



# 数据挖掘导论

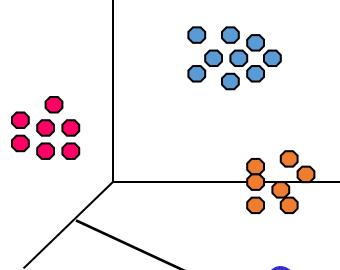
## 第五章：回归

王静远

北京航空航天大学

# 回归分析模型的概述

# 数据挖掘的任务类型



聚类

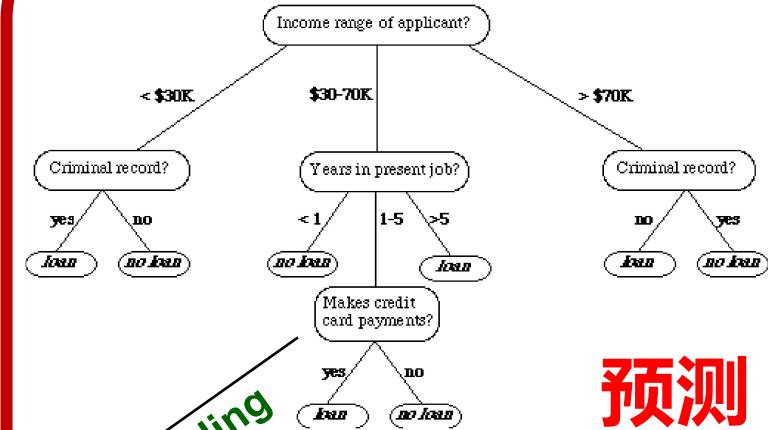
Clustering

关联分析

Association Analysis



Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes
11	No	Married	60K	No
12	Yes	Divorced	220K	No
13	No	Single	85K	Yes
14	No	Married	75K	No
15	No	Single	90K	Yes

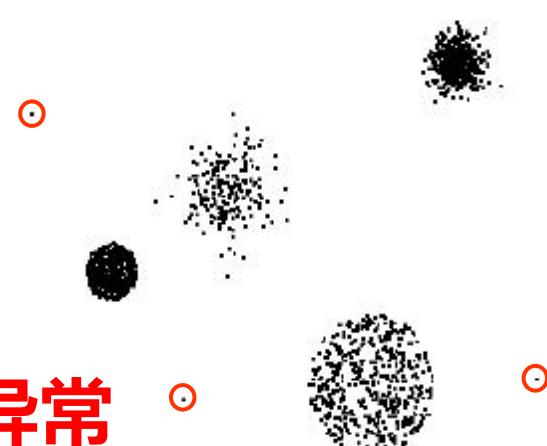


预测建模

Predictive Modeling

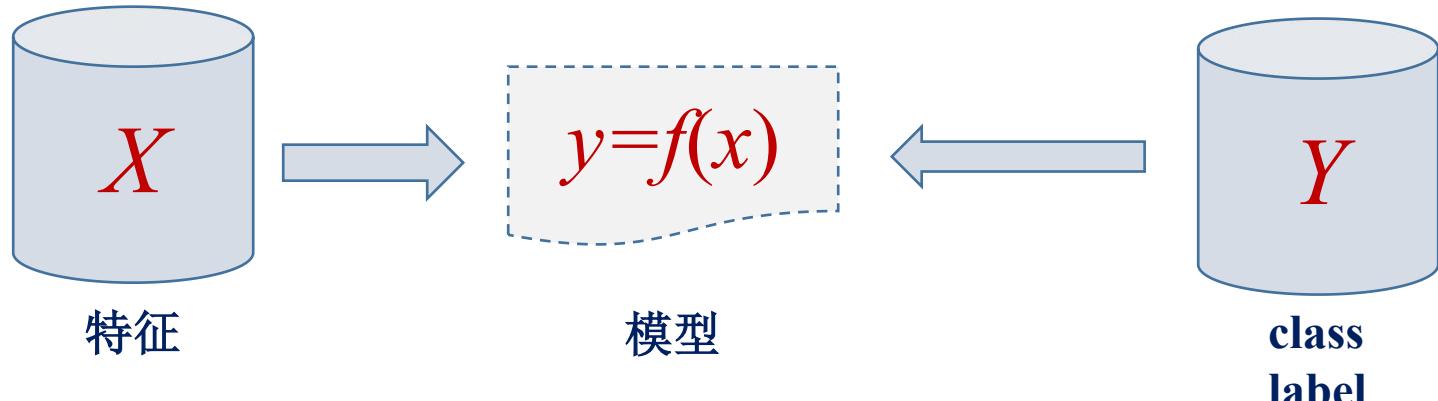
异常检测

Anomaly Detection



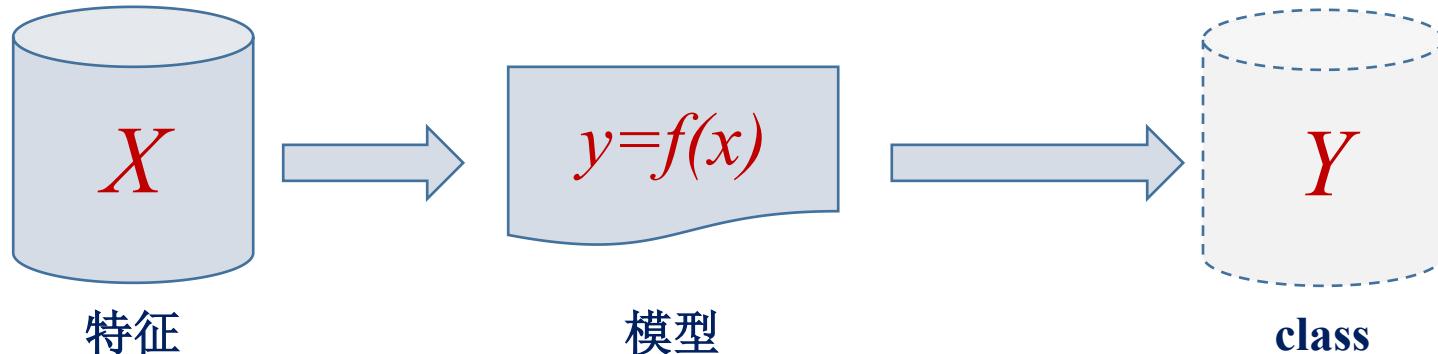
# 回归问题

## 模型训练



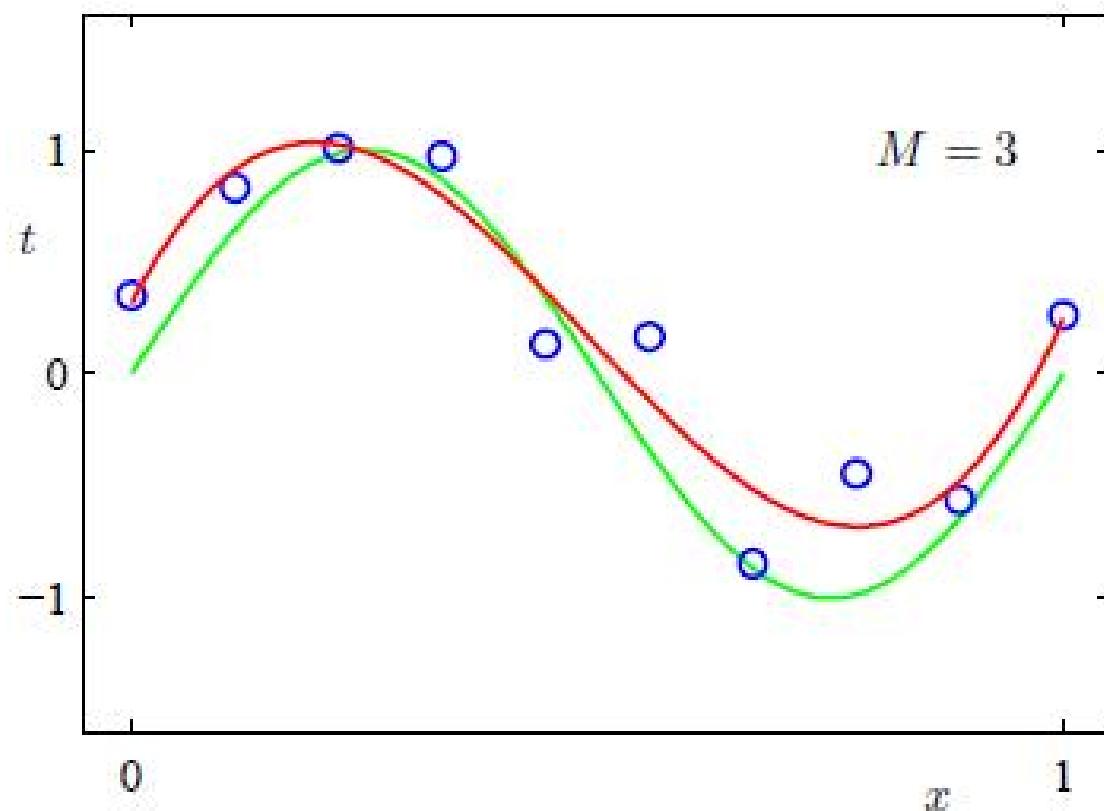
给定数据集 $\{X, Y\}$ , 寻找 $y=f(x)$ 的相关关系

## 模型使用



# 回归问题

- 使用模型函数去拟合数据的生成过程函数



数据的生成过程

$$y = \sin(2\pi x) + \epsilon$$



使用多项式函数进行拟合

$$y = ax + bx^2 + cx^3 + d$$

# 回归分析的定义

- 在统计模型中，**回归分析** (Regression Analysis) 是研究一个因变量  $Y$  关于另一个 (或多个) 自变量  $X$  的依赖关系的计算方法和理论
- 给定一组数据  $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_M, y_M)\}$ ，其中  $x_m$  是**解释变量**， $y_m$  是对应的**被解释变量**。回归分析的目的是寻找一个函数来根据  $x_m$  估计  $y_m$ ，即

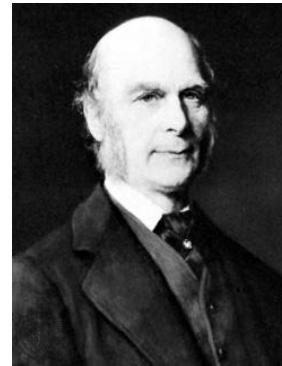
$$\hat{y}_m = f_{\theta}(x_m)$$

# 回归模型的建模要素

---

- **回归函数**: 建模自变量  $x$  和因变量  $y$  之间的关系
- **损失函数**: 评估预测效果的好坏，也称代价函数
- **参数求解**: 在所有可能的参数  $\theta$  中，能够使损失函数最小的  $\theta$  是最优参数

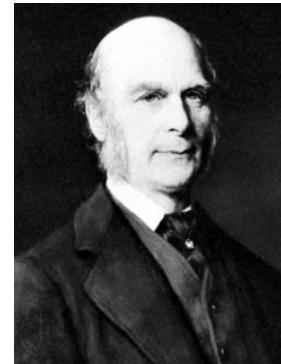
# 回归问题的由来



Francis Galton  
1822~1911  
达尔文的表弟

**Regression** 这个词之所以被用到统计学领域，最初是因为 **弗朗西斯·高尔顿 (Francis Galton)** 十九世纪末发表的一篇题为《遗传身高向平均回归》的论文 (*Regression towards mediocrity in hereditary stature*)。在这篇论文里，高尔顿研究了父母的身高（与父辈身高平均值的差）对儿子的身高（与子辈身高平均值的差）的影响。

