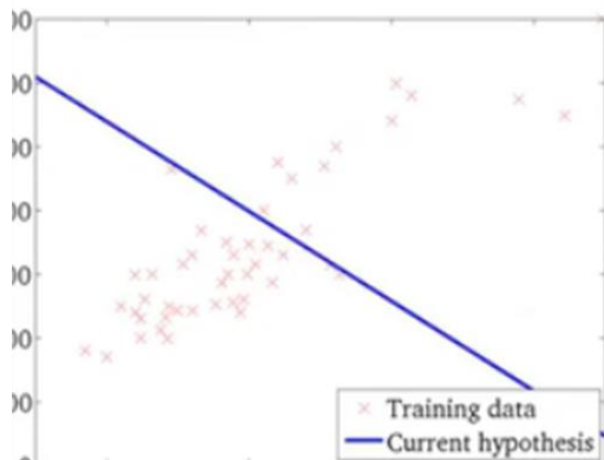
(function of the parameters  $\theta_0, \theta_1$ )



# 梯度下降法

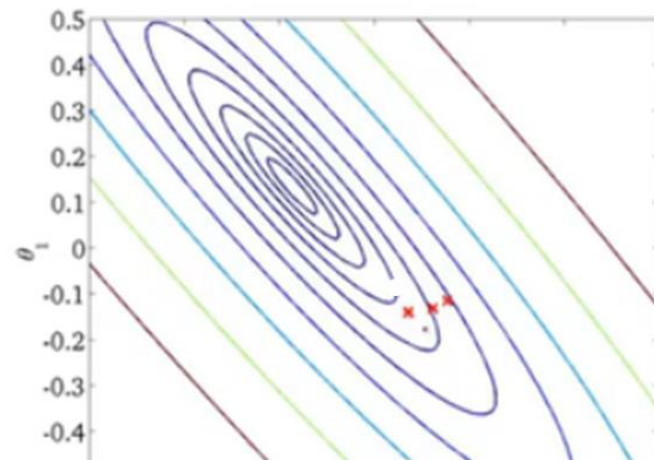
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



$$J(\theta_0, \theta_1)$$

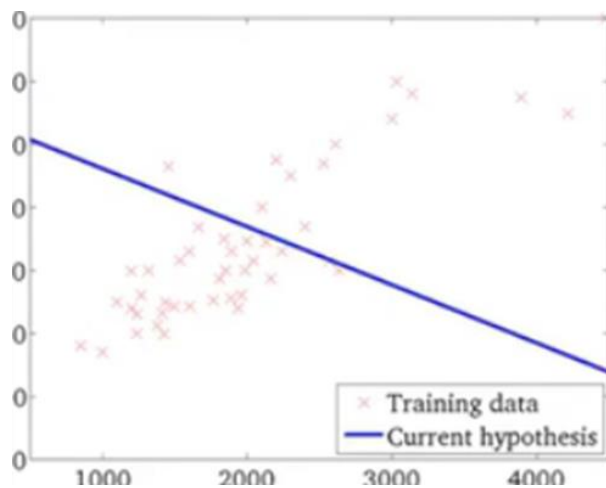
(function of the parameters  $\theta_0, \theta_1$ )



# 梯度下降法

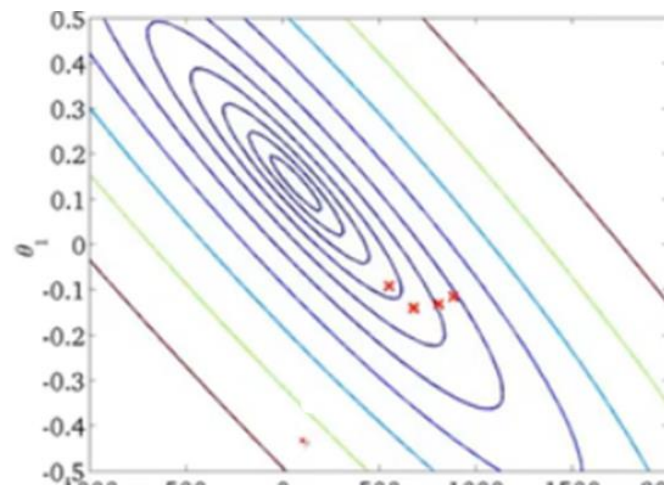
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



$$J(\theta_0, \theta_1)$$

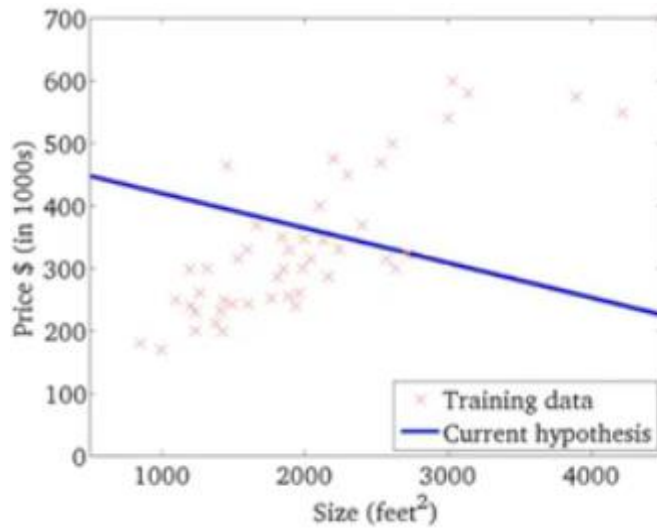
(function of the parameters  $\theta_0, \theta_1$ )



# 梯度下降法

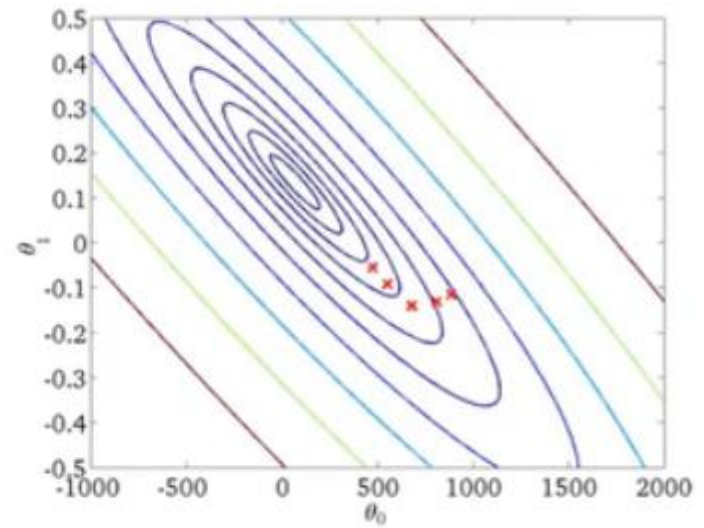
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



$$J(\theta_0, \theta_1)$$

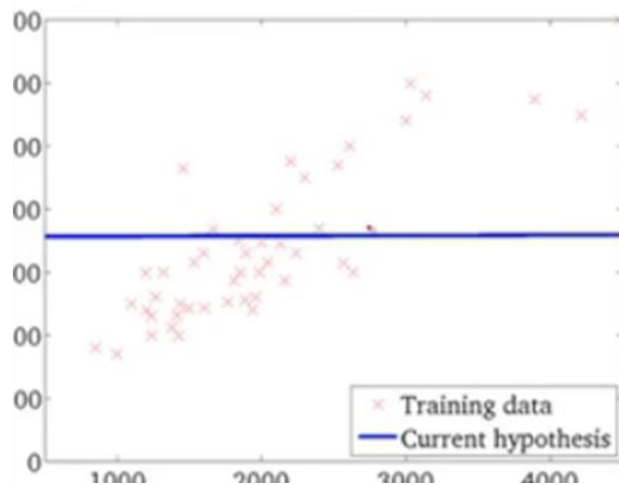
(function of the parameters  $\theta_0, \theta_1$ )



# 梯度下降法

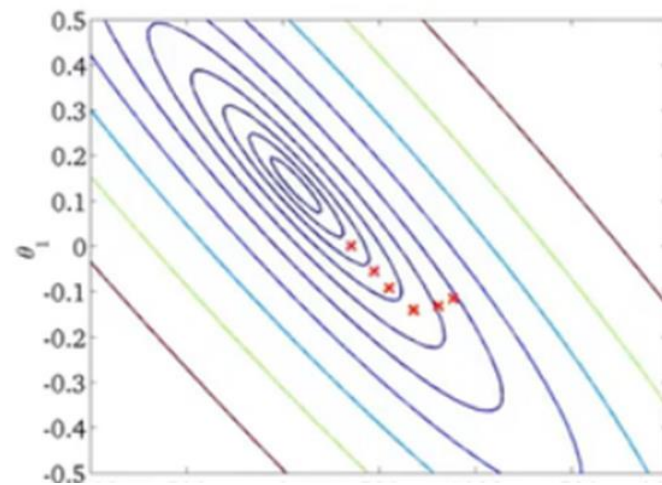
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



$$J(\theta_0, \theta_1)$$

(function of the parameters  $\theta_0, \theta_1$ )

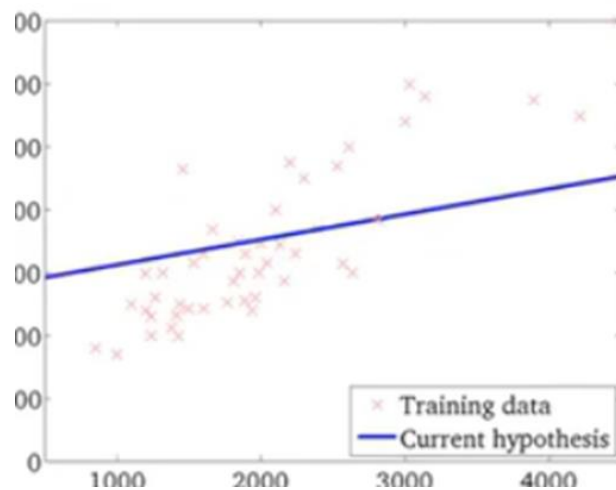




# 梯度下降法

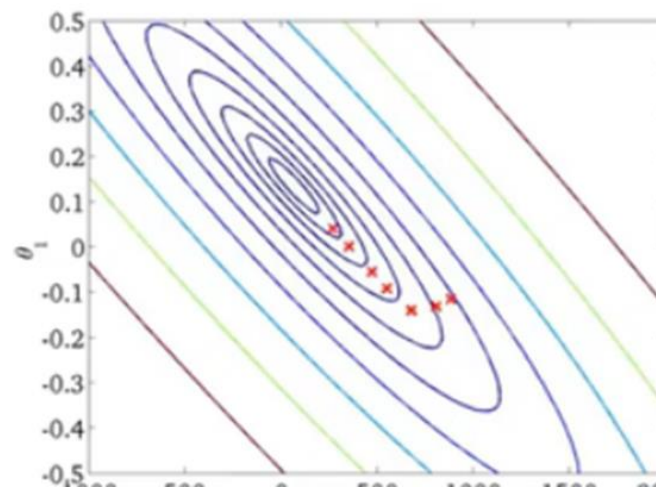
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



$$J(\theta_0, \theta_1)$$

(function of the parameters  $\theta_0, \theta_1$ )