# 回归问题的由来



Francis Galton  
1822~1911  
达尔文的表弟

**弗朗西斯·高尔顿 (Francis Galton) 和他的学生卡尔·皮尔逊 (Karl Pearson)** 通过观察 1078 对夫妇的身高数据，分析了儿子身高与父母身高之间的关系。他们发现父母的身高可以预测儿子的身高：儿子的身高  $y$  与父母的身高  $x$  大致可归结为以下关系：

$$y = 0.516 * x + 0.8567 \text{ (单位为米)}$$

父母身高每增加 1 个单位，其成年儿子的身高平均增加 0.516 个单位

# 回归问题的由来

$$y = 0.516 * x + 0.8567 \text{ (单位为米)}$$

父母身高每增加 1 个单位，其成年儿子的身高平均增加 0.516 个单位

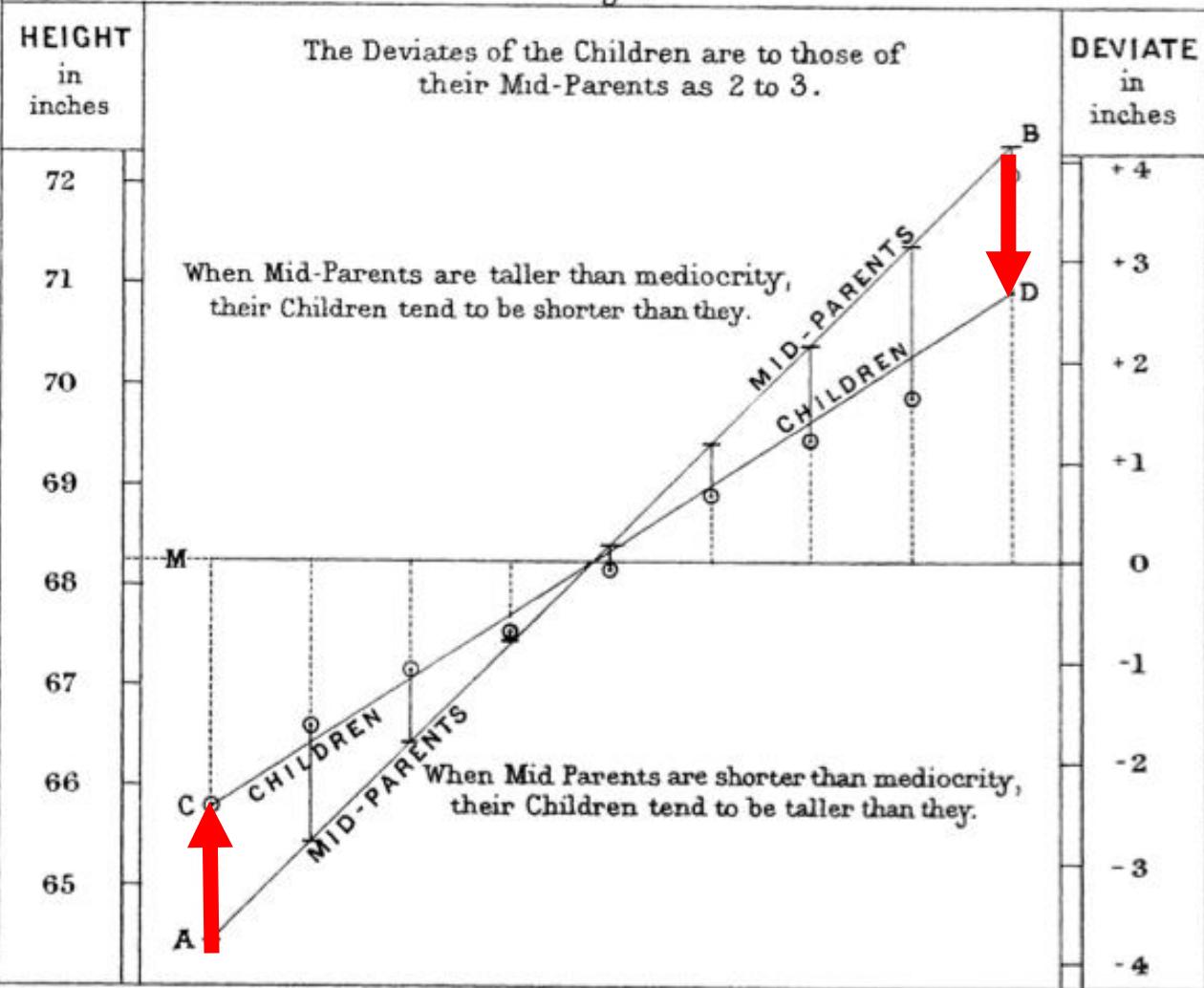
- 该公式有一个有趣的性质：当  $x$  大于 1.77 米时  $y$  会小于  $x$ ，也就是子辈的身高会低于父辈；当  $x$  小于 1.77 米时  $y$  会大于  $x$ ，也就是子辈的身高会高于父辈。即：**矮个父母所生的儿子比其父母要高，身材较高的父母所生子女的身高却下降到多数人的平均身高（父母1.5m，子女1.64m；父母2m，子女1.88m）。**
- 换言之，无论父辈的身高是高于平均身高，还是低于平均身高，子辈的身高都会趋向于“向均值回归”，这就是统计学上最初出现“回归”时的涵义。
- “均值回归”现象最早是通过建立变量  $x$  与变量  $y$  之间的关系模型的方式被发现的，后来人们就将“回归”这个名词保留了下来，特指这种使用解释变量预测被解释变量的建模方法。虽然其研究的问题已经不再局限于“向均值回归”现象。

# 回归问题的由来

Plate IX.

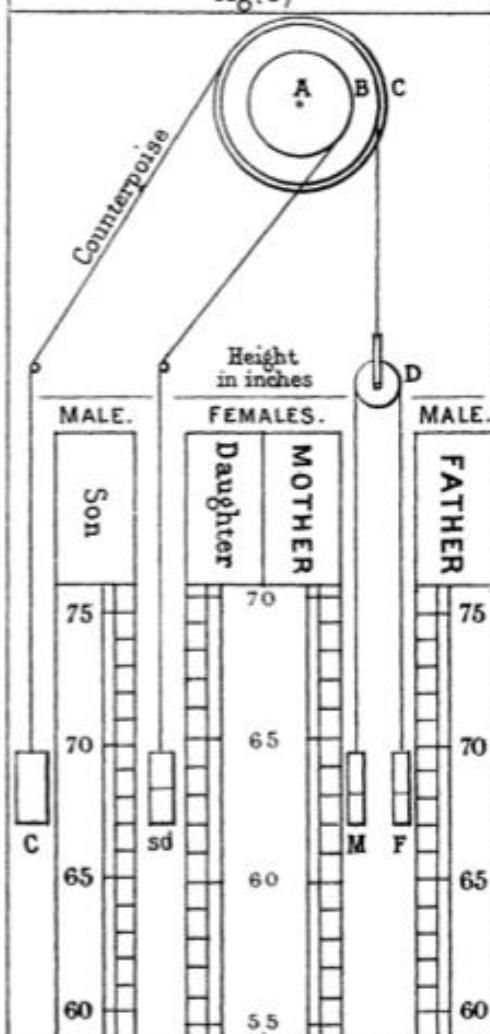
## RATE OF REGRESSION IN HEREDITARY STATURE.

Fig.(a)



## FORECASTER OF STATURE

Fig.(b)



# 线性回归模型

# 线性回归模型的回归函数

给定一个包含了  $M$  个样本的数据集：

$$\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \dots, (x_M, y_M)\},$$

其中  $x_m \in \mathbb{R}^N$  为  $N$  维列向量,  $y_m \in \mathbb{R}$  为标量,  $m = 1, 2, \dots, M$

- 使用自变量  $\mathbf{x}$  的**线性组合**预测因变量  $y$

$$\begin{aligned}\hat{y}_m &= f_{\theta}(x_m) = b + w_1 x_{m,1} + \dots + w_N x_{m,N} \\ &= b + \mathbf{w}^\top \mathbf{x}_m \\ &= \boldsymbol{\theta}^\top \tilde{\mathbf{x}}_m.\end{aligned}$$

# 线性回归模型的回归函数

$$\begin{aligned}\hat{y}_m &= f_{\theta}(\mathbf{x}_m) = b + w_1 x_{m,1} + \cdots + w_N x_{m,N} \\ &= b + \mathbf{w}^\top \mathbf{x}_m \\ &= \boldsymbol{\theta}^\top \tilde{\mathbf{x}}_m.\end{aligned}$$

- $\mathbf{w} = (w_1, \dots, w_N)^\top \in \mathbb{R}^N$  为  $N$  维列向量，称为权重 (Weight)；
- $b \in \mathbb{R}$  称为偏置 (Bias)；
- $\boldsymbol{\theta} = (b, w_1, \dots, w_N)^\top$  称为参数 (Parameter)；
- $\tilde{\mathbf{x}}_m$  是在  $\mathbf{x}_m$  的第一个分量前加上 1 形成的，即  $\tilde{\mathbf{x}}_m = (1, x_{m,1}, \dots, x_{m,N})^\top$ ；
- $\hat{y}_m$  代表模型对真实值  $y_m$  的预测或估计。

# 线性回归模型的损失函数

- 线性回归使用**均方误差** (Mean Square Error, 简称 MSE) 作为损失函数

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{M} \sum_{m=1}^M (\hat{y}_m - y_m)^2 = \frac{1}{M} \sum_{m=1}^M (\boldsymbol{\theta}^\top \tilde{\mathbf{x}}_m - y_m)^2$$

- 均方误差也是因变量为连续值的回归模型（也就是在机器学习或数据挖掘中的回归问题）最常用的损失函数

# 线性回归模型的参数求解

- 优化目标：**最小化损失函数**
- 方法：**最小二乘法**（本质是求函数极值）

- 定义自变量矩阵：

$$\tilde{\mathbf{X}} = \begin{pmatrix} 1 & x_{1,1} & \cdots & x_{1,N} \\ 1 & x_{2,1} & \cdots & x_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{M,1} & \cdots & x_{M,N} \end{pmatrix}$$

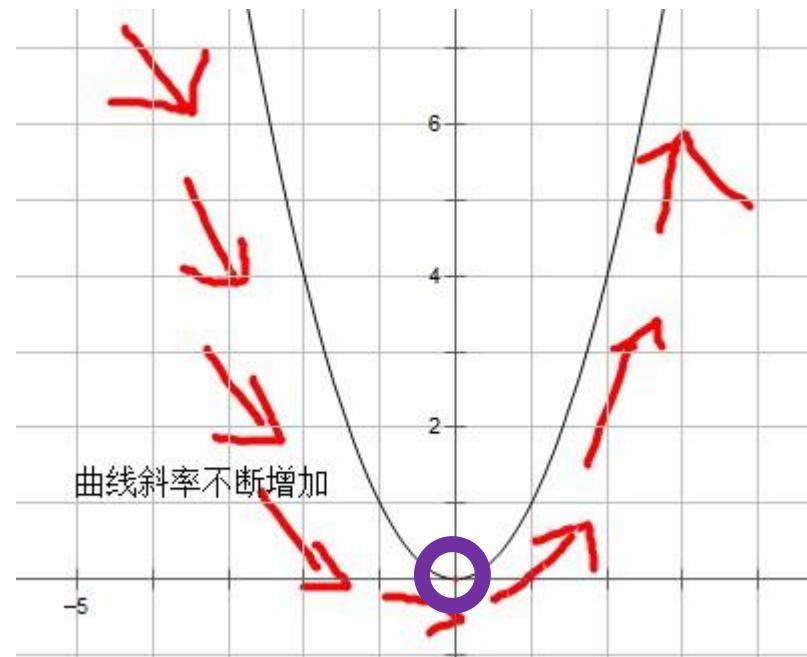
- 线性回归的损失函数可以表示为：

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) &= \sum_{m=1}^M (f_{\boldsymbol{\theta}}(\mathbf{x}_m) - y_m)^2 \\ &= \sum_{m=1}^M (\boldsymbol{\theta}^\top \tilde{\mathbf{x}}_m - y_m)^2 \\ &= (\tilde{\mathbf{X}} \boldsymbol{\theta} - \mathbf{y})^\top (\tilde{\mathbf{X}} \boldsymbol{\theta} - \mathbf{y})\end{aligned}$$

# 线性回归模型的参数求解

## 如何最小化?

- 极(小)值点的存在条件
- 一阶导数 = 0
- 二阶导数  $>= 0$



# 线性回归模型的参数求解

- 求一阶导  $\frac{\partial L(\theta)}{\partial \theta}$ ，令其为零，可得**极值点**为

$$\theta^* = (\tilde{X}^\top \tilde{X})^{-1} \tilde{X}^\top y$$

- 再求二阶导

$$\frac{\partial^2 \mathcal{L}(\theta)}{\partial \theta^2} = \frac{\partial (2\tilde{X}^\top \tilde{X}\theta)}{\partial \theta} - \frac{\partial (2\tilde{X}^\top y)}{\partial \theta} = 2\tilde{X}^\top \tilde{X}$$

- 根据正定矩阵的性质，当  $\tilde{X}$  是列满秩矩阵，则是正定矩阵。因此  $\theta = \theta^*$  时  $L$  取极小值。

# 矩阵求逆

$$(\mathbf{X}^\top \mathbf{X})^{-1}$$

使用相当多的栈空间

• 方法1：伴随矩阵法 → 复杂度  $O(n^4)$

• 需要计算  $n^2$  个元素的代数余子式  $A_{ij}$

• 计算某个元素的代数余子式时用递归，复杂度  $O(n^2)$

• 总体复杂度为  $O(n^4)$

$$\mathbf{A}^{-1} = \frac{1}{|\mathbf{A}|} \begin{bmatrix} A_{11} & A_{21} & \dots & A_{n1} \\ A_{12} & A_{22} & \dots & A_{n2} \\ \dots & \dots & \dots & \dots \\ A_{1n} & A_{2n} & \dots & A_{nn} \end{bmatrix}$$

某个元素的代数余子式

行列式

# 矩阵求逆

$$(\mathbf{X}^\top \mathbf{X})^{-1}$$

同样会使用相当多空间

• 方法2：LU分解法 → 复杂度  $O(n^3)$

• 矩阵的 LU 分解： $A = LU$ ，其中 L 为下三角矩阵，U 为上三角矩阵。则， $A^{-1} = (LU)^{-1} = U^{-1}L^{-1}$

→ 复杂度  $O(n^3)$

• 求解上三角矩阵的逆可以采用：解线程方程组  $\mathbf{Ax} = \mathbf{b}$  的方式求逆矩阵。 $\mathbf{b}$  分别取单位阵的各个列向量，所得到的解向量  $\mathbf{x}$  就是逆矩阵的各个列向量，拼成逆矩阵即可。