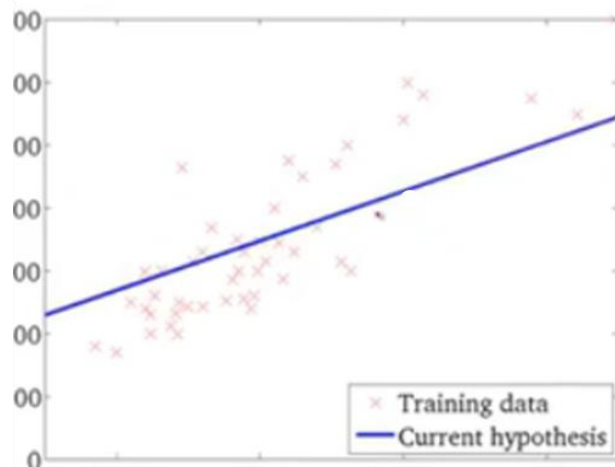




# 梯度下降法

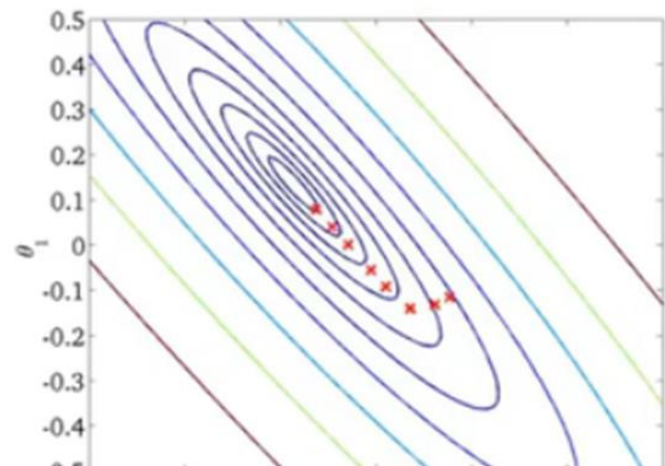
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



$$J(\theta_0, \theta_1)$$

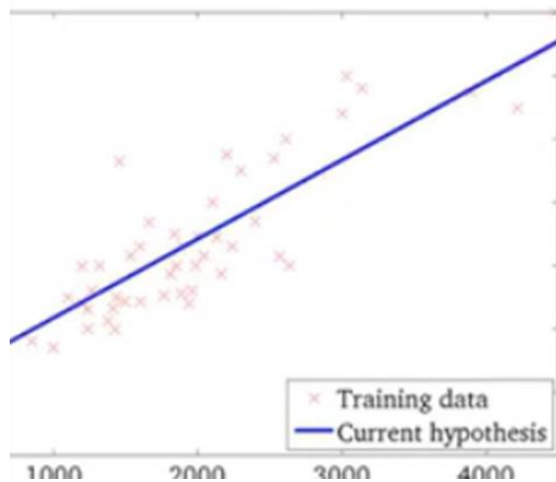
(function of the parameters  $\theta_0, \theta_1$ )



# 梯度下降法

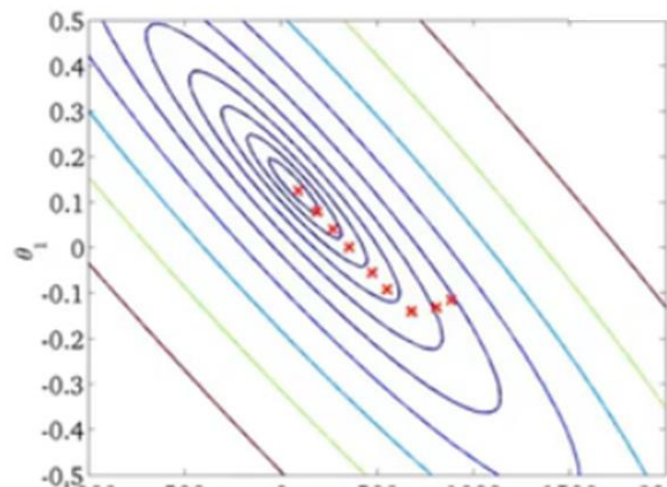
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



$$J(\theta_0, \theta_1)$$

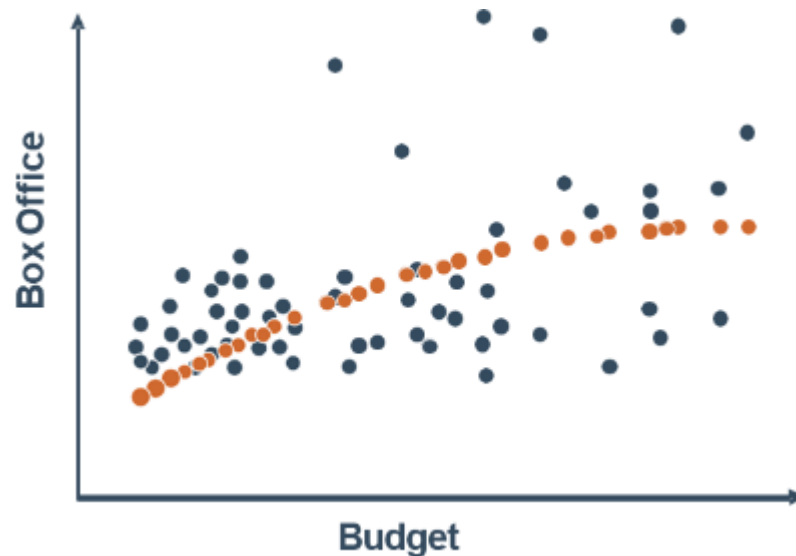
(function of the parameters  $\theta_0, \theta_1$ )



# 增加多项式特征

- 通过增加多项式特征来捕捉更高阶的数据特征

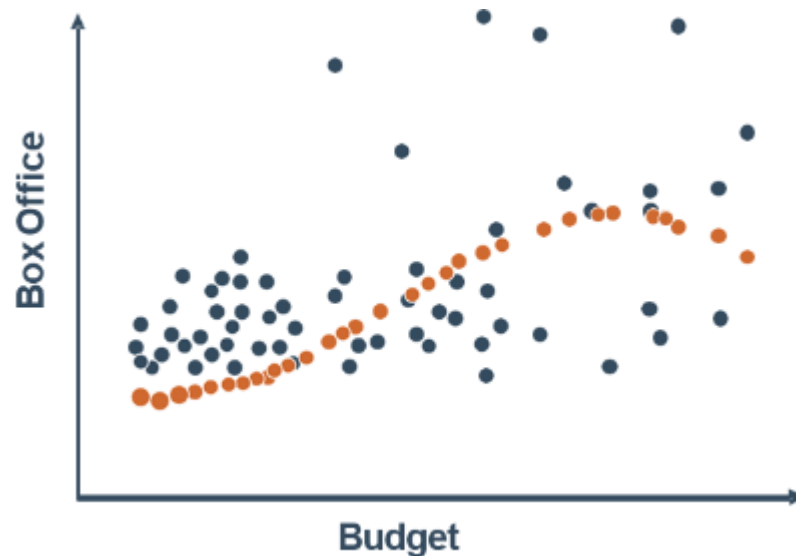
$$y_{\beta}(x) = \beta_0 + \beta_1 x + \beta_2 x^2$$



# 增加多项式特征

- 通过增加多项式特征来捕捉更高阶的数据特征
- “线性回归”意味着特征间的线性组合

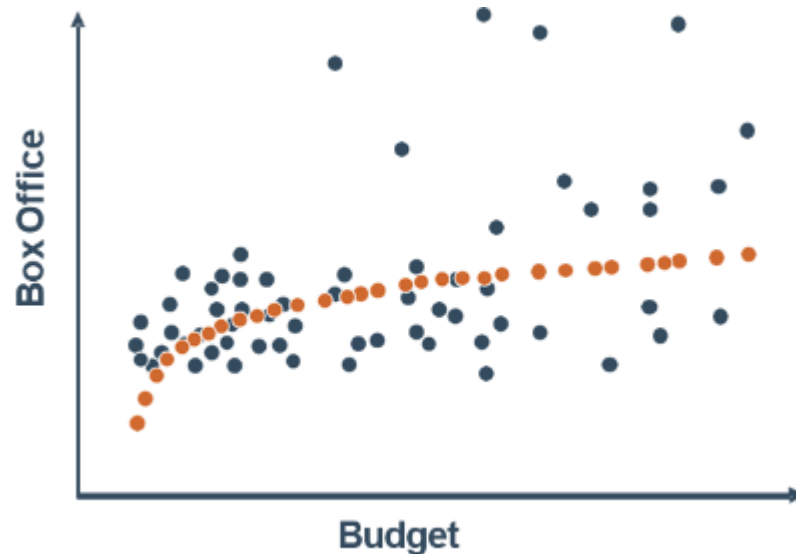
$$y_{\beta}(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$$



# 增加多项式特征

- 通过增加多项式特征来捕捉更高阶的数据特征
- “线性回归”意味着特征间的线性组合

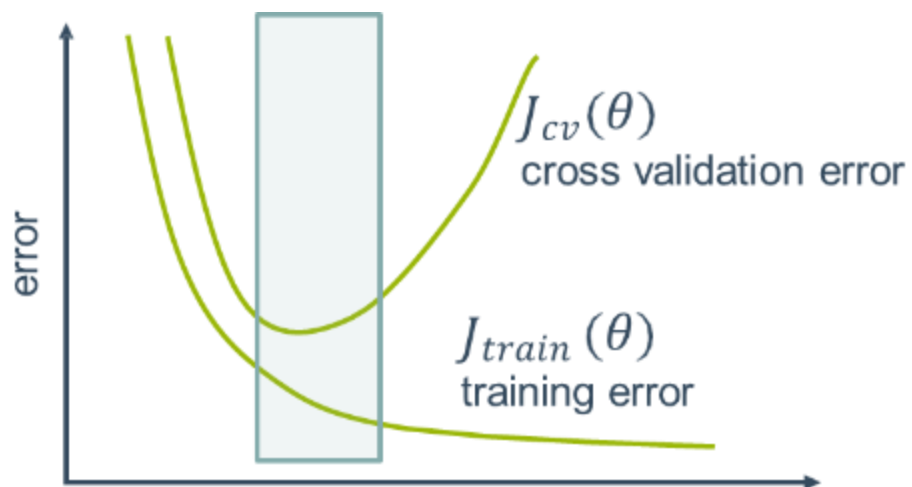
$$y_{\beta}(x) = \beta_0 + \beta_1 \log(x)$$



# 增加多项式特征

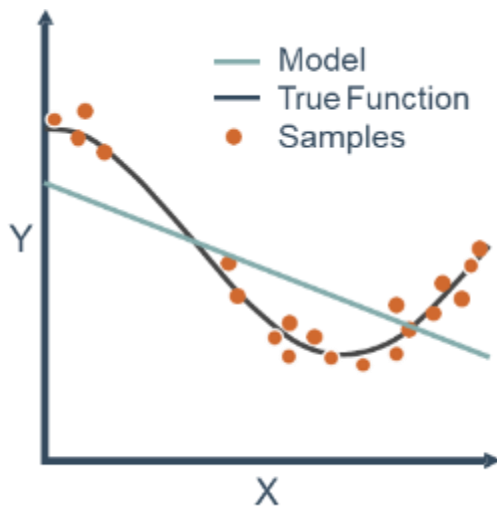
- 可以选择变量间的交互项:  $y_{\beta}(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2$
- 如何选择正确的函数形式: ➡ 检查每个变量与结果之间的关系

# 模型复杂度与误差

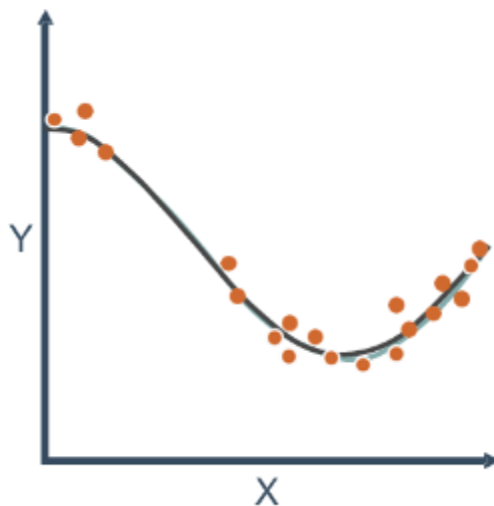


# 防止欠拟合与过拟合

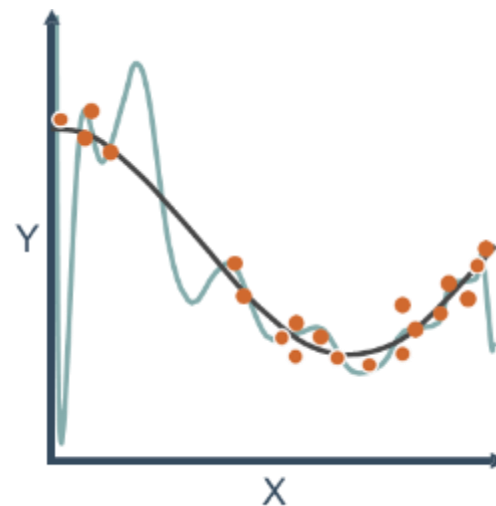
Polynomial Degree = 1



Polynomial Degree = 3



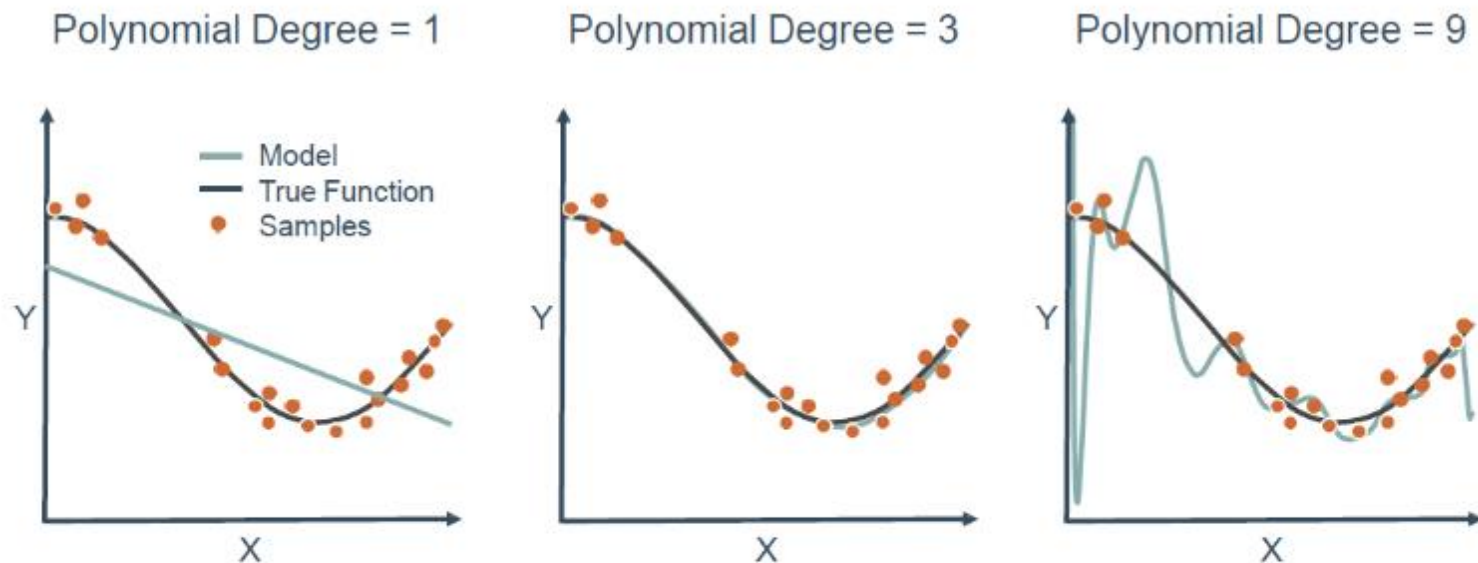
Polynomial Degree = 9



如何用一个**9**次多项式拟合数据，并防止过拟合？

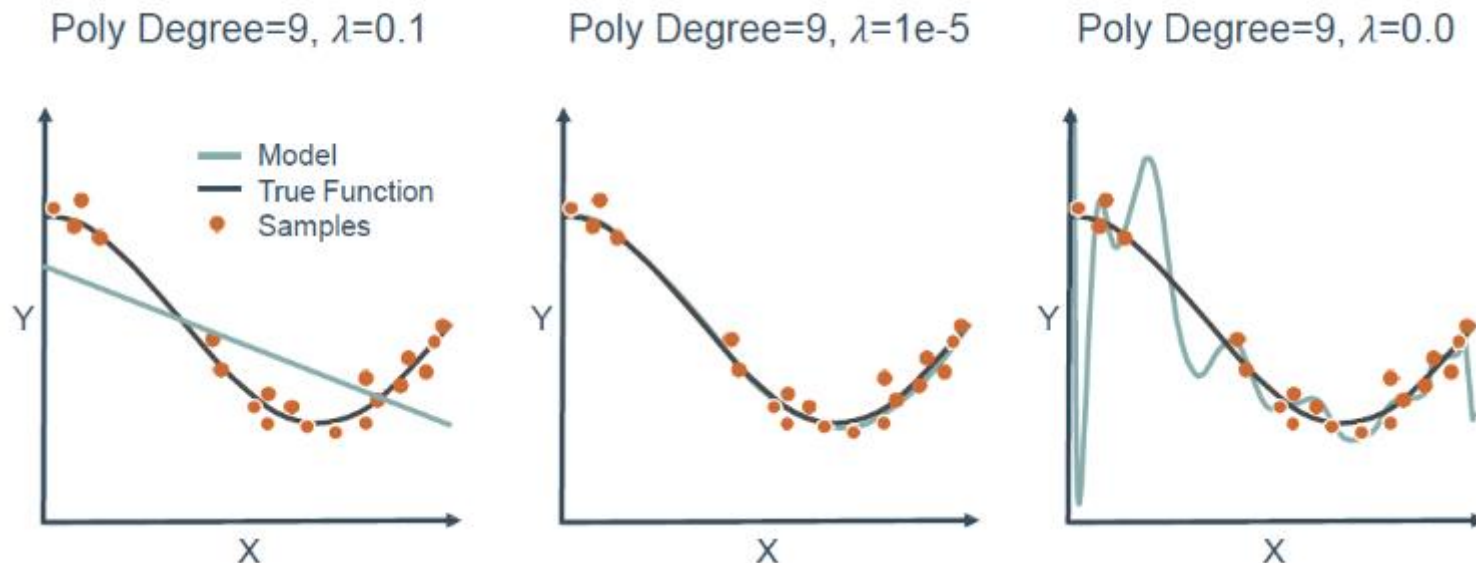


# 防止欠拟合与过拟合



$$J(\beta) = \frac{1}{2m} \sum_{i=1}^m (y_{\beta}(x^{(i)}) - y^{(i)})^2$$

# 正则化 (regularization)



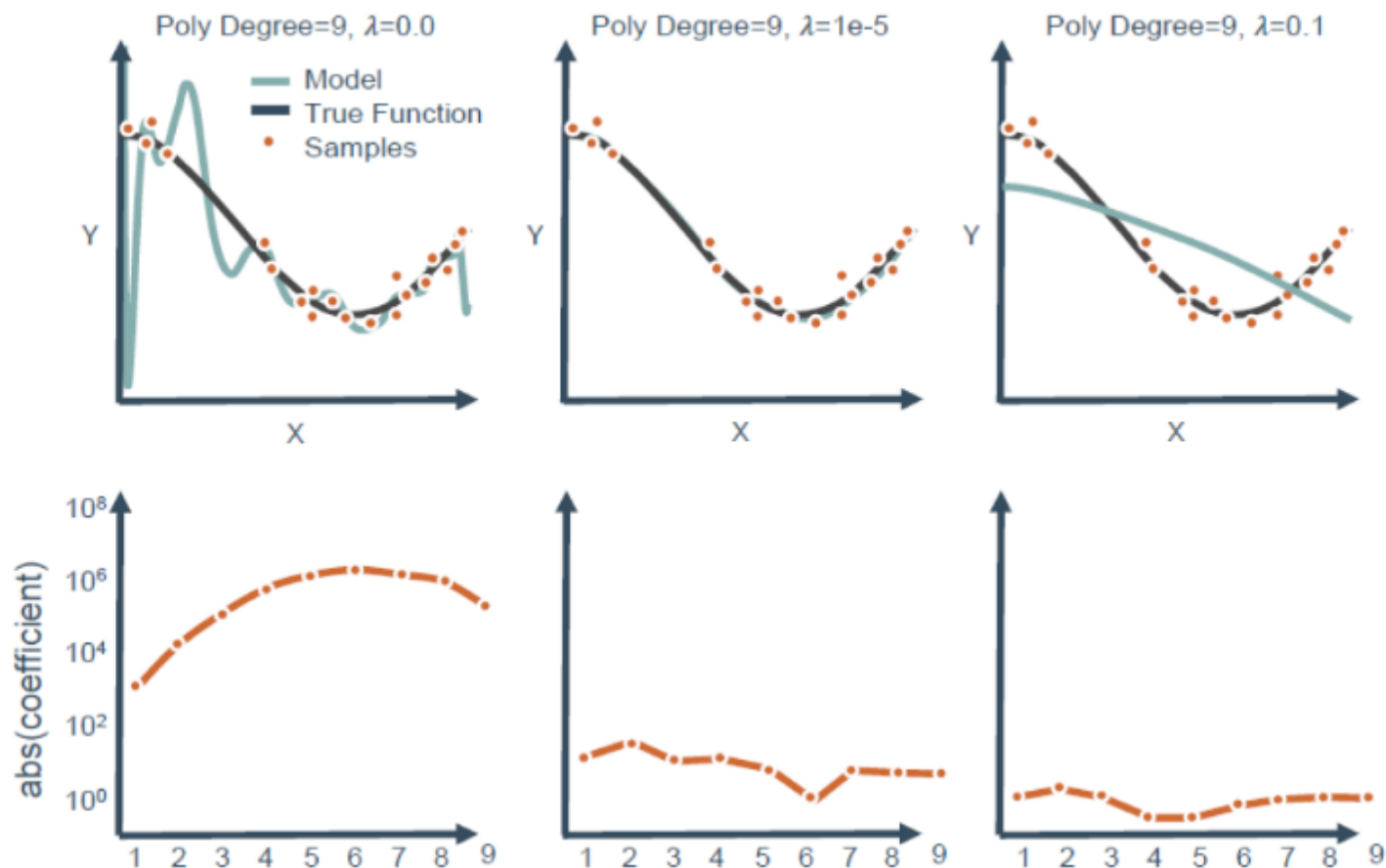
$$J(\beta) = \frac{1}{2m} \sum_{i=1}^m (y_{\beta}(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n \beta_j^2$$

# 岭回归（Ridge Regression）(L2)

$$J(\beta) = \frac{1}{2m} \sum_{i=1}^m (y_{\beta}(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n \beta_j^2$$