→ 复杂度  $O(n^2)$

• 要解  $2n$  个  $\mathbf{Ax} = \mathbf{b}$  的方程组 → 复杂度  $O(n^3)$

• 总体复杂度为  $O(n^3)$

# 最小二乘法的不足

- **计算复杂度问题：**矩阵求逆的复杂度较高 $O(n^3)$ ,  $n$ 是特征维度
- **无法求解问题：**

自变量样本矩阵  $\tilde{X}$  需要满足如下两个条件：

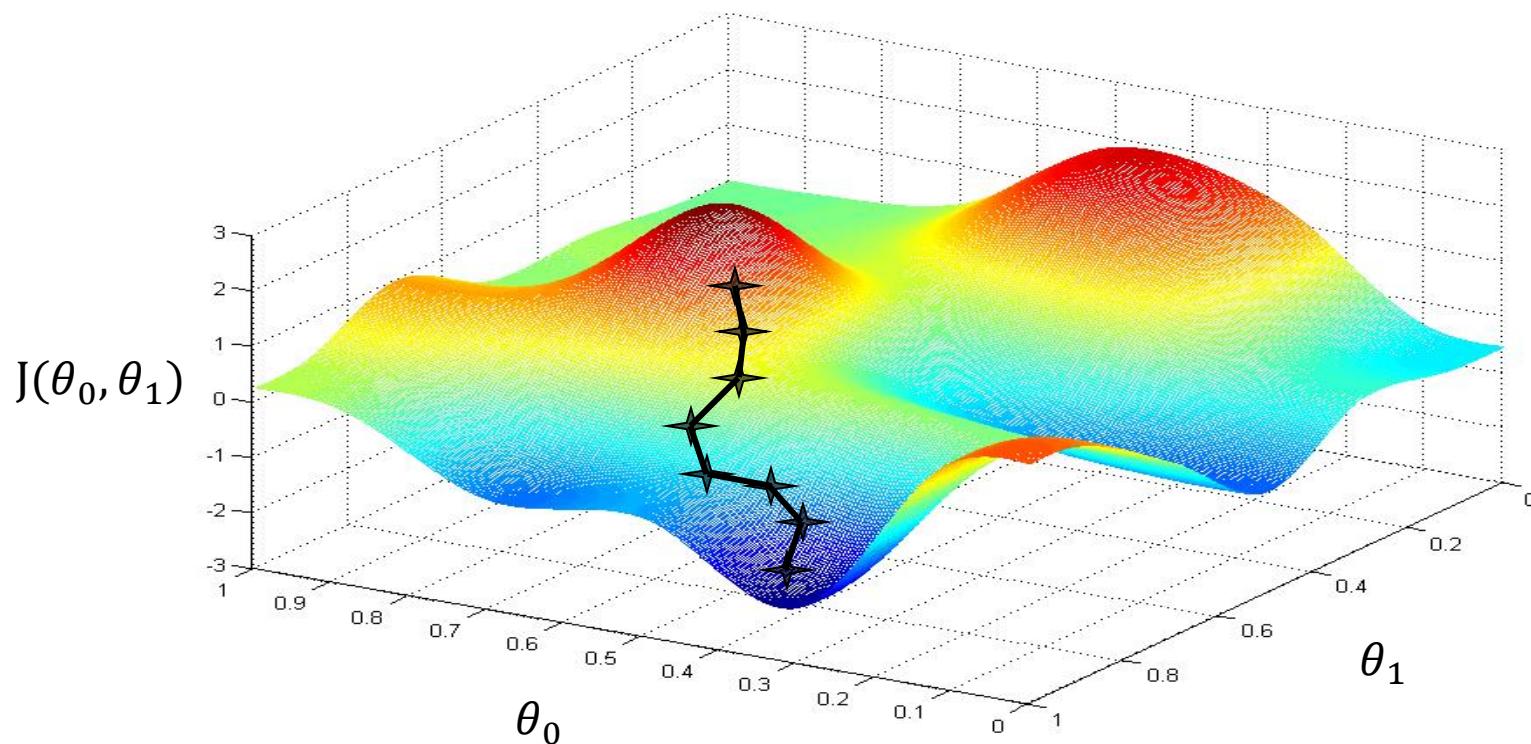
- 训练样本数量不少于特征数量
- 特征之间不可以存在线性关系

面对数据量大、特征维度高的实际数据挖掘问题，人们通常使用**迭代式优化**的方法进行参数求解，如梯度下降法。

# 迭代优化算法 Iterative optimization algorithm

- 优化目标:  $\min_{\theta} J(\theta)$
- 核心思想: 从一个初始的  $\theta$  值出发, 通过迭代的方式不断更新  $\theta$ ,  
**确保每次更新的  $J(\theta)$  取值都有所下降**

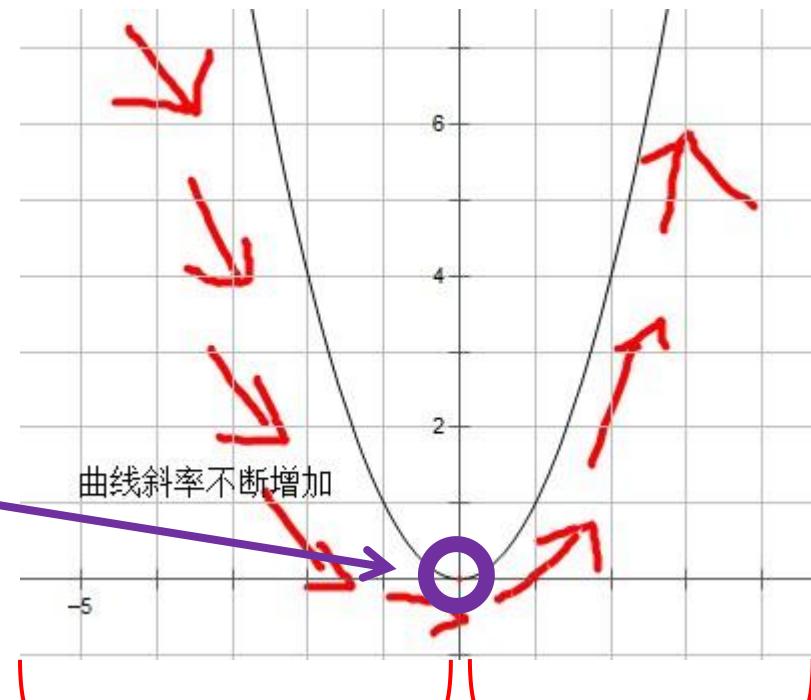
对于非凸的函数同样适用  
只是不能确保收敛到最优解



# 如何确保每次迭代目标函数都下降?

$$\frac{\partial J(\theta)}{\partial \theta} = 0$$

临界点 或 驻点



$$\frac{\partial J(\theta)}{\partial \theta} < 0 \quad \theta++$$

$$\frac{\partial J(\theta)}{\partial \theta} > 0 \quad \theta--$$

# 梯度下降与坐标下降

## • Gradient Descent and Coordinate descent

repeat until convergence {

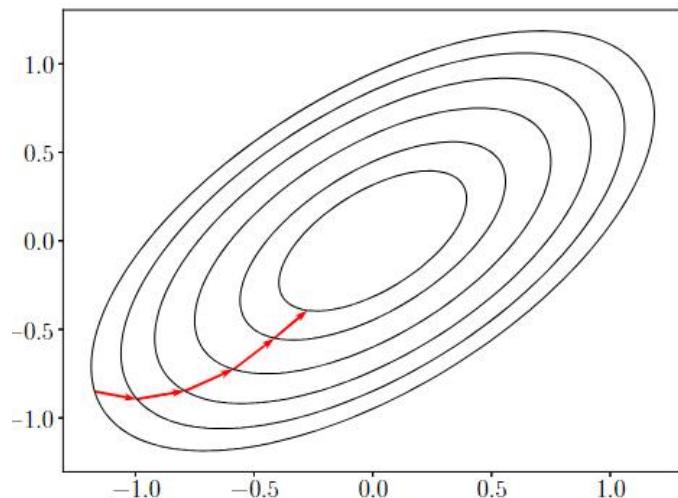
$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$

}

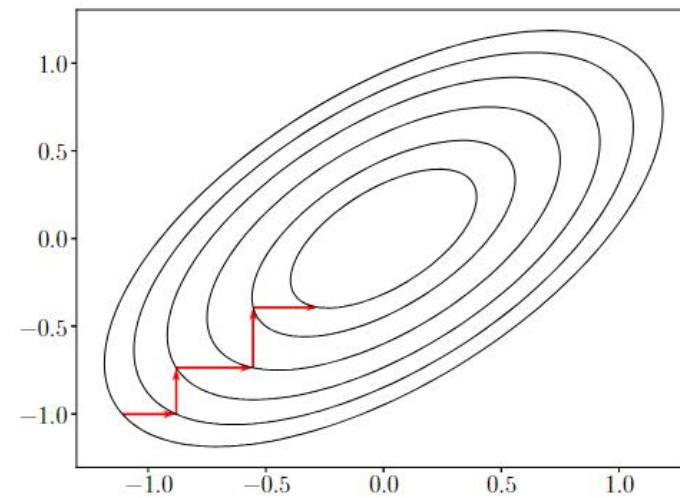
learning rate

Derivative

```
temp0 :=  $\theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$ 
temp1 :=  $\theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$ 
 $\theta_0 := \text{temp0}$ 
 $\theta_1 := \text{temp1}$ 
```



(a) 梯度下降 (同时更新  $\theta_0, \theta_1$ )

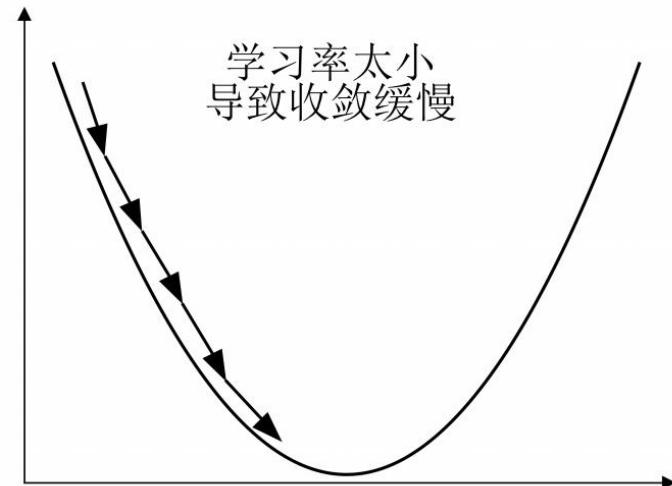


(b) 坐标下降 (轮流更新  $\theta_0, \theta_1$ )

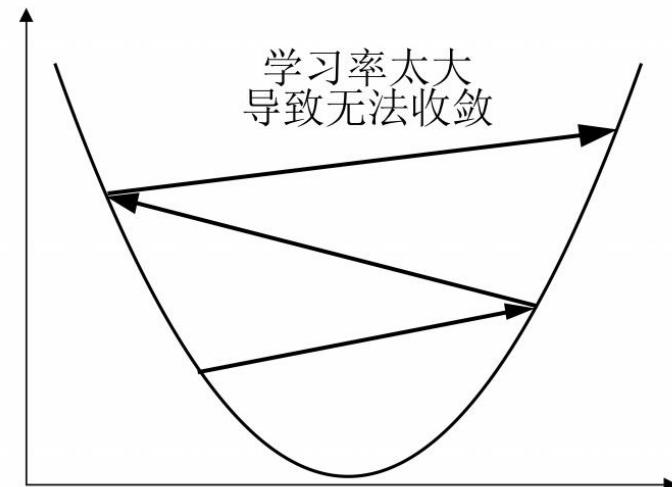
# 学习率的选择

$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

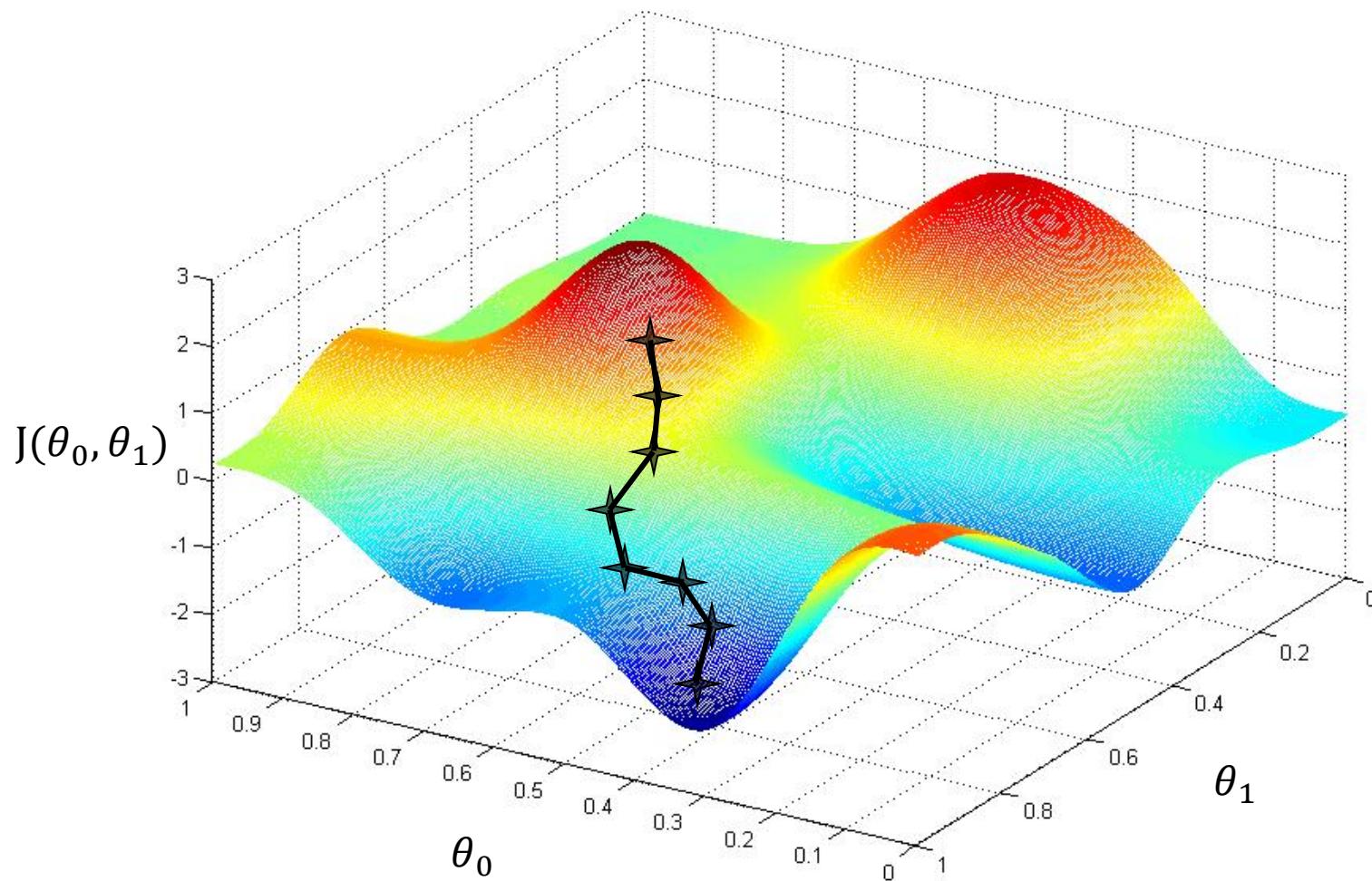
- 如果学习率**过低**, 会导致迭代效率, 收敛缓慢。



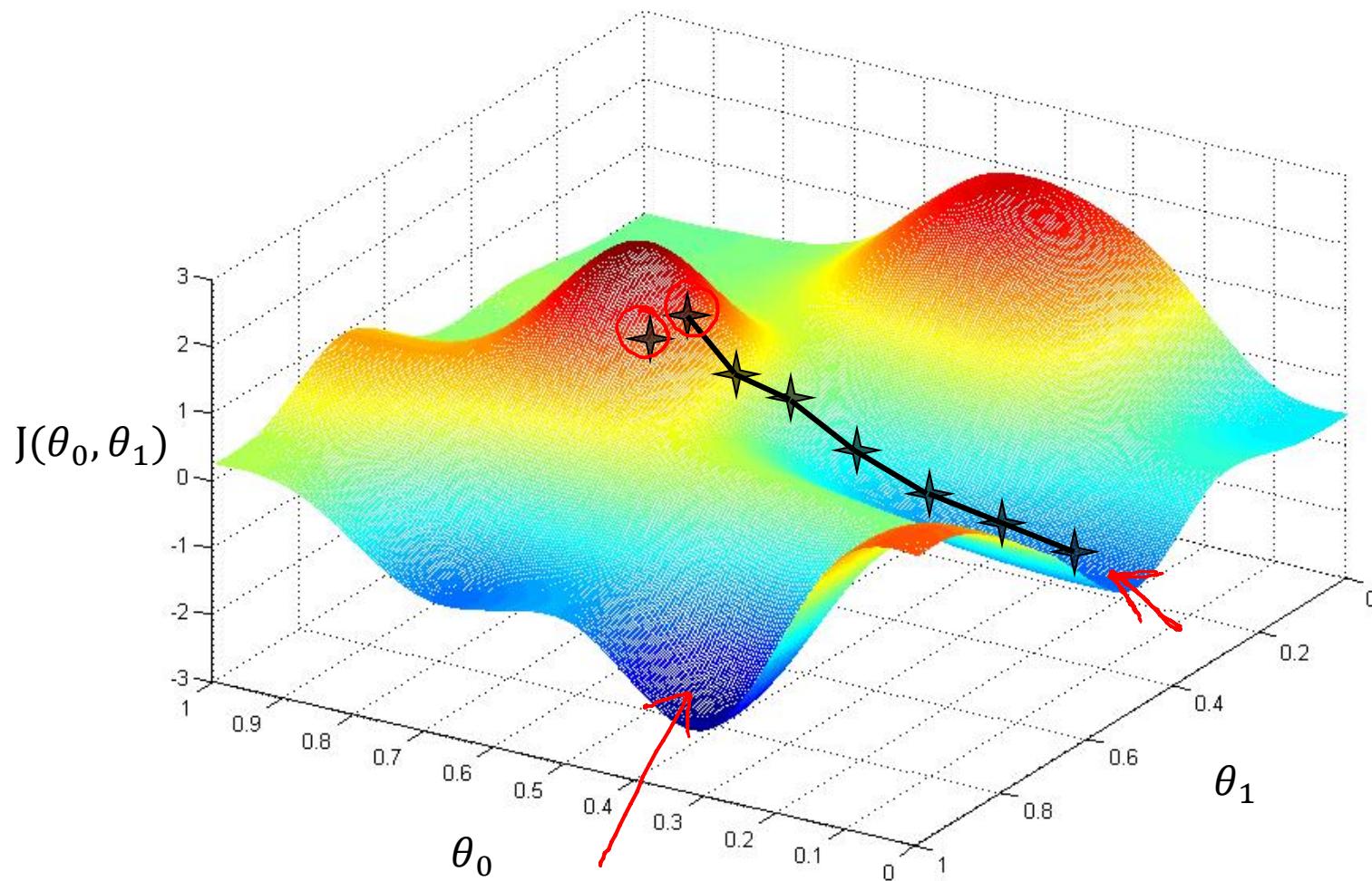
- 如果学习率**过高**, 会导致迭代梯度overshoot目标, 模型无法收敛甚至发散。



# 局部最优问题



# 局部最优问题



# 训练数据的组织方式

## 批量梯度下降 Batch gradient descent

- 每次迭代使用**全部**数据样本

Repeat until convergence {

$$\theta_j := \theta_j + \alpha \sum_{i=1}^N (y^{(i)} - f_\theta(x^{(i)})) x_j^{(i)}$$

}

## 随机梯度下降 Stochastic gradient descent

- 每次迭代使用**一个**数据样本

Repeat until convergence {

for  $i = 1 \dots N$  {

$$\theta_j := \theta_j + \alpha (y^{(i)} - f_\theta(x^{(i)})) x_j^{(i)}$$

}

}

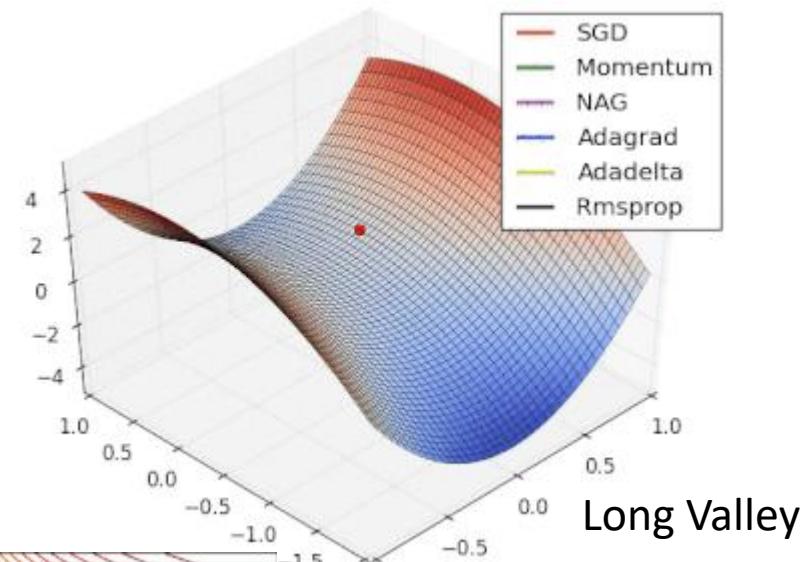
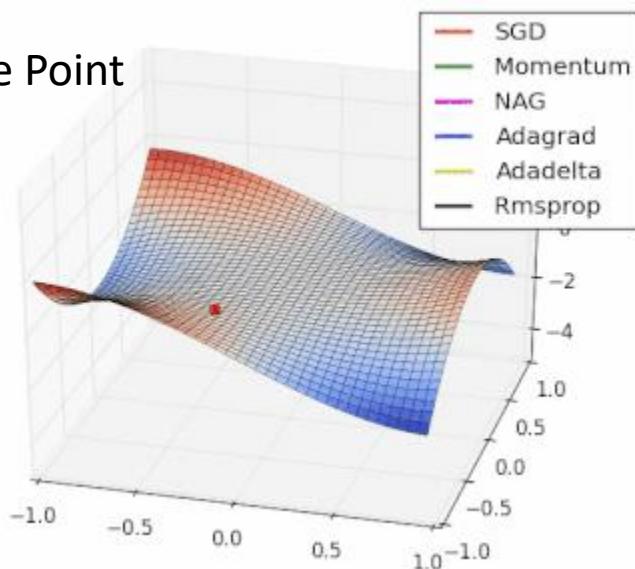
- 更容易收敛到全局最优解
- 迭代速度慢，尤其是当样本数量较大时

- 收敛速度快
- 可能无法收敛到全局最优解
- 适合在线学习

- 在工程实践中，通常使用SGD，同时对学习率进行递减
- 一些其它的训练数据组方式包括：Mini-batch Gradient Descent、带Mini-batch的SGD等