- 惩罚项收缩了所有系数的大小
- 越大的系数被惩罚得越多，因为惩罚的是平方

# 岭回归对模型参数的效果



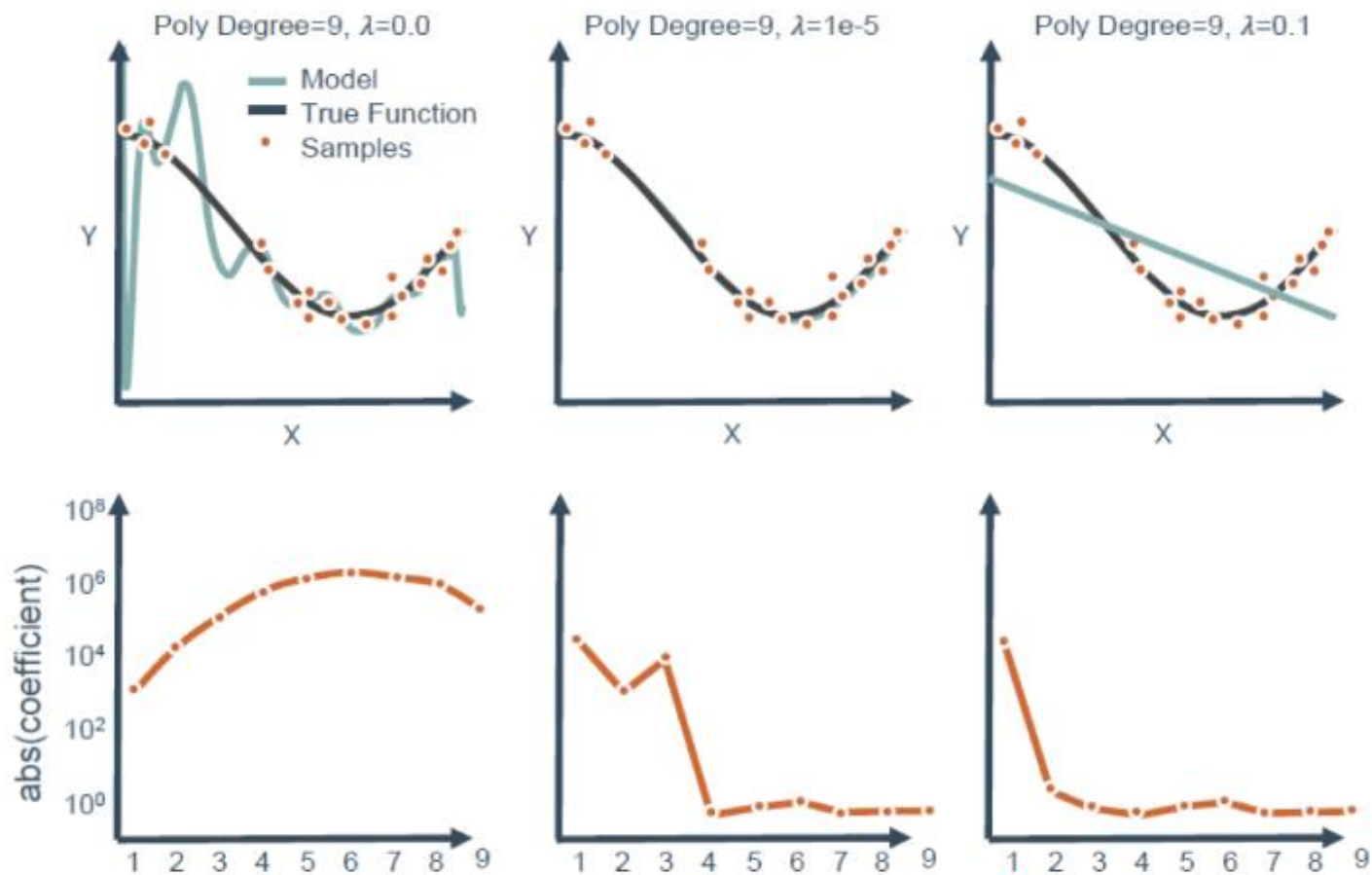
$$J(\beta) = \frac{1}{2m} \sum_{i=1}^m (y_{\beta}(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n \beta_j^2$$

# 套索回归（Lasso Regression）(L1)

$$J(\beta) = \frac{1}{2m} \sum_{i=1}^m (y_{\beta}(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n |\beta_j|$$

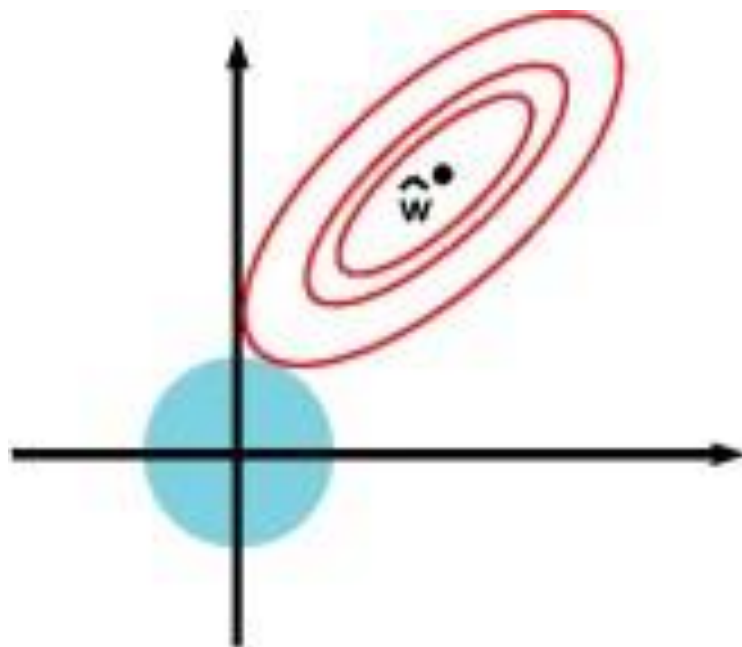
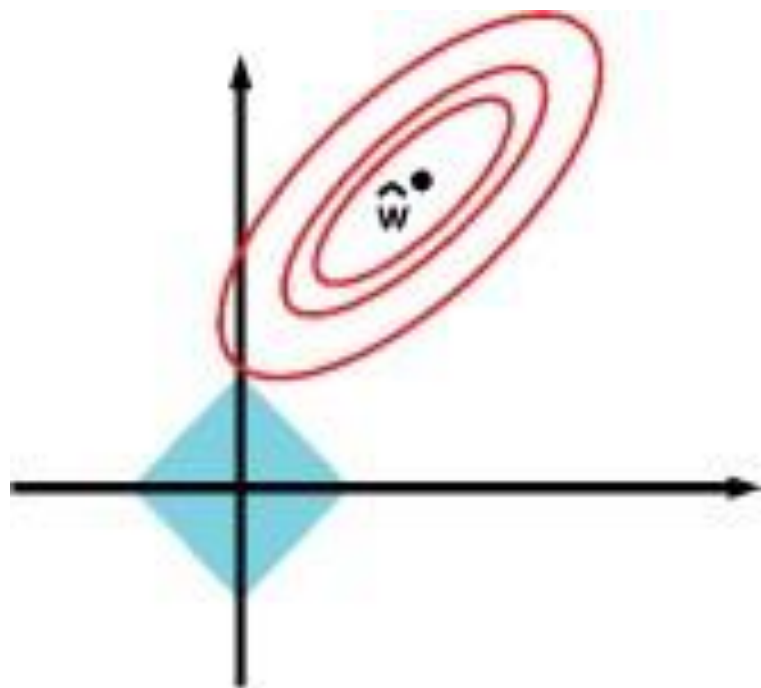
- 惩罚项有选择地收缩了某些系数
- 可以被用来做特征选择
- 比岭回归收敛速度慢

# 套索回归对模型参数的效果



$$J(\beta) = \frac{1}{2m} \sum_{i=1}^m (y_{\beta}(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n |\beta_j|$$

# L1与L2正则化



# ElasticNet正则化

$$J(\beta) = \frac{1}{2m} \sum_{i=1}^m (y_{\beta}(x^{(i)}) - y^{(i)})^2 + \lambda_1 \sum_{j=1}^n |\beta_j| + \lambda_2 \sum_{j=1}^n \beta_j^2$$

- 岭回归和套索回归的综合，用以平衡稀疏和平滑两个问题
- 需要调节额外的参数，来分配L1和L2正则化惩罚项的比例



# 超参数及其优化

- 正则化系数 ( $\lambda_1$  和  $\lambda_2$ ) 是根据经验决定的

使用测试数据调节  $\lambda$ ?

	Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22	The Hunger Games: Catching Fire	130000000	424068047	Francis Lawrence	PG-13	146
1	2013-05-03	Iron Man 3	200000000	409513994	Shane Black	PG-13	121
2	2013-11-29	Frozen	130000000	401334305	Chris BuckJennifer Lee	PG	103
3	2013-07-03	Dispicable Me 2	75000000	368901285	Pierre CoffinChris Renaud	PG	98
4	2013-06-14	Man of Steel	220000000	66895518	Zack Snyder	PG-13	143
5	2013-10-04	Gravity			Guillermo del Toro	PG-13	91
6	2013-08-21	Monsters University			Dan Abrahams	G	107
7	2013-12-13	The Hobbit: The Desolation of Smaug			Jackson	PG-13	161
8	2013-05-24	Fast & Furious 6			Justin Lin	PG-13	130
9	2013-03-08	On the Beach and Beyond	215000000	294911825	Sam Raimi	PG	127
10	2013-05-16	Star Trek Into Darkness	380000000	368773981	J.J. Abrams	PG-13	123
11	2013-11-08	Thor: The Dark World	650000000	649362140	Alan Taylor	PG-13	120
12	2013-05-21	World War Z	780000000	200369711	Marc Forster	PG-13	118
13	2013-03-02	The Croods			Kristen BreaChris Sanders	PG	98
14	2013-04-26	The Heat			John Dahl	R	117
15	2013-05-07	We're the Millers			Mark Waters	R	110
16	2013-12-13	American Hustle			Jesse EisenbergDavid O. Russell	R	138
17	2013-05-10	The Great Gatsby	100000000	145494712	Baz Luhrmann	PG-13	143

训练数据

测试数据

# 超参数及其优化

- 正则化系数 ( $\lambda_1$  和  $\lambda_2$ ) 是根据经验决定的

测试数据集来调节  $\lambda_1$  和  $\lambda_2$

使用测试数据集调节  $\lambda$ ?

	Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22	The Hunger Games: Catching Fire	150000000	424000000	Francis Lawrence	PG-13	146
1	2013-05-03	Iron Man 3			Shane Black	PG-13	121
2	2013-11-29	Frozen			Chris Buck, Jennifer Lee	PG	103
3	2013-07-03	Dispicable Me 2			Coffee-Olin Rosend	PG	98
4	2013-06-14	Man of Steel				PG-13	143
5	2013-10-04	Gravity				PG-13	91
6	2013-06-21	Monsters University				G	107
7	2013-12-13	The Hobbit: The				PG-13	141
8	2013-05-24	Fant & Furious 6				PG-13	130
9	2013-03-08	On The Green on				PG	127
10	2013-05-16	Star Trek Into Di				PG-13	123
11	2013-11-08	Thor: The Dark W				PG-13	120
12	2013-05-21	World War Z				PG-13	116
13	2013-03-02	The Croods			Michael Chris Sanders	PG	98
14	2013-04-26	The Heat				R	117
15	2013-05-07	We're the Millers			Marshall Thrasher	R	110
16	2013-12-13	American Hustle			Russell	R	138
17	2013-05-10	The Great Gatsby	105000000	145000000	Baz Luhrmann	PG-13	143

# 超参数及其优化

- 正则化系数 ( $\lambda_1$  和  $\lambda_2$ ) 是根据经验决定的
- 想让模型泛化---不要使用测试数据集来调节  $\lambda_1$  和  $\lambda_2$
- 划分出另一个数据集来调节超参数---验证集 (validation set)

用交叉验证来调节  $\lambda$

id	date	title	genres	duration	director	rating	votes
0	2013-11-22	The Hunger Games: Catching Fire	131000000	424008047	Francis Lawrence	PG-13	146
1	2013-05-03	Iron Man 3	200000000	408013994	Shane Black	PG-13	129
2	2013-11-29	Frozen			Jackie Miller Lee	PG	108
3	2013-07-03	Dispicable Me 2			Jeff-Orin Rensud	PG	98
4	2013-06-14	Man of Steel			Wyler	PG-13	143
5	2013-10-04	Gravity			Guaron	PG-13	91
6	2013-05-21	Monsters University	NaN	269492764	Den Scanlon	G	107
7	2013-05-03	Fast & Furious 6			John Singleton	PG-13	130
8	2013-05-24	Fast & Furious 6			John Singleton	PG-13	130
9	2013-03-08	On The Green and For			ine	PG	127
10	2013-05-16	Star Trek Into Darkness			James	PG-13	123
11	2013-11-08	Thor: The Dark World			Asa	PG-13	120
12	2013-05-21	Monsters U	130000000	408000000	John Singleton	PG-13	116
13	2013-03-29	The Croods			Chris Sanders	PG	98
14	2013-04-26	The Heat			li	R	117
15	2013-05-07	We're the Millers			Marshall Thurber	R	110
16	2013-12-13	American Hustle			Russell	R	138
17	2013-05-10	The Great Gatsby	105000000	148840474	Bez Luemann	PG-13	143

# 线性回归实例

——李宏毅2020机器学习深度学习



# 线性回归实例

## Step 1: Model

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



w and b are parameters  
(can be any value)

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

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

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

.....