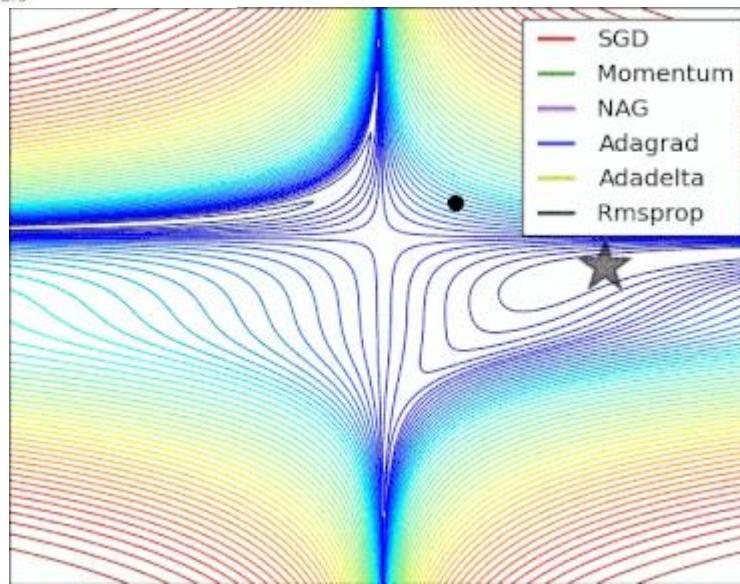
# 不同的梯度下降方法比较

Saddle Point



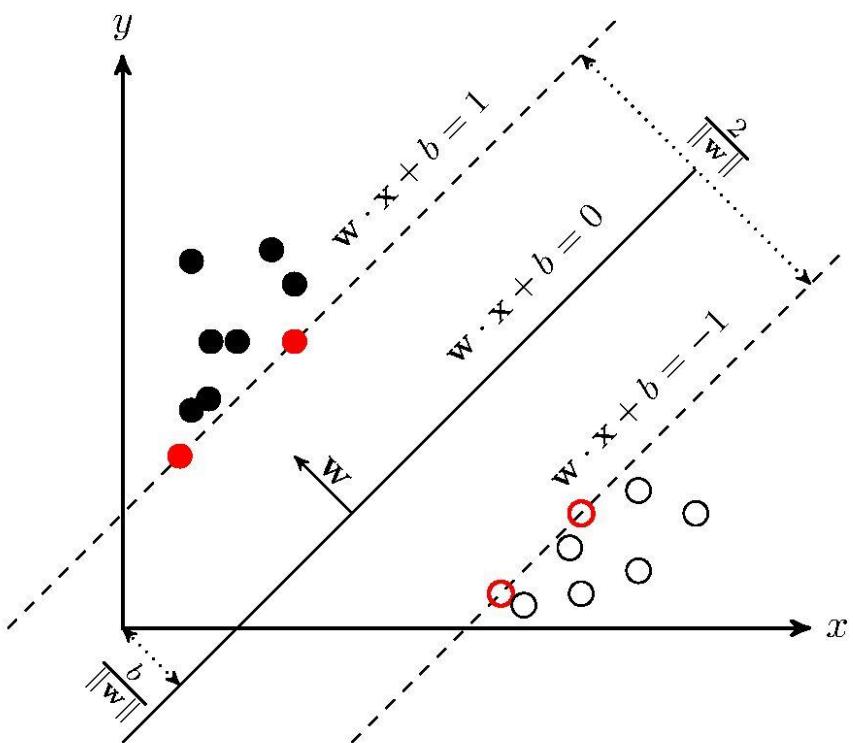
Long Valley

Beale's Function



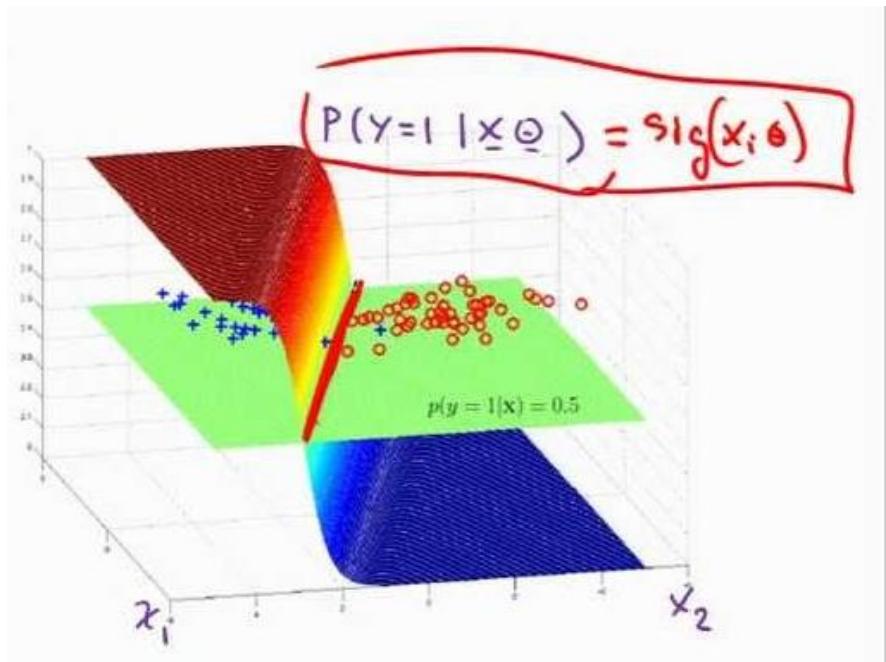
# 逻辑回归

# 逻辑回归进行分类



SVM的分类方式

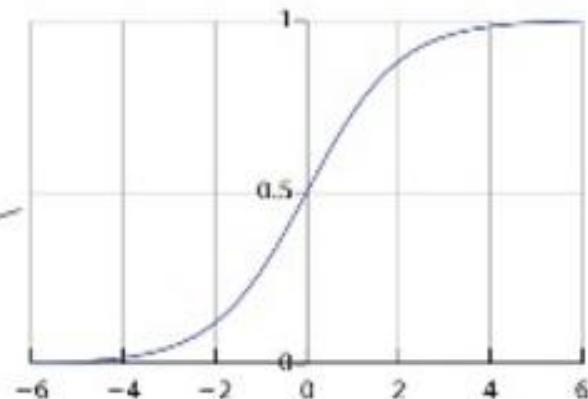
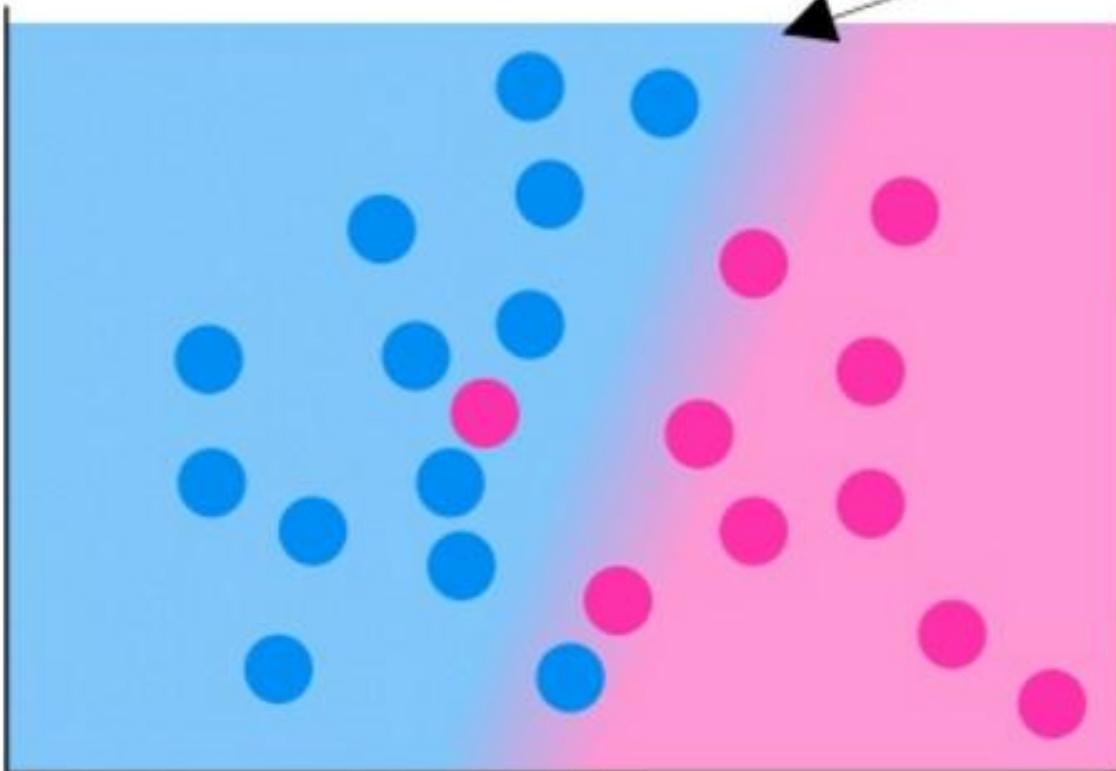
$$\{x_1, x_2, \text{sigmoid}(w_1x_1+w_2x_2)\}$$



逻辑回归的分类方式

# logistic regression

*"divide it with a log function"*



# 逻辑回归的回归函数

- **逻辑回归**: 应对二分类因变量的回归方法

$D = \{(x_1, y_1), (x_2, y_2), \dots, (x_M, y_M)\}$ , 其中  $y_m \in \{0, 1\}$

- 最理想的回归函数:

$$\hat{y}_m = g(\mathbf{w}^\top \mathbf{x}_m + b)$$

**单位跃阶函数**      
$$g(z) = \begin{cases} 0, & z < 0 \\ \frac{1}{2}, & z = 0 \\ 1, & z > 0 \end{cases}$$

# 逻辑回归的回归函数

$$\hat{y}_m = \frac{1}{1 + e^{-(w^\top x_m + b)}} \in (0, 1)$$

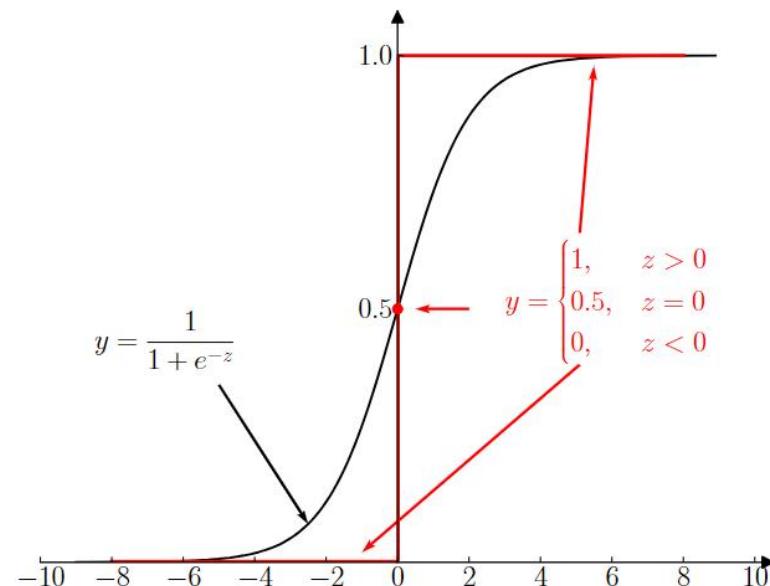
- 单位阶跃函数缺点

- 不连续

- 替代函数——标准逻辑斯谛函数 (Standard Logistic Function)

- 单调可微、任意阶可导
- Sigmoid函数

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



单位阶跃函数与sigmoid函数的比较

# 逻辑回归的损失函数

## • 交叉熵损失 (对数损失)

$$\mathcal{L}(\boldsymbol{\theta}) = -\frac{1}{M} \sum_{m=1}^M (y_m \log \hat{y}_m(\boldsymbol{\theta}, \mathbf{x}_m) + (1 - y_m) \log (1 - \hat{y}_m(\boldsymbol{\theta}, \mathbf{x}_m)))$$

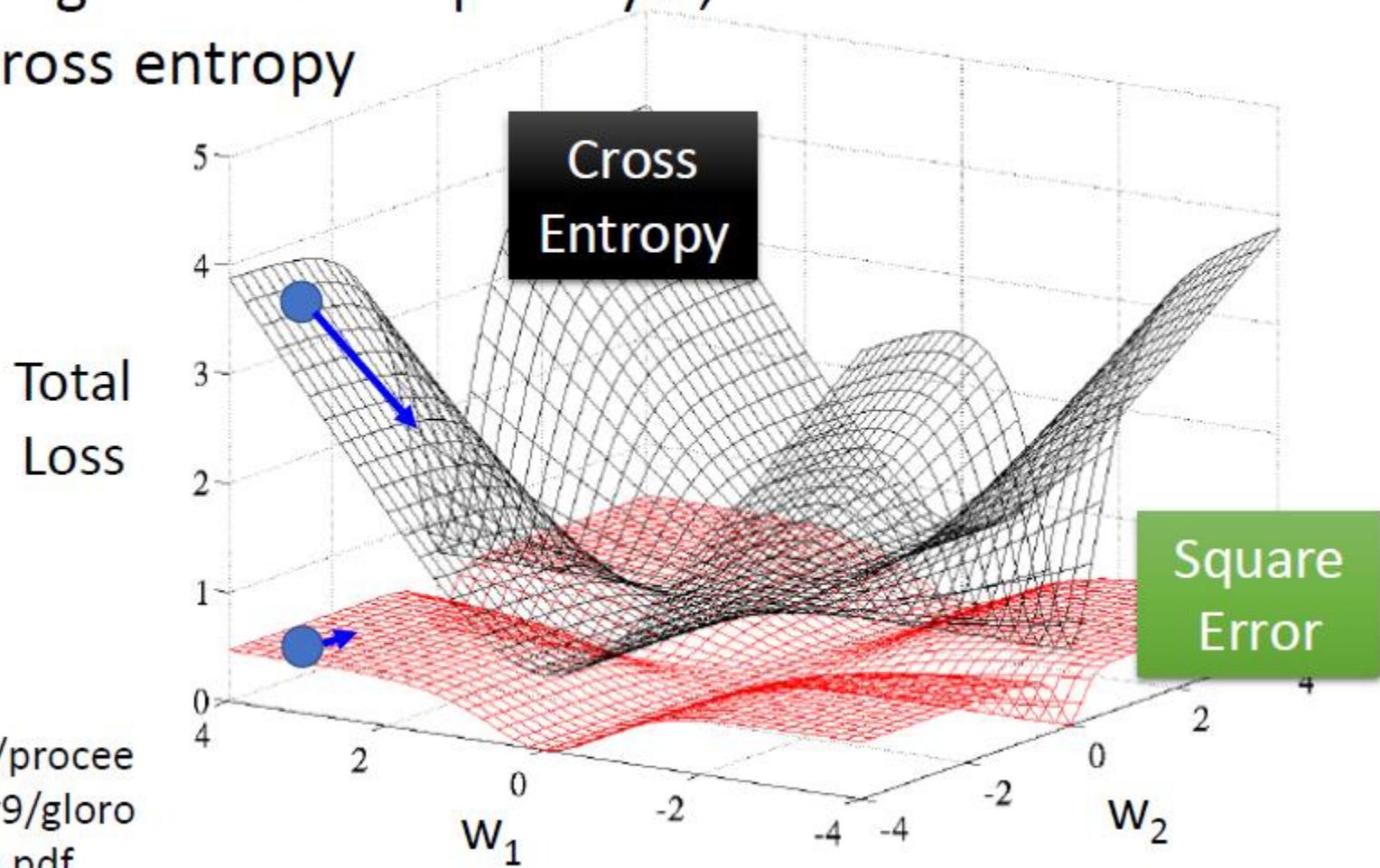
当  $y_m = \hat{y}_m = 0$  或  $y_m = \hat{y}_m = 1$  时，对应的损失项取得最小值 0

当  $y_m = 1$ 、 $\hat{y}_m = 0$  或  $y_m = 0$ 、 $\hat{y}_m = 1$  时，对应的损失项趋近于  $\infty$

$$\mathcal{L}(\boldsymbol{\theta}) = -\frac{1}{M} \sum_{m=1}^M \left( y_m \log \frac{1}{1 + e^{-\boldsymbol{\theta}^\top \tilde{\mathbf{x}}_m}} + (1 - y_m) \log \frac{e^{-\boldsymbol{\theta}^\top \tilde{\mathbf{x}}_m}}{1 + e^{-\boldsymbol{\theta}^\top \tilde{\mathbf{x}}_m}} \right)$$

# 交叉熵损失函数

When using softmax output layer,  
choose cross entropy



# 逻辑回归的参数求解方法

- 梯度下降解

$$\begin{aligned}\frac{\partial J(\theta)}{\partial \theta} &= -\frac{1}{M} \sum_{m=1}^M \frac{\partial L_m(\theta)}{\partial \theta} \\ &= \frac{1}{M} \sum_{m=1}^M \left( \sigma(\boldsymbol{\theta}^\top \tilde{x}_m) - y_m \right) \tilde{x}_m\end{aligned}$$

- 解析解

$$\begin{cases} w = \Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) \\ b = \frac{1}{2} \left( \boldsymbol{\mu}_0^\top \Sigma^{-1} \boldsymbol{\mu}_0 - \boldsymbol{\mu}_1^\top \Sigma^{-1} \boldsymbol{\mu}_1 \right) + \log \frac{\Pr(y=1)}{\Pr(y=0)} \end{cases}$$

# 多项逻辑回归

# 多项逻辑回归

- **SoftMax 回归**

- 多分类数据集

其中  $x_m \in \mathbb{R}^N$ ,  $y'_m \in \{1, 2, \dots, K\}$ ,  $m = 1, 2, \dots, M$

- 对于  $y'_m = k$ , **独热编码** 构造一个向量  $y_m \in R^K$ ,  
其中第  $k$  个分量  $y_{m,k} = 1$ , 其他分量为 0
- 新数据集

$$D = \{(x_1, y_1), (x_2, y_2), \dots, (x_M, y_M)\}$$

# 多项逻辑回归的回归函数

- 对每一个样本  $x_m$ , 使用  $K$  个线性组合对其进行变换, 生成  $K$  个得分

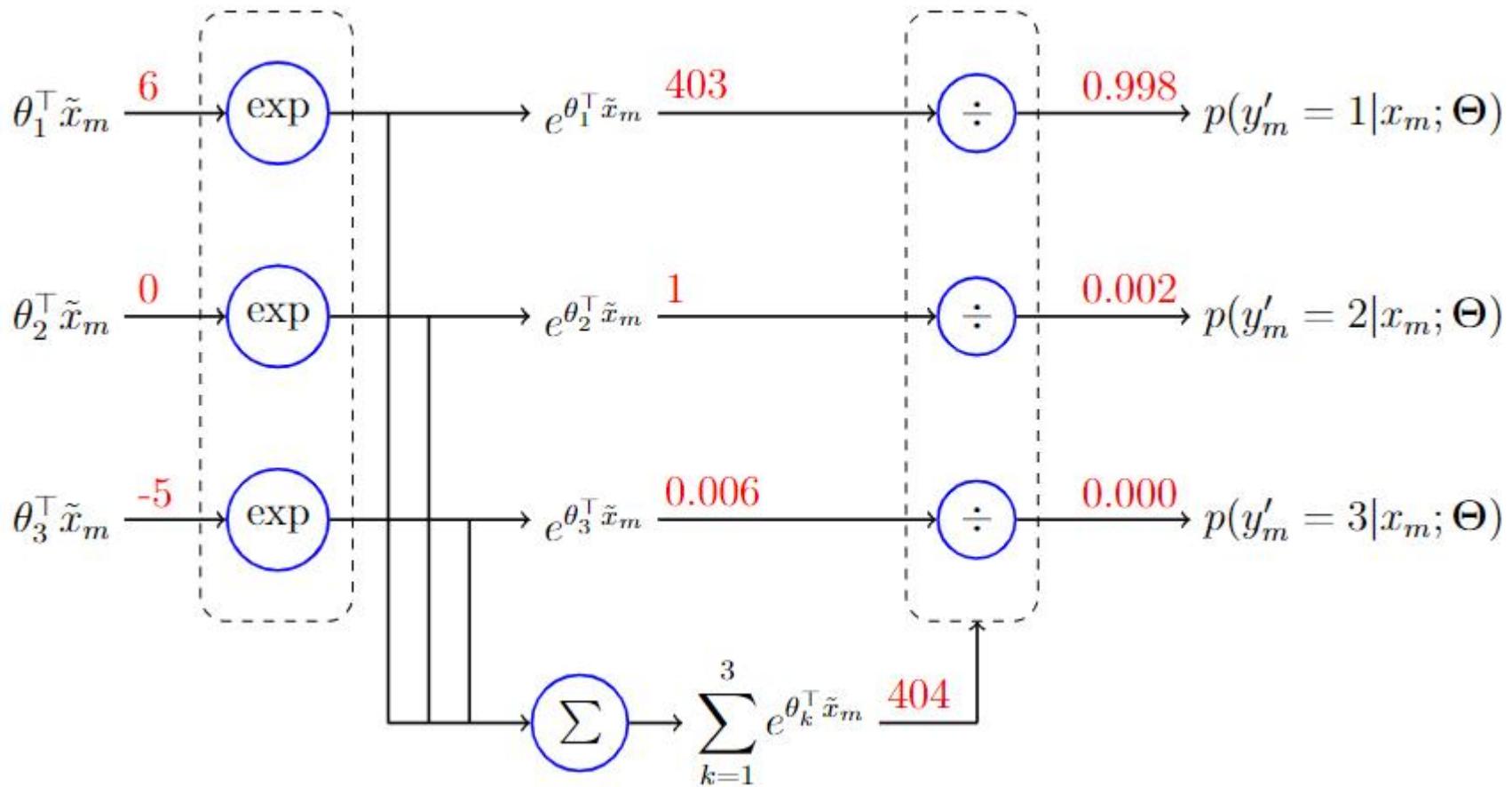
$$\begin{pmatrix} z_{m,1} \\ \vdots \\ z_{m,k} \\ \vdots \\ z_{m,K} \end{pmatrix} = \begin{pmatrix} \theta_1^\top \tilde{x}_m \\ \vdots \\ \theta_k^\top \tilde{x}_m \\ \vdots \\ \theta_K^\top \tilde{x}_m \end{pmatrix} \quad \hat{y}'_m = \arg \max_k \{z_{m,1}, \dots, z_{m,k}, \dots, z_{m,K}\}$$

归一化

$$\hat{y}_m = \begin{pmatrix} \hat{y}_{m,1} \\ \vdots \\ \hat{y}_{m,k} \\ \vdots \\ \hat{y}_{m,K} \end{pmatrix} = \begin{pmatrix} \frac{e^{\theta_1^\top \tilde{x}_m}}{\sum_{j=1}^K e^{\theta_j^\top \tilde{x}_m}} \\ \vdots \\ \frac{e^{\theta_k^\top \tilde{x}_m}}{\sum_{j=1}^K e^{\theta_j^\top \tilde{x}_m}} \\ \vdots \\ \frac{e^{\theta_K^\top \tilde{x}_m}}{\sum_{j=1}^K e^{\theta_j^\top \tilde{x}_m}} \end{pmatrix}$$

$\hat{y}_{m,k}$  可以被视为  $y'_m = k$  的概率

# 多项逻辑回归的解释示例



# 多项逻辑回归的损失函数

- 在多分类问题中，我们依然采用交叉熵和来度量  $y_m$  与  $\hat{y}_m$  之间的误差

$$\mathcal{L}(\Theta) = -\frac{1}{M} \sum_{m=1}^M \left( \sum_{k=1}^K y_{m,k} \log \hat{y}_{m,k} (\theta_k, x_m) \right)$$

其中  $\hat{y}_{m,k} (\theta_k, x_m)$  表示  $\hat{y}_{m,k}$  是  $\theta_k$  和  $x_m$  的函数

# 多项逻辑回归的参数求解方法

- 梯度下降解

$$\begin{aligned}\frac{\partial \mathcal{L}(\Theta)}{\partial \theta_i} &= -\frac{1}{M} \sum_{m=1}^M \frac{\partial}{\partial \theta_i} \left[ \sum_{k=1}^K \mathbb{I}(y'_m = i) \log \frac{e^{\theta_k^\top \tilde{x}_m}}{\sum_{j=1}^K e^{\theta_j^\top \tilde{x}_m}} \right] \\ &= -\frac{1}{M} \sum_{m=1}^M \frac{\partial}{\partial \theta_i} \left[ \mathbb{I}(y'_m = i) \left( \theta_i^\top \tilde{x}_m - \log \sum_{j=1}^K e^{\theta_j^\top \tilde{x}_m} \right) \right] \\ &= -\frac{1}{M} \sum_{m=1}^M \left[ \mathbb{I}(y'_m = i) \left( \tilde{x}_m - \frac{e^{\theta_i^\top \tilde{x}_m}}{\sum_{j=1}^K e^{\theta_j^\top \tilde{x}_m}} \cdot \tilde{x}_m \right) \right] \\ &= \frac{1}{M} \sum_{m=1}^M \left( \frac{e^{\theta_i^\top \tilde{x}_m}}{\sum_{j=1}^K e^{\theta_j^\top \tilde{x}_m}} - \mathbb{I}(y'_m = i) \right) \tilde{x}_m\end{aligned}$$

- 解析解

$$\begin{cases} w_k = \Sigma^{-1} \mu_k \\ b_k = \log \Pr(y = k) - \frac{1}{2} \mu_k^\top \Sigma^{-1} \mu_k \end{cases}$$

# 非线性回归

# 多项式回归

- $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_M, y_M)\}$  (x为标量)
- 回归函数：**自变量  $x_m$  的多项式线性组合**

$$\hat{y}_m(x_m, \theta) = \theta_0 + \theta_1 \cdot x_m + \theta_2 \cdot x_m^2 + \dots + \theta_K \cdot x_m^K = \sum_{k=0}^K \theta_k \cdot x_m^k$$

其中， $\theta = (\theta_1, \dots, \theta_K)$  是回归函数的参数

- 损失函数：残差平方和

$$\mathcal{L}(\theta) = \sum_{m=1}^M (y_m - \hat{y}_m)^2$$

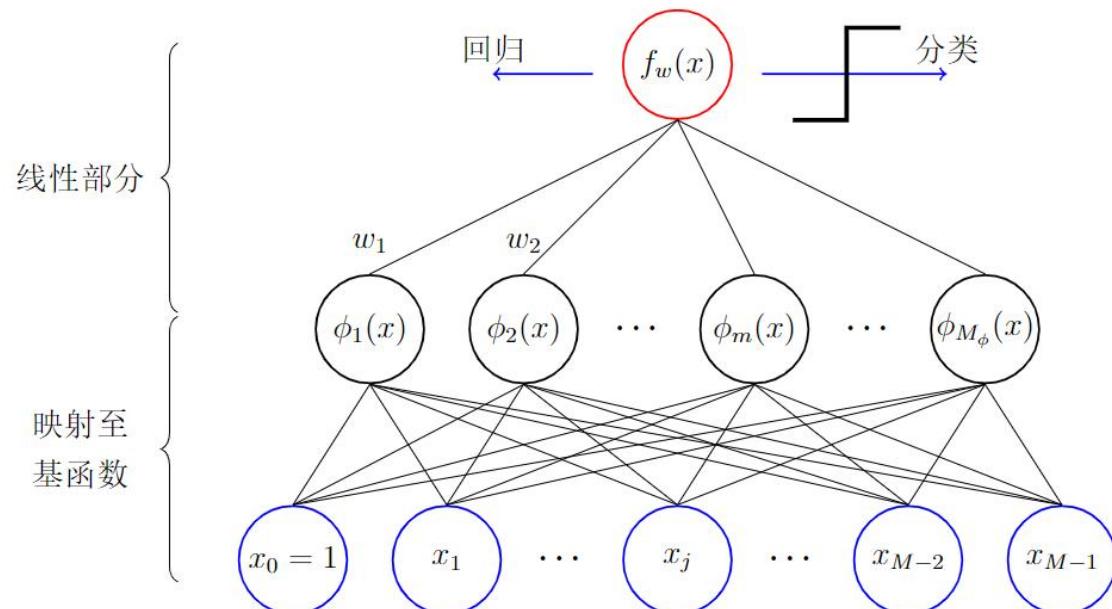
# 线性基函数模型

- $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_M, y_M)\}$  ( $\mathbf{x}$ 为向量)
- 回归函数：

$$\hat{y}_m = \theta_0 + \sum_{k=1}^N \theta_k \phi_k(\mathbf{x}_m) = \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x}_m)$$

$$\boldsymbol{\phi}(\mathbf{x}_m) = (\phi_0, \phi_1(\mathbf{x}_m), \dots, \phi_N(\mathbf{x}_m))^\top, \text{ 其中 } \phi_0 = 1$$

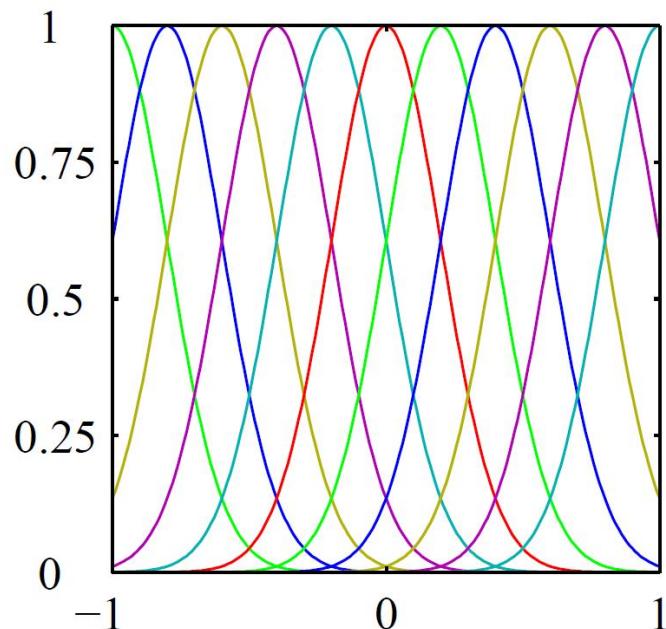
- 基函数： $\phi_k(\cdot)$



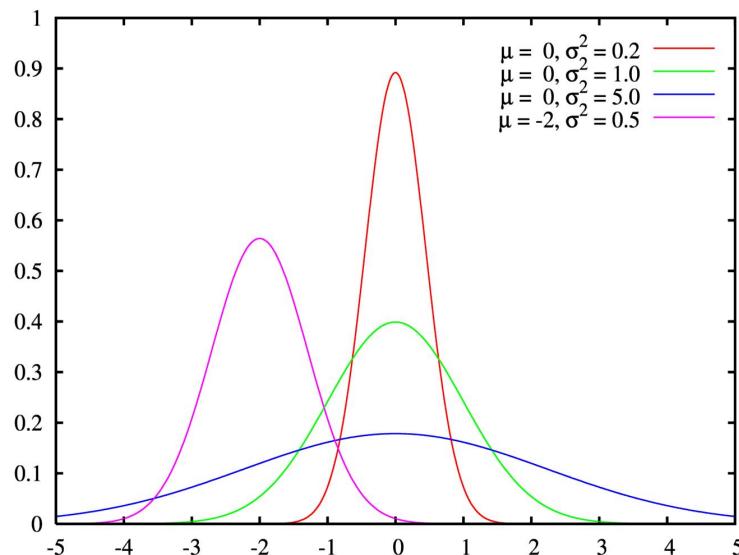
# 不同的基函数

- 径向基函数

$$\phi_n(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \mu_n\|^2}{2s^2}\right)$$



其中  $\mu_n$  是第  $n$  个基函数在高维空间上的均值， $s$  是方差， $\|\cdot\|_{L_2}$  表示向量的 L2 范数



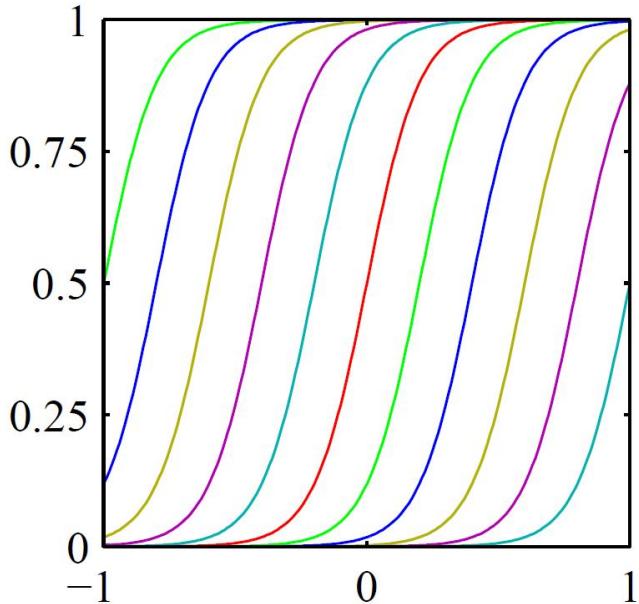
# 不同的基函数

- Sigmoid 基函数

$$\phi_n(x) = \sigma\left(\frac{\mathbf{1}^\top (x - \mu_n)}{s}\right)$$

函数 $\sigma(\cdot)$ 是 Sigmoid 函数

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



$\mu_n$  是第 n 个基函数的中心位置

# 线性基函数模型的损失函数与闭式解

$$\hat{y}_m = \theta_0 + \sum_{k=1}^N \theta_k \phi_k(\mathbf{x}_m) = \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x}_m) \quad \text{线性基函数模型}$$

- 使用最小均方误差构建损失函数

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{m=1}^M \left( \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x}_m) - \hat{y}_m \right)^2$$

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = 0 \longrightarrow \boxed{\boldsymbol{\theta}^* = (\boldsymbol{\Phi}^\top \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top \mathbf{y}} \quad \text{闭式解}$$

$$\boldsymbol{\Phi} = \begin{pmatrix} \boldsymbol{\phi}(\mathbf{x}_1) \\ \vdots \\ \boldsymbol{\phi}(\mathbf{x}_m) \\ \vdots \\ \boldsymbol{\phi}(\mathbf{x}_M) \end{pmatrix} = \begin{pmatrix} \phi_0, \phi_1(\mathbf{x}_1), \dots, \phi_K(\mathbf{x}_1) \\ \vdots \\ \phi_0, \phi_1(\mathbf{x}_m), \dots, \phi_K(\mathbf{x}_m) \\ \vdots \\ \phi_0, \phi_1(\mathbf{x}_M), \dots, \phi_K(\mathbf{x}_M) \end{pmatrix}$$

# 核函数

- 给定基函数  $\phi(\mathbf{x})$ , 以及任意两个样本  $x_i$  和  $x_j$

核函数

$$\kappa(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle \quad \text{内积}$$

- 对一个新的输入样本  $x_i$   
将  $\theta^* = (\Phi^\top \Phi)^{-1} \Phi^\top y$  代入  $\hat{y}_m = \theta_0 + \sum_{k=1}^N \theta_k \phi_k(x_m) = \boldsymbol{\theta}^\top \phi(x_m)$

可以得到其对应的预测值  $\hat{v}$ ,

$$\hat{y}_i = \boldsymbol{\theta}^{*\top} \phi(x_i) = \mathbf{y} (\Phi \Phi^\top)^{-1} \Phi \phi(x_i)$$

# 核函数

$$\hat{y}_i = \boldsymbol{\theta}^{*\top} \phi(\mathbf{x}_i) = \mathbf{y} (\Phi \Phi^\top)^{-1} \Phi \phi(\mathbf{x}_i)$$

• 核技巧：使用核函数代替内积操作

$$\boxed{\hat{y}_i = \mathbf{y} K^{-1} \kappa(\mathbf{X}, \mathbf{x}_i)} \quad \kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$$

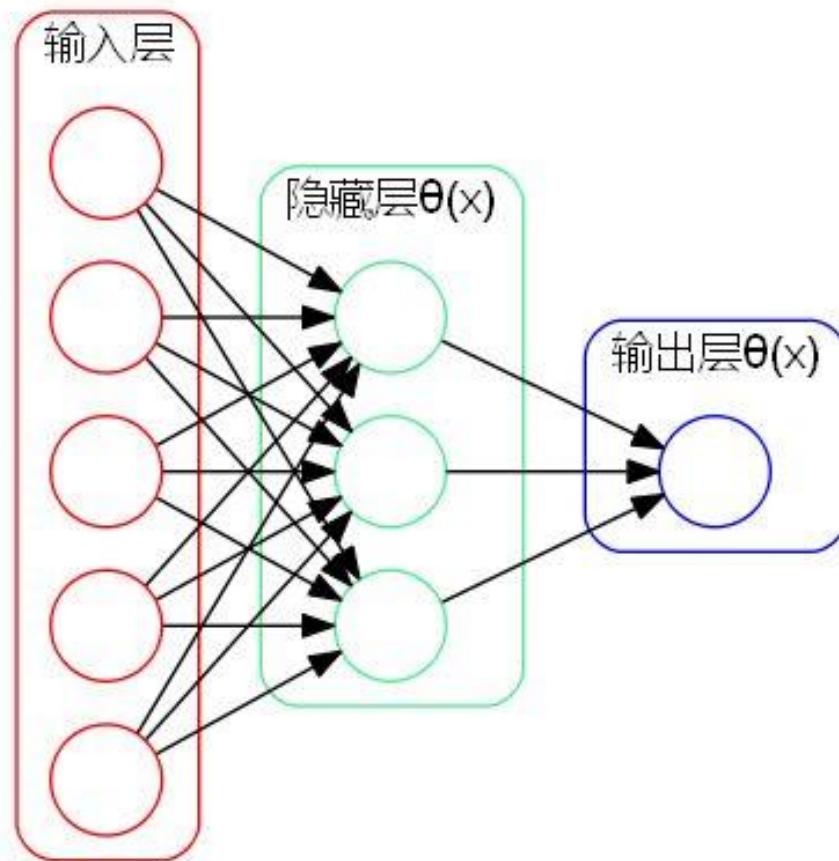
$$K = \begin{pmatrix} \kappa(\mathbf{x}_1, \mathbf{x}_1), \dots, \kappa(\mathbf{x}_1, \mathbf{x}_M) \\ \vdots \\ \kappa(\mathbf{x}_m, \mathbf{x}_1), \dots, \kappa(\mathbf{x}_m, \mathbf{x}_M) \\ \vdots \\ \kappa(\mathbf{x}_M, \mathbf{x}_1), \dots, \kappa(\mathbf{x}_M, \mathbf{x}_M) \end{pmatrix}, \kappa(\mathbf{X}, \mathbf{x}_i) = \begin{pmatrix} \kappa(\mathbf{x}_1, \mathbf{x}_i) \\ \vdots \\ \kappa(\mathbf{x}_m, \mathbf{x}_i) \\ \vdots \\ \kappa(\mathbf{x}_M, \mathbf{x}_i) \end{pmatrix}$$

# 常用核函数

名称	表达式	参数
线性核	$\kappa(x_i, x_j) = \mathbf{x}_i^\top \mathbf{x}_j$	
多项式核	$\kappa(x_i, x_j) = (\mathbf{x}_i^\top \mathbf{x}_j)^d$	$d \geq 1$ 为多项式的次数
高斯核	$\kappa(x_i, x_j) = \exp\left(-\frac{\ \mathbf{x}_i - \mathbf{x}_j\ ^2}{2\sigma^2}\right)$	$\sigma > 0$ 为高斯核的带宽 (width)
拉普拉斯核	$\kappa(x_i, x_j) = \exp\left(-\frac{\ \mathbf{x}_i - \mathbf{x}_j\ }{\sigma}\right)$	$\sigma > 0$
Sigmoid 核	$\kappa(x_i, x_j) = \tanh(\beta \mathbf{x}_i^\top \mathbf{x}_j + \theta)$	$\tanh$ 为双曲正切函数, $\beta > 0, \theta < 0$

# 思考一下

- 如果基函数是逻辑回归，那么模型会变成什么样子？



# 正则项

# 正则项

- **正则项**: 对回归模型的参数求解进行显示正则化, 在回归模型的损失函数上增加一个**关于模型复杂度的度量项**

$$\mathcal{J}(\theta) = \mathcal{L}(\theta; X, y) + \lambda \Omega(\theta)$$



经验风险, 描述模型与  
训练数据的契合程度

结构风险, 描述  
模型的某些性质

$$\frac{\partial J(\theta)}{\partial \theta} = 0 \Rightarrow \frac{\partial L(\theta)}{\partial \theta} = -\lambda \frac{\partial \Omega(\theta)}{\partial \theta}$$

参数最优解在**经验风险函数**和**结构风险函数相切处**

# 岭回归与 LASSO 回归

$$\|\theta\|_p = \left( \sum_i |\theta_i|^p \right)^{\frac{1}{p}}$$

$L_p$ , p阶范数

- 岭回归 (以参数的L2范数为正则项)

$$\mathcal{J}(\theta) = \frac{1}{M} \sum_{m=1}^M (\theta^\top x_m - \hat{y}_m)^2 + \lambda \|\theta\|_2^2$$

- LASSO回归 (以参数的L1范数为正则项)

$$\mathcal{J}(\theta) = \frac{1}{M} \sum_{m=1}^M (\theta^\top x_m - \hat{y}_m)^2 + \lambda \|\theta\|_1$$

# 岭回归与 LASSO 回归的闭式解

- 岭回归 (以参数的L2范数为正则项)

$$\theta = (\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}} + \lambda \mathbf{I})^{-1} \tilde{\mathbf{X}}^\top \mathbf{y}$$

绝对值减小

- LASSO回归 (以参数的L1范数为正则项)

$$\theta = (\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}})^{-1} \left( \tilde{\mathbf{X}}^\top \mathbf{y} - \frac{1}{2} \lambda \text{sign}(\theta) \right)$$