Linear model:

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

# 线性回归实例

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

A set of  
function

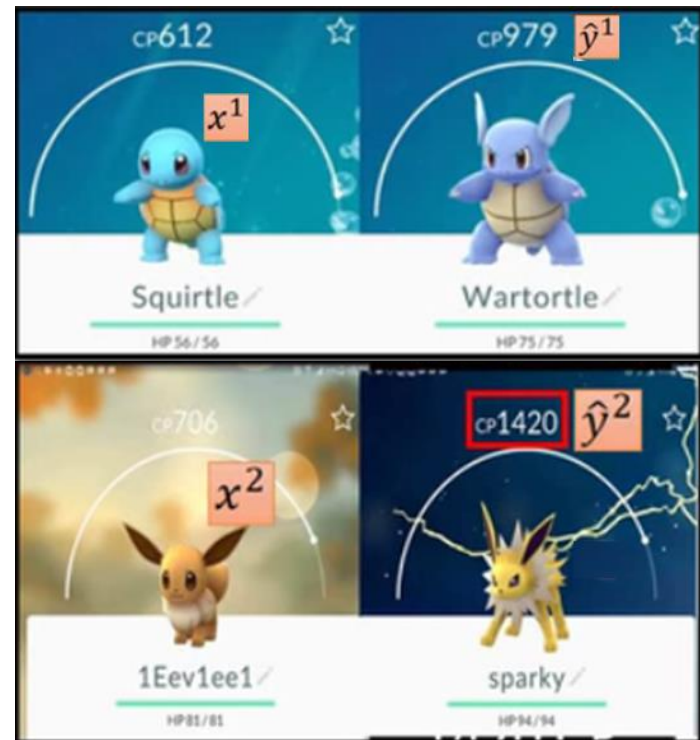
Model

$f_1, f_2 \dots$

Training  
Data

输入

输出



# 线性回归实例

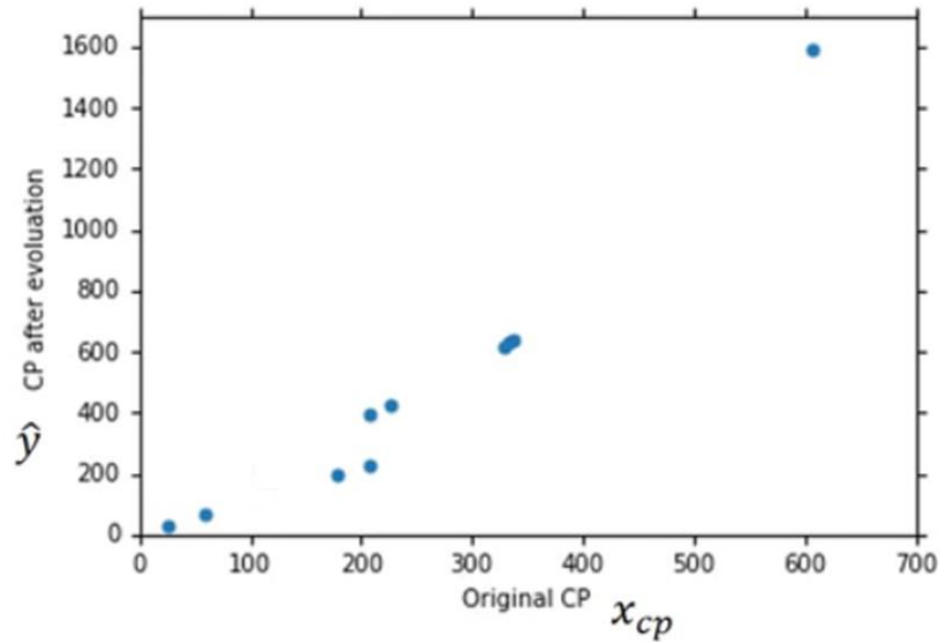
Training Data:  
10 pokemons

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

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

$\vdots$

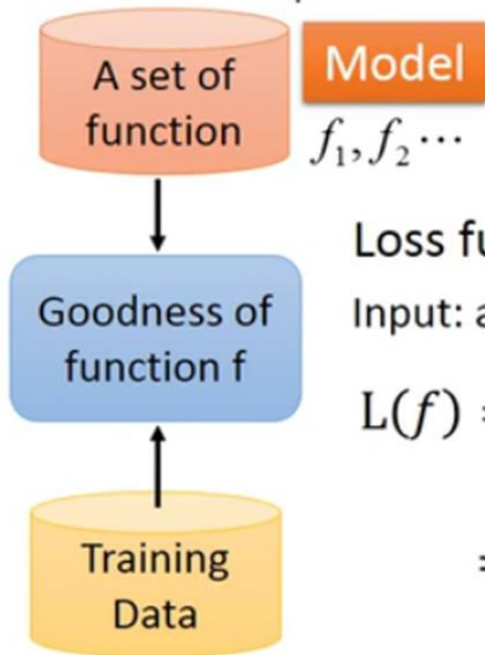
$$(x^{10}, \hat{y}^{10})$$



# 线性回归实例

## Step 2: Goodness of Function

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



Loss function  $L$ :