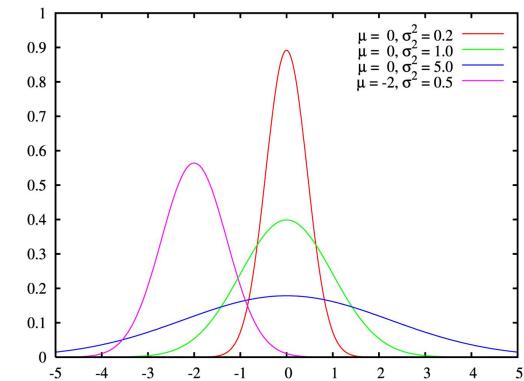
# L1与L2正则项的比较

高斯分布

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

L2 正则

$$\mathcal{J}(\theta) = \frac{1}{M} \sum_{m=1}^M (\theta^\top x_m - \hat{y}_m)^2 + \lambda \|\theta\|_2^2$$

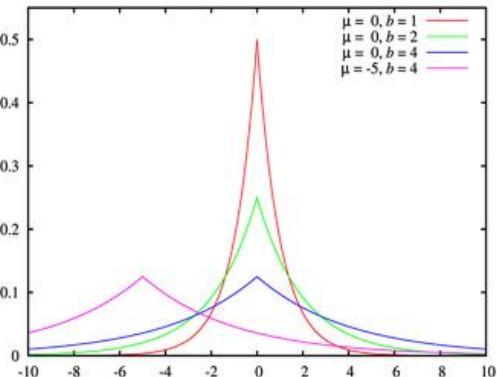


拉普拉斯分布

$$f(x) = \frac{1}{\sqrt{2}\sigma} e^{[-\frac{\sqrt{2}}{\sigma}|x|]}$$

L1 正则

$$\mathcal{J}(\theta) = \frac{1}{M} \sum_{m=1}^M (\theta^\top x_m - \hat{y}_m)^2 + \lambda \|\theta\|_1$$



# L1与L2正则项的比较

高斯分布

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

L2正则

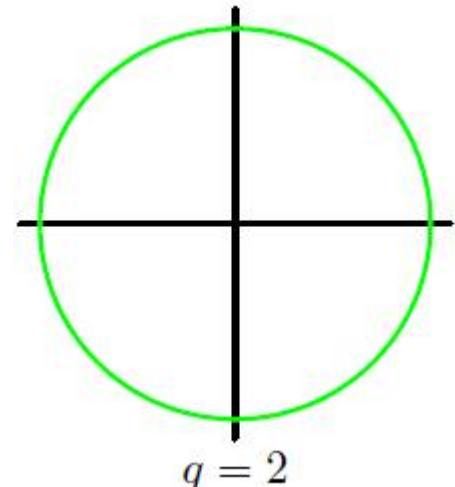
$$\mathcal{J}(\theta) = \frac{1}{M} \sum_{m=1}^M (\theta^\top x_m - \hat{y}_m)^2 + \lambda \|\theta\|_2^2$$

拉普拉斯分布

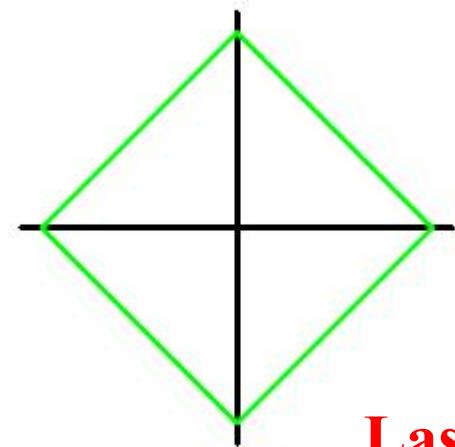
$$f(x) = \frac{1}{\sqrt{2\sigma}} e^{-\frac{\sqrt{2}}{\sigma}|x|}$$

L1正则

$$\mathcal{J}(\theta) = \frac{1}{M} \sum_{m=1}^M (\theta^\top x_m - \hat{y}_m)^2 + \lambda \|\theta\|_1$$



Ridge

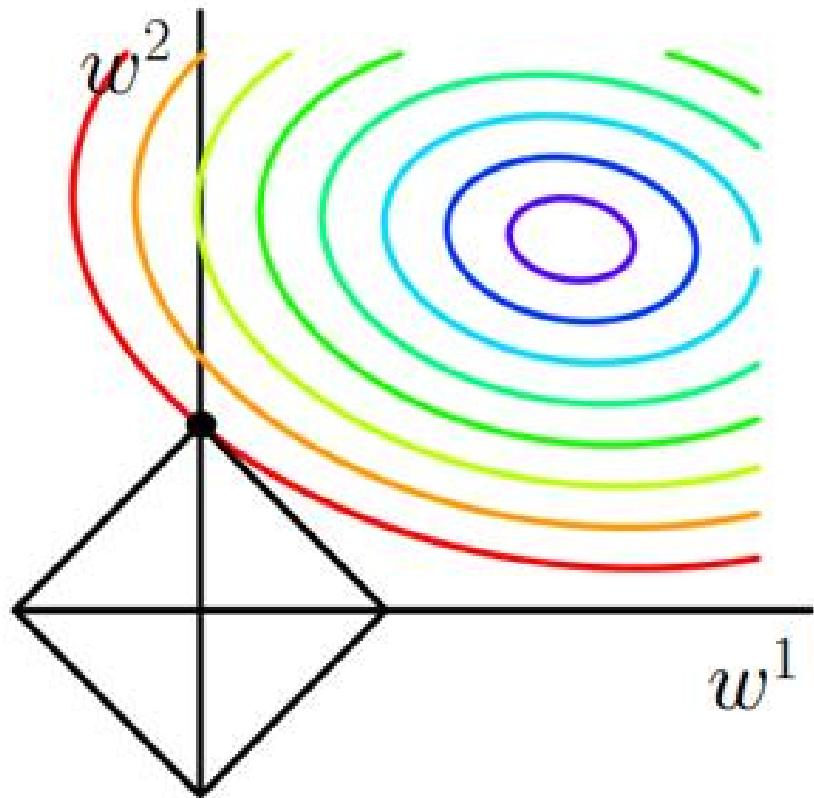


Least Absolute Shrinkage  
and Selection Operator

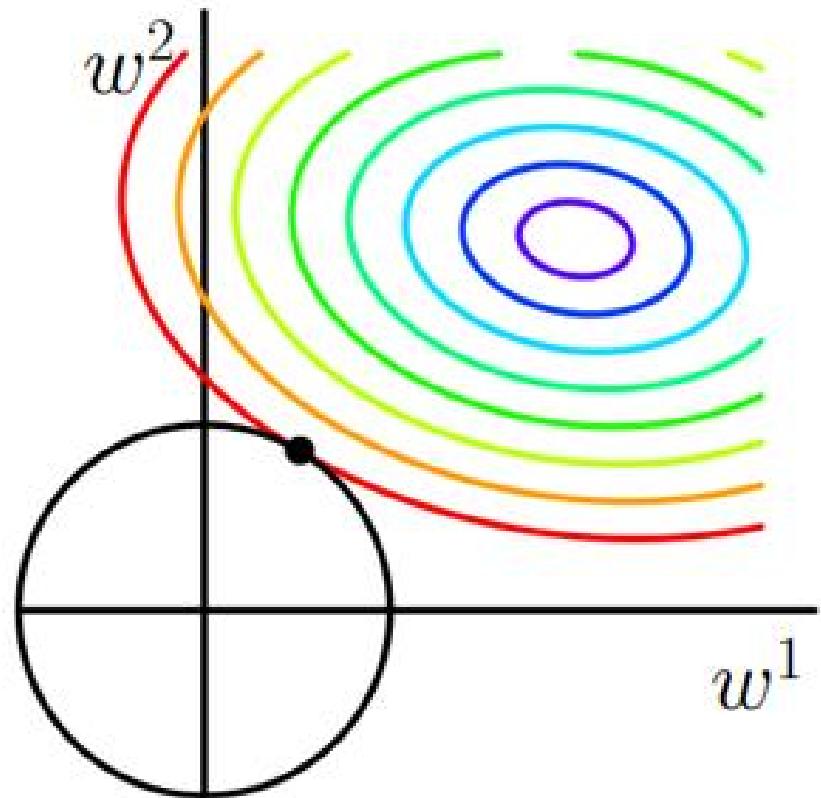
Lasso

# L1与L2正则项的比较

- L1正则的解更加稀疏

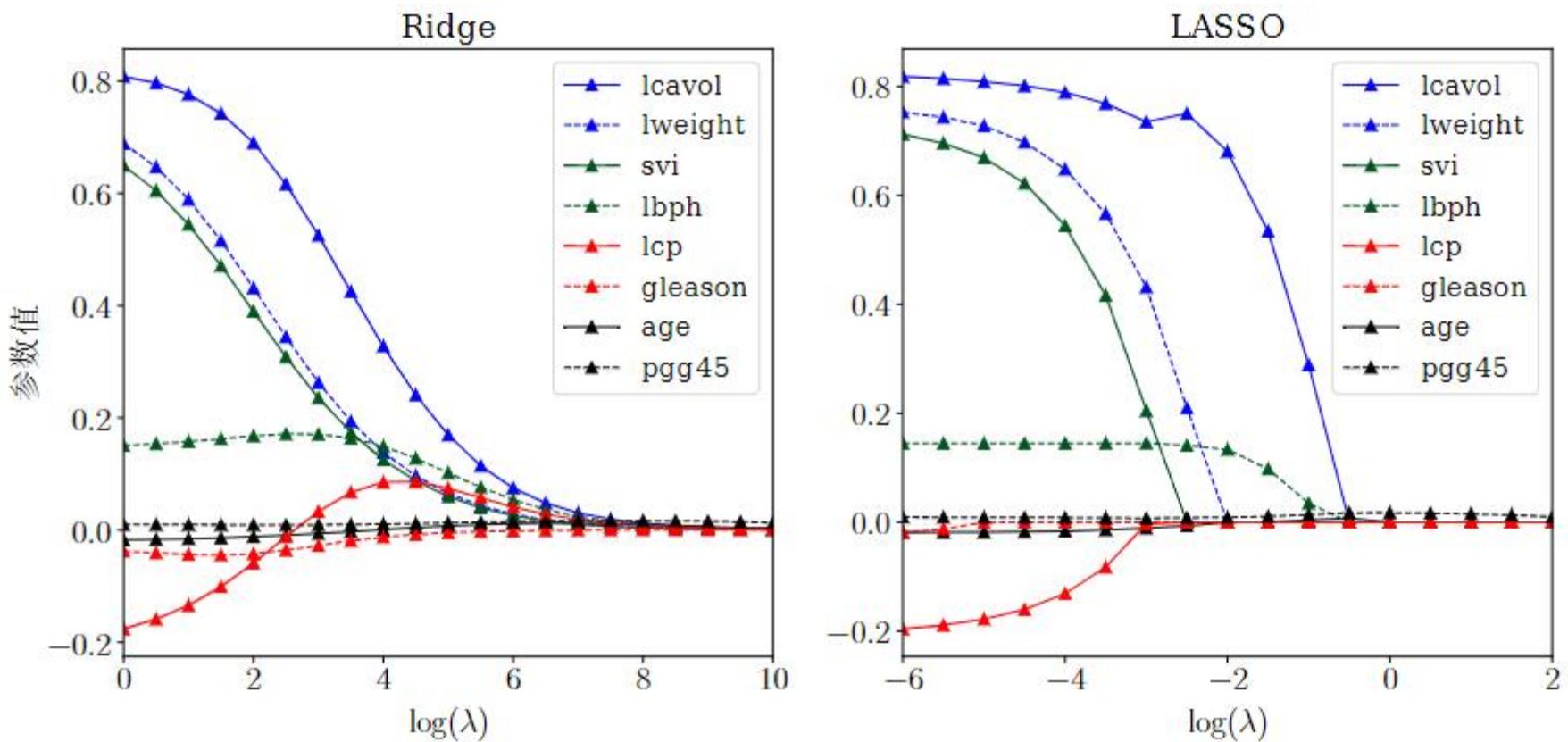


L1 regularization

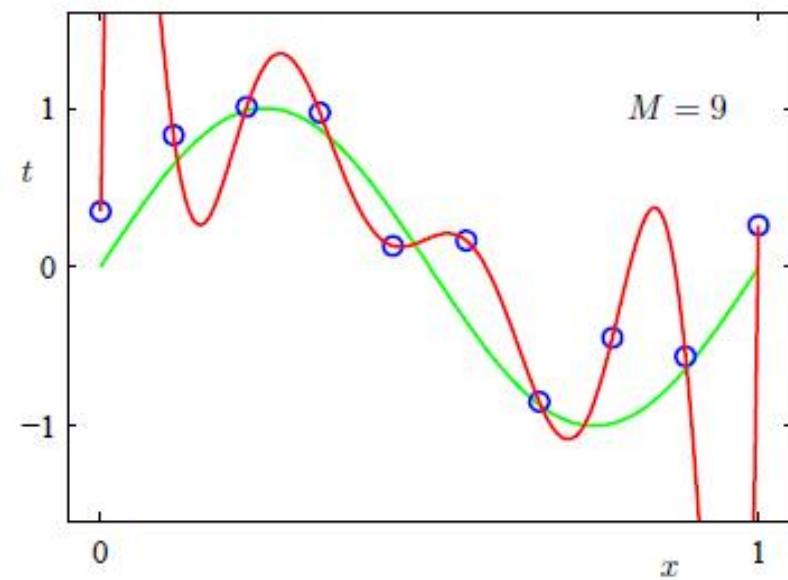