Input: a function, output: how bad it is

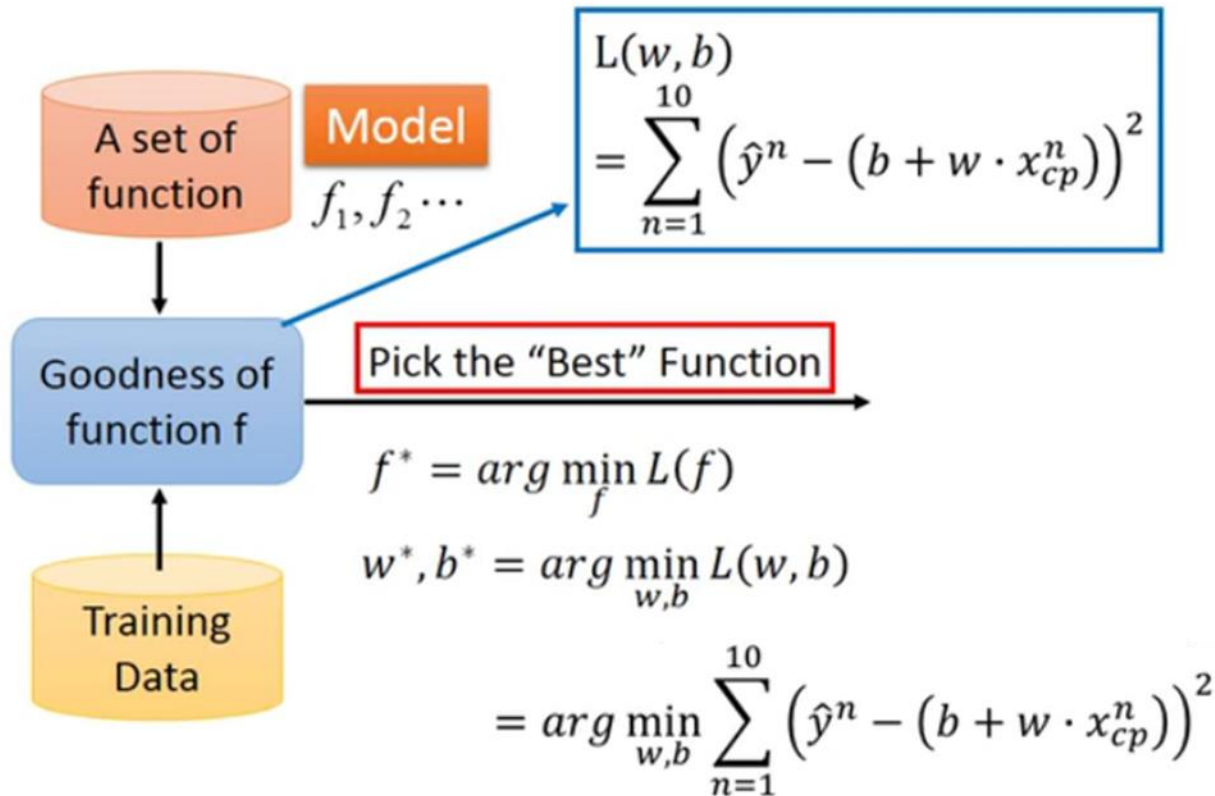
$$L(f) = L(w, b)$$

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



# 线性回归实例

## Step 3: Best Function



# 线性回归实例

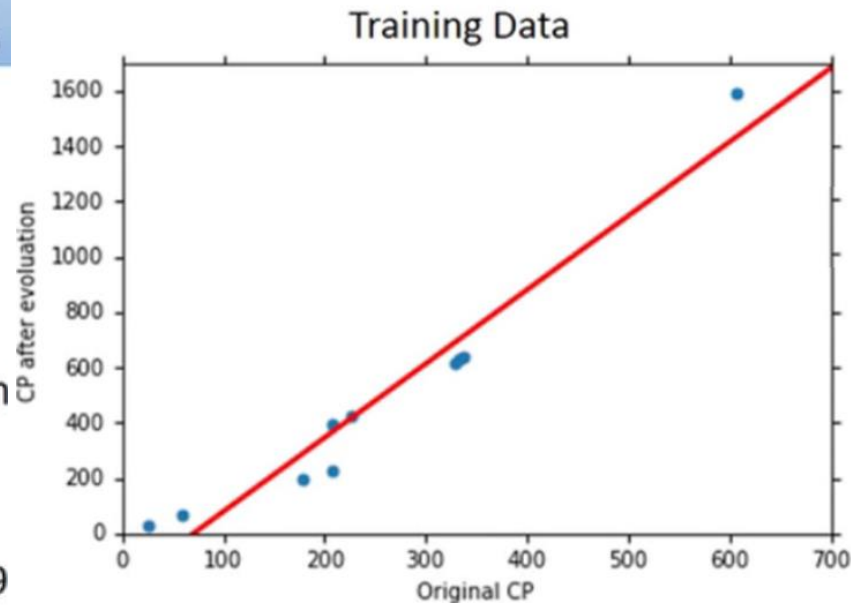
$$y = b + w \cdot x_{cp}$$

$$b = -188.4$$

$$w = 2.7$$

Average Error on  
Training Data

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



# 线性回归实例

How's the results?  
- Generalization

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

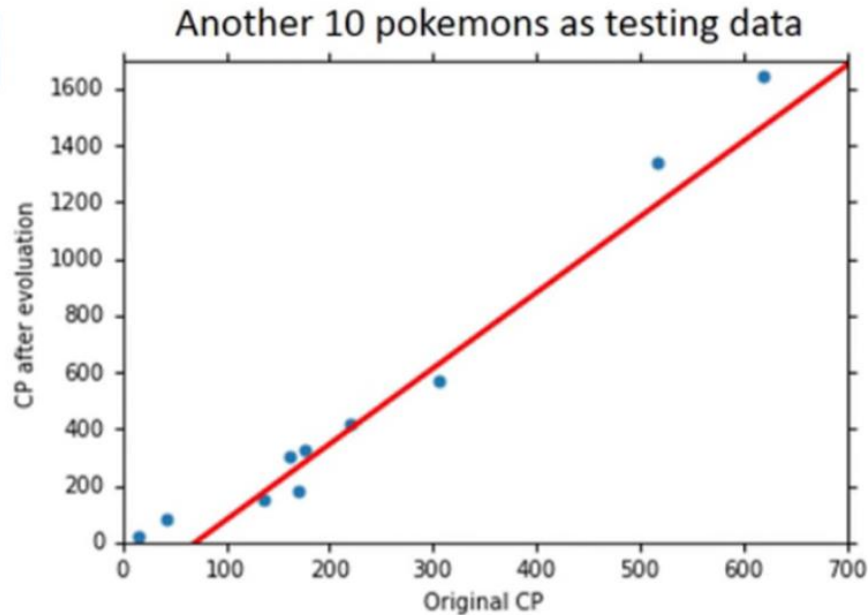
$$y = b + w \cdot x_{cp}$$

$$b = -188.4$$

$$w = 2.7$$

Average Error on Testing Data

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



# 线性回归实例

## Selecting another Model

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

## Best Function

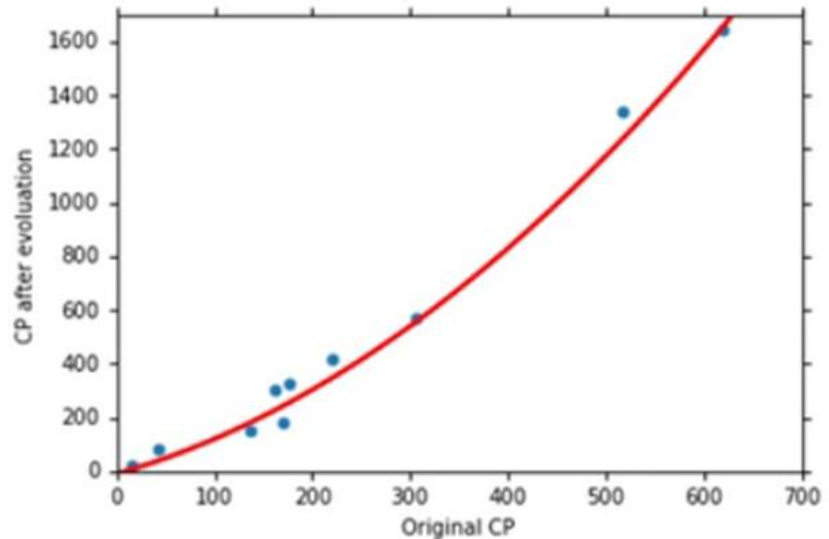
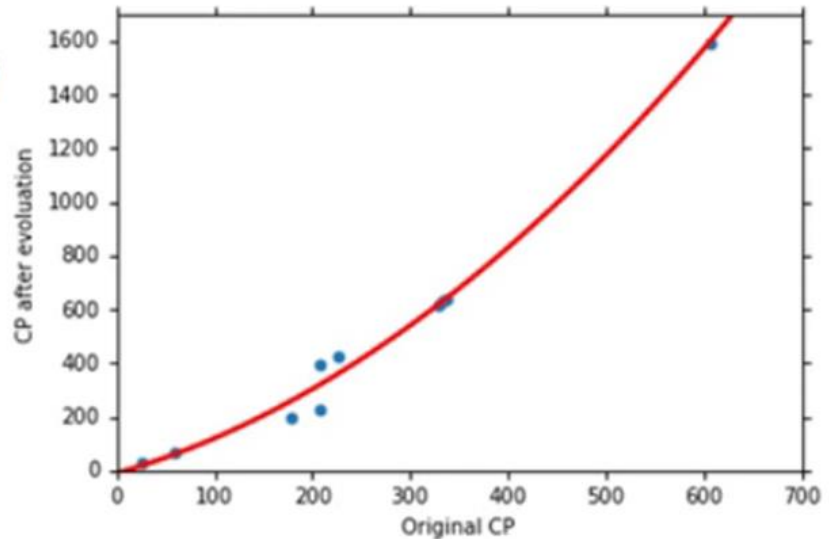
$$b = -10.3$$

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

$$\text{Average Error} = 15.4$$

## Testing:

$$\text{Average Error} = 18.4$$



# 线性回归实例

## Selecting another Model

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

## Best Function

$$b = 6.4, w_1 = 0.66$$

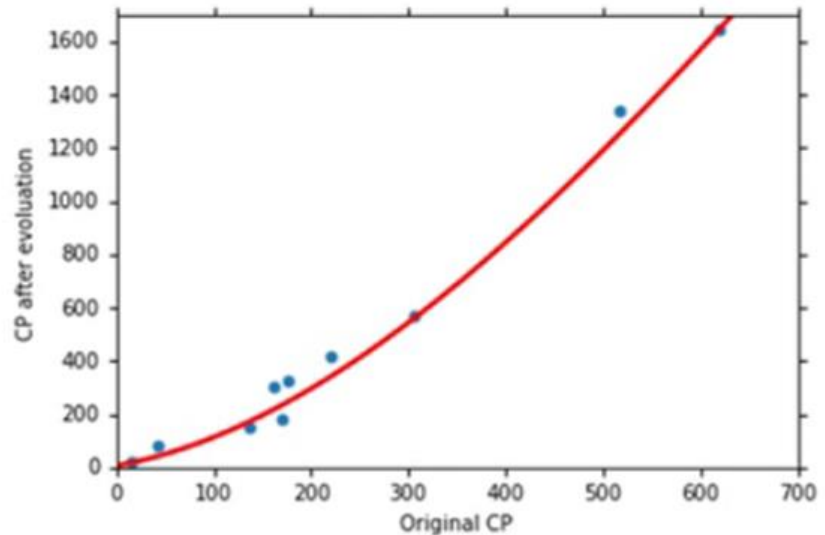
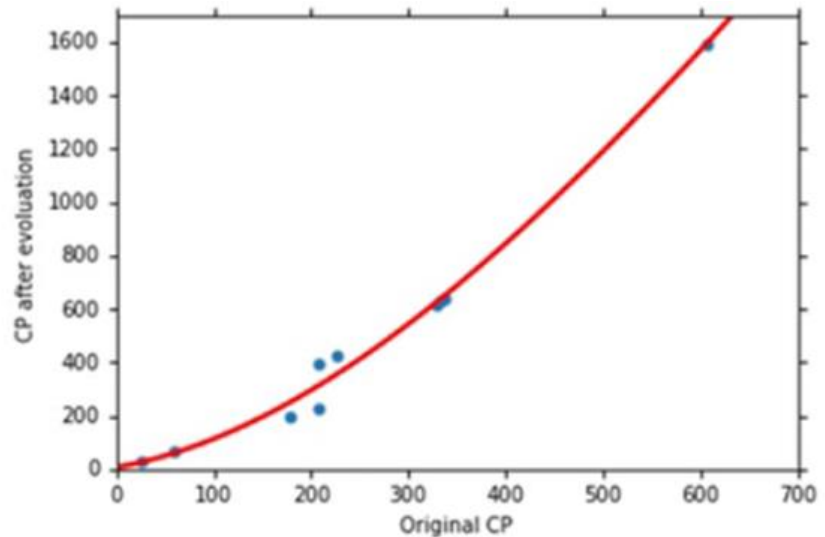
$$w_2 = 4.3 \times 10^{-3}$$

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

Average Error = 15.3

## Testing:

Average Error = 18.1



# 线性回归实例

## Selecting another Model

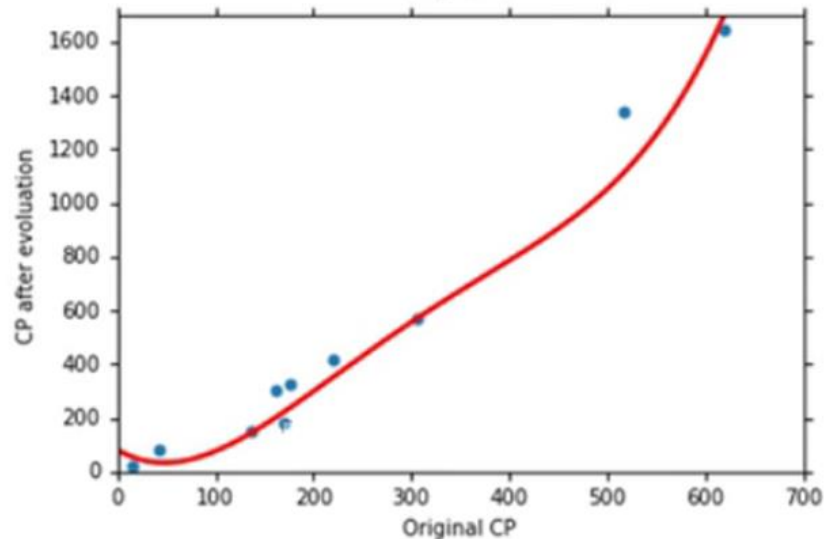
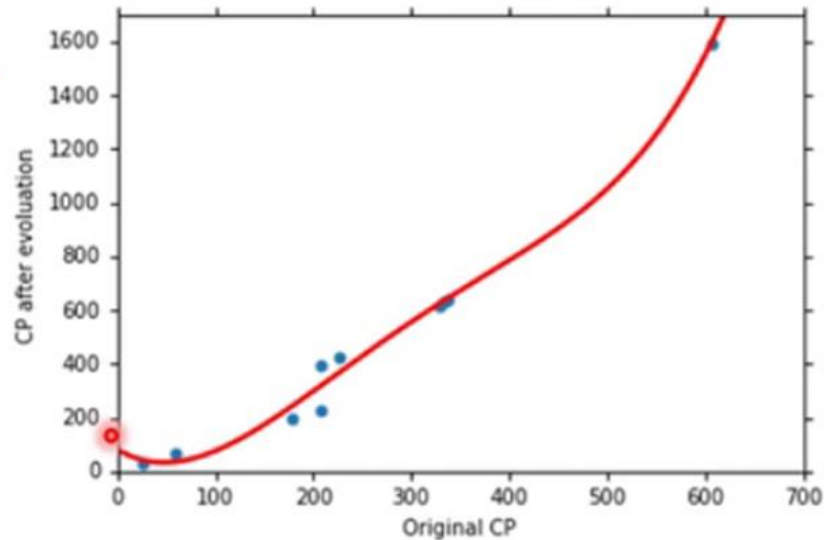
$$y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2 + w_3 \cdot (x_{cp})^3 + w_4 \cdot (x_{cp})^4$$

## Best Function

Average Error = 14.9

## Testing:

Average Error = 28.8



# 线性回归实例

## Selecting another Model

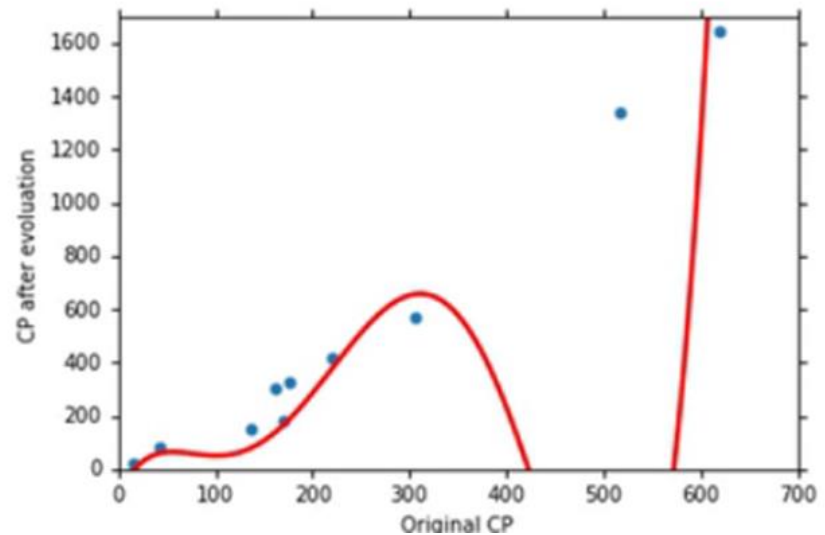
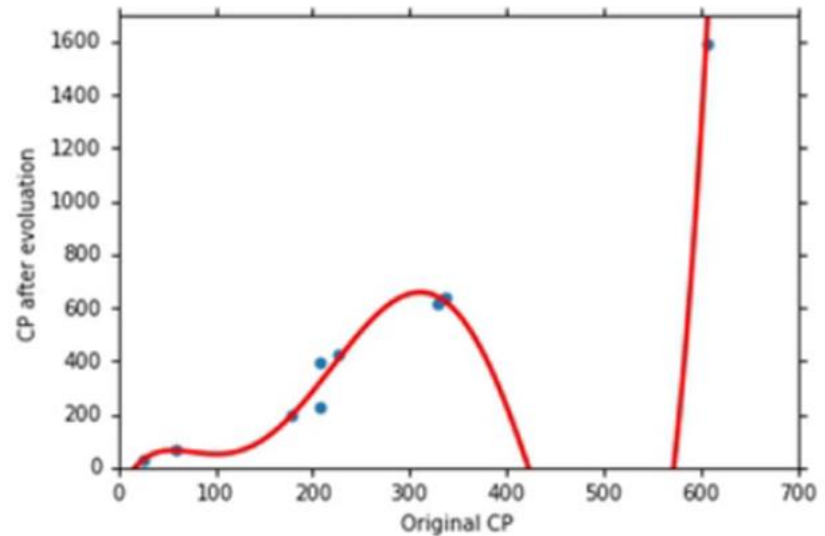
$$y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2 + w_3 \cdot (x_{cp})^3 + w_4 \cdot (x_{cp})^4 + w_5 \cdot (x_{cp})^5$$

## Best Function

Average Error = 12.8

## Testing:

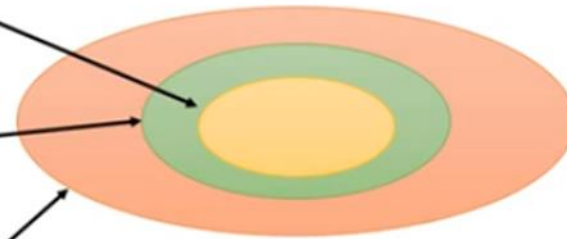
Average Error = 232.1



# 线性回归实例

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

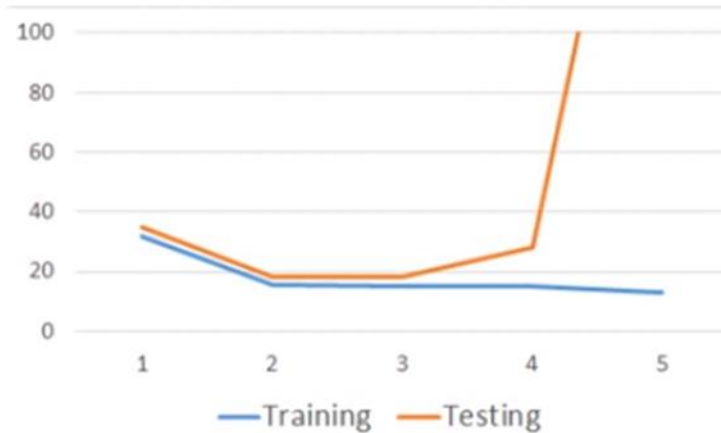


A more complex model yields lower error on training data.



# 线性回归实例

## 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

# 线性回归实例

## 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

# 线性回归实例

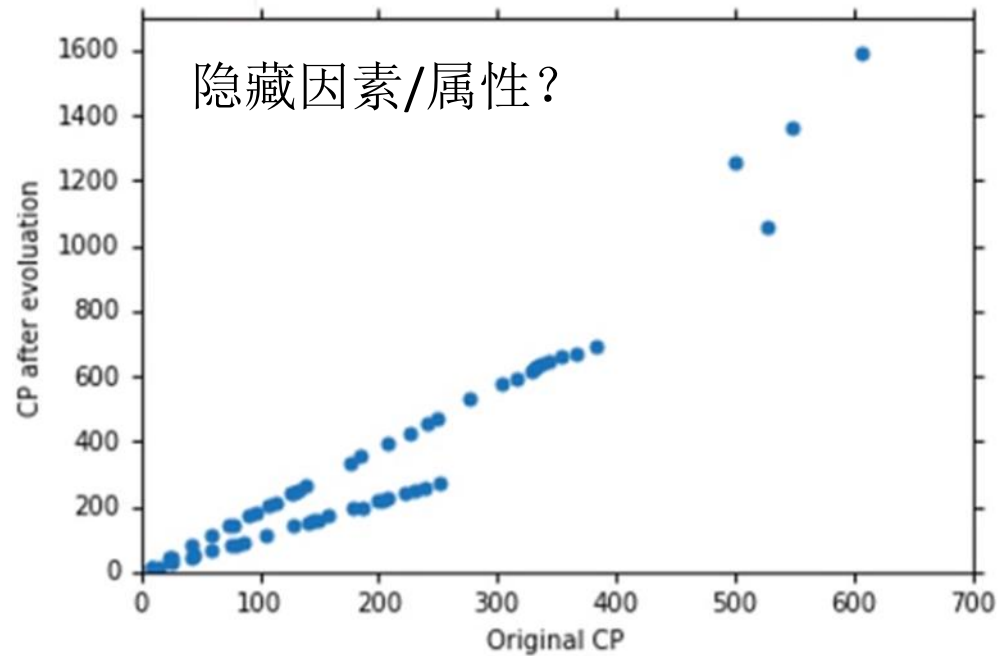
## Model Selection



	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

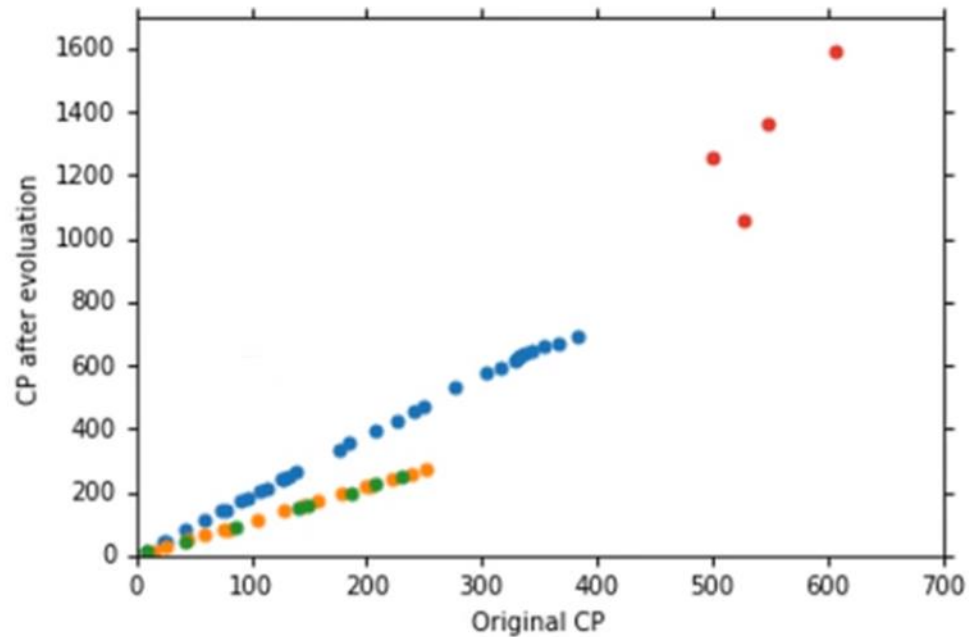
# 线性回归实例

Let's collect more data (60只)



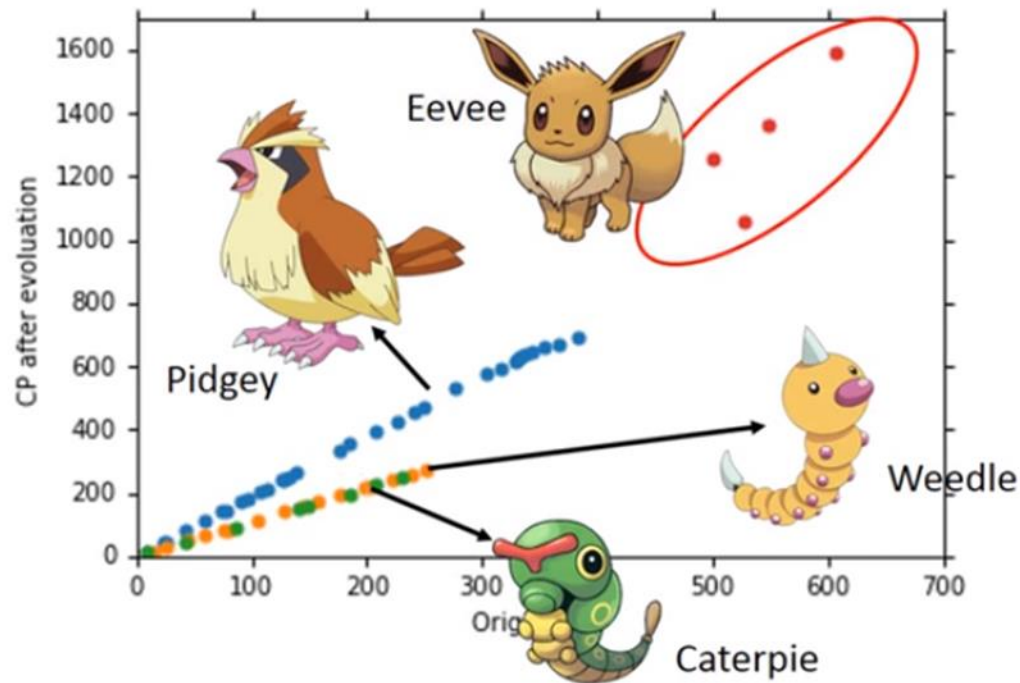
# 线性回归实例

What are the hidden factors?



# 线性回归实例

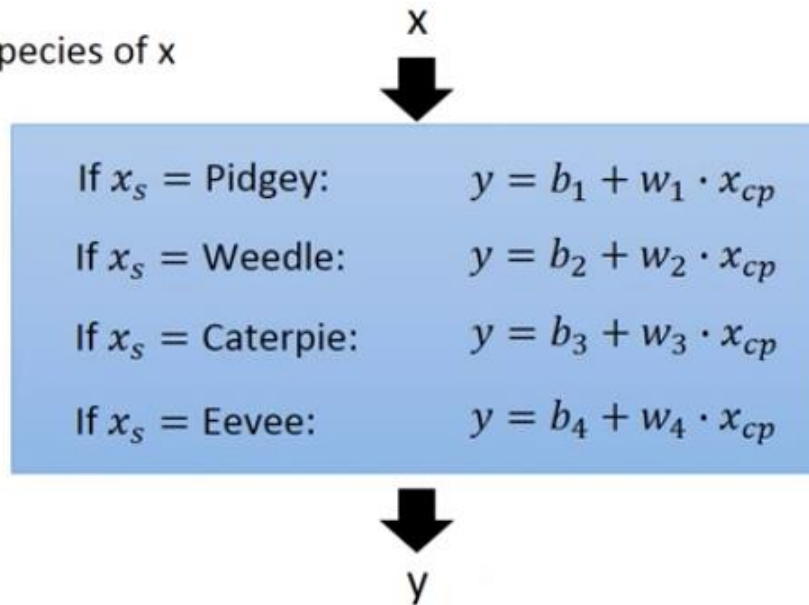
What are the hidden factors?



# 线性回归实例

Back to step 1:  
Redesign the Model

$x_s$  = species of  $x$



# 线性回归实例

Back to step 1:  
Redesign the Model

$$\begin{aligned} y = & b_1 \cdot \delta(x_s = \text{Pidgey}) \\ & + w_1 \cdot \delta(x_s = \text{Pidgey})x_{cp} \\ & + b_2 \cdot \delta(x_s = \text{Weedle}) \\ & + w_2 \cdot \delta(x_s = \text{Weedle})x_{cp} \\ & + b_3 \cdot \delta(x_s = \text{Caterpie}) \\ & + w_3 \cdot \delta(x_s = \text{Caterpie})x_{cp} \\ & + b_4 \cdot \delta(x_s = \text{Eevee}) \\ & + w_4 \cdot \delta(x_s = \text{Eevee})x_{cp} \end{aligned}$$

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

Linear model?

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

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

If  $x_s = \text{Pidgey}$



# 线性回归实例

Back to step 1:  
Redesign the Model

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

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

Linear model?

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

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

If  $x_s = \text{Pidgey}$

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

# 线性回归实例

Back to step 1:  
Redesign the Model

$$\begin{aligned} y = & b_1 \cdot \delta(x_s = \text{Pidgey}) \\ & + w_1 \cdot \delta(x_s = \text{Pidgey})x_{cp} \\ & + b_2 \cdot \delta(x_s = \text{Weedle}) \\ & + w_2 \cdot \delta(x_s = \text{Weedle})x_{cp} \\ & + b_3 \cdot \delta(x_s = \text{Caterpie}) \\ & + w_3 \cdot \delta(x_s = \text{Caterpie})x_{cp} \\ & + b_4 \cdot \delta(x_s = \text{Eevee}) \\ & + w_4 \cdot \delta(x_s = \text{Eevee})x_{cp} \end{aligned}$$

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

Linear model?

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

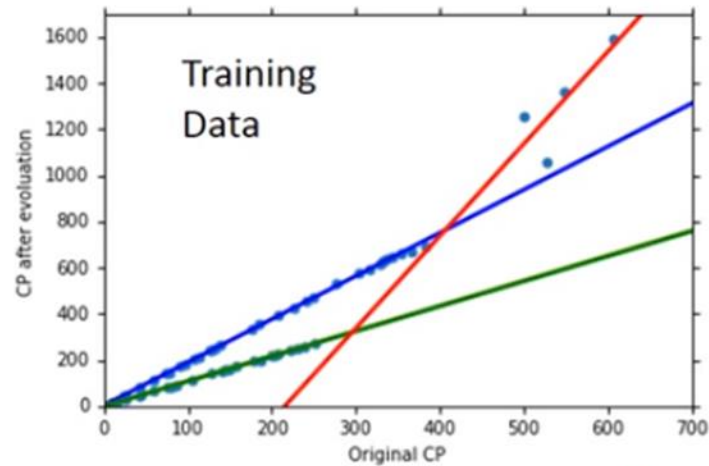
$$\begin{cases} =1 & \text{If } x_s = \text{Pidgey} \\ =0 & \text{otherwise} \end{cases}$$

If  $x_s = \text{Pidgey}$

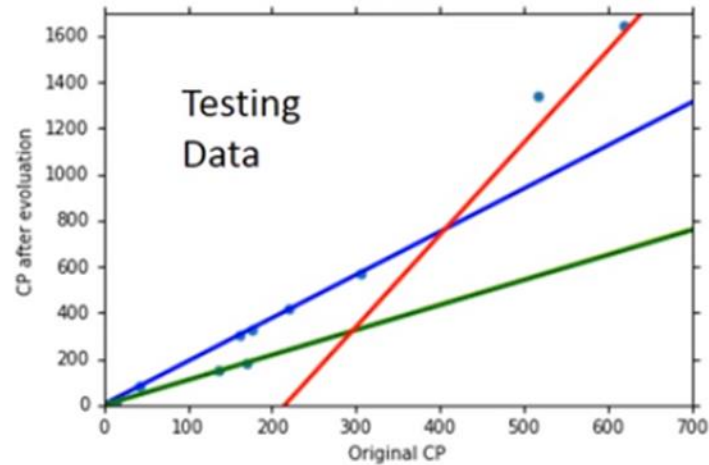
$$y = b_1 + w_1 \cdot x_{cp}$$

# 线性回归实例

Average error  
= 3.8



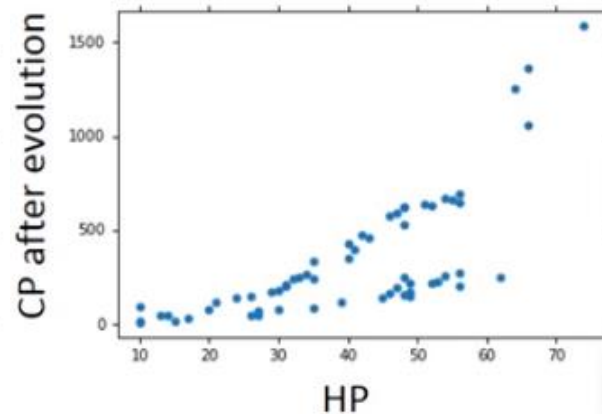
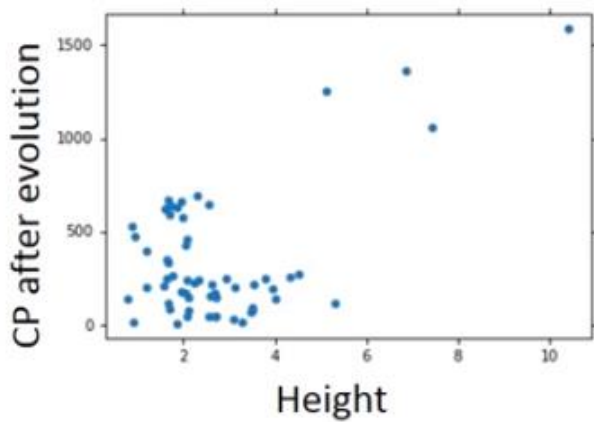
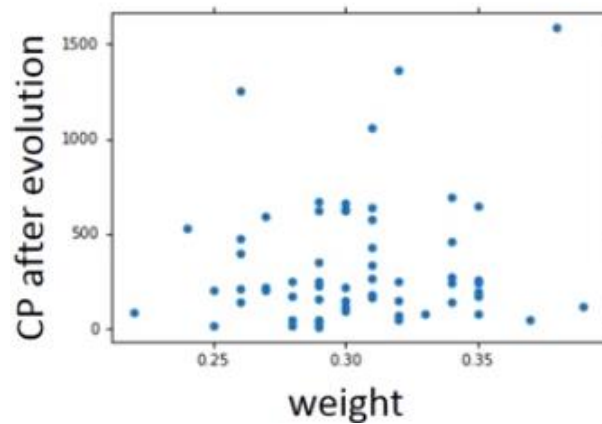
Average error  
= 14.3





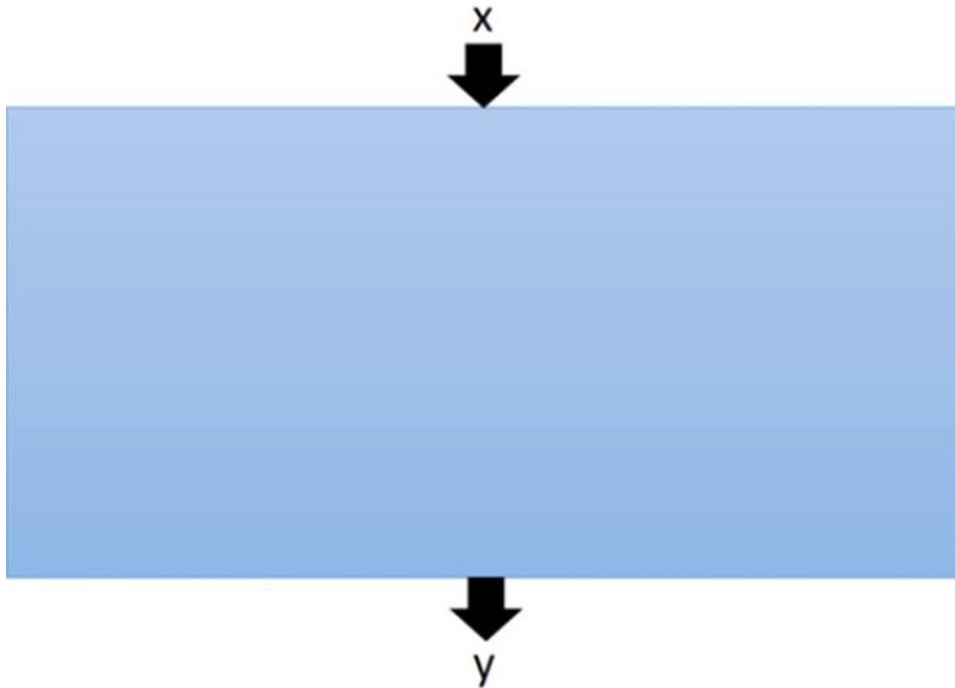
# 线性回归实例

Are there any other hidden factors?



# 线性回归实例

Back to step 1:  
Redesign the Model Again



# 线性回归实例

Back to step 1:  
Redesign the Model Again



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



Training Error  
= 1.9

Testing Error  
= 102.3

Overfitting!

# 线性回归实例

Back to step 2: Regularization

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

The functions with  
smaller  $w_i$  are better

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

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

➤ Why smooth functions are preferred?

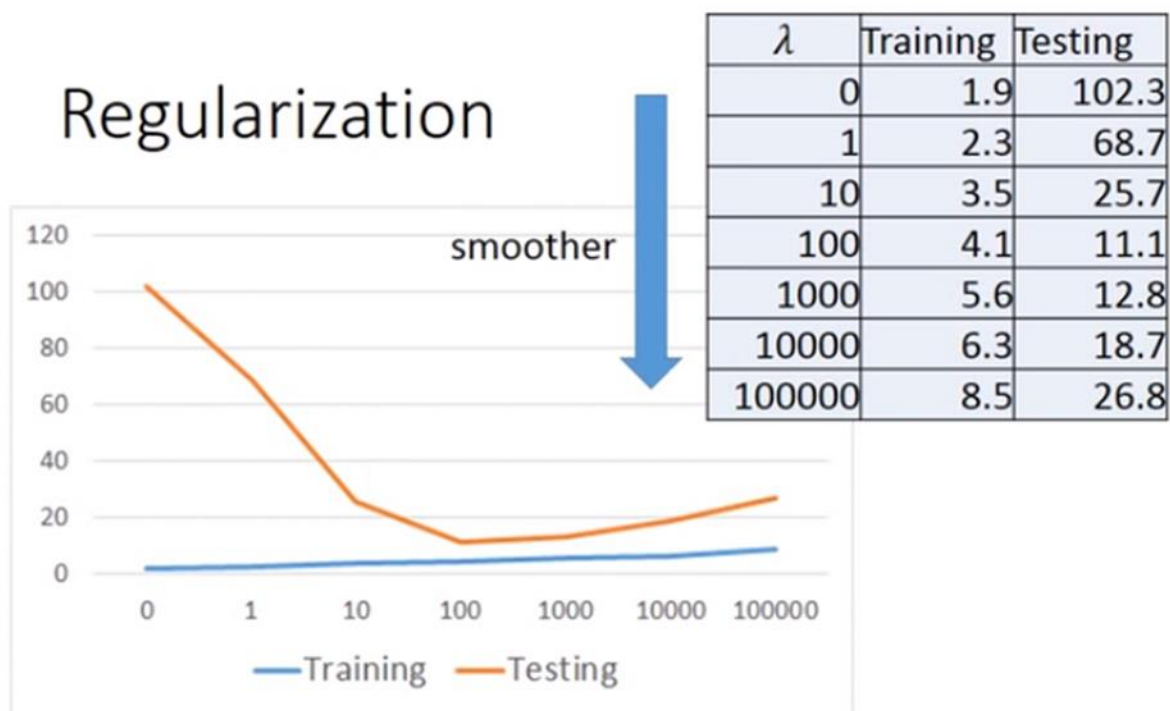
$$y = b + \sum w_i x_i$$

$+w_i \Delta x_i$   $+ \Delta x_i$

➤ If some noises corrupt input  $x_i$  when testing

A smoother function has less influence.

# 线性回归实例



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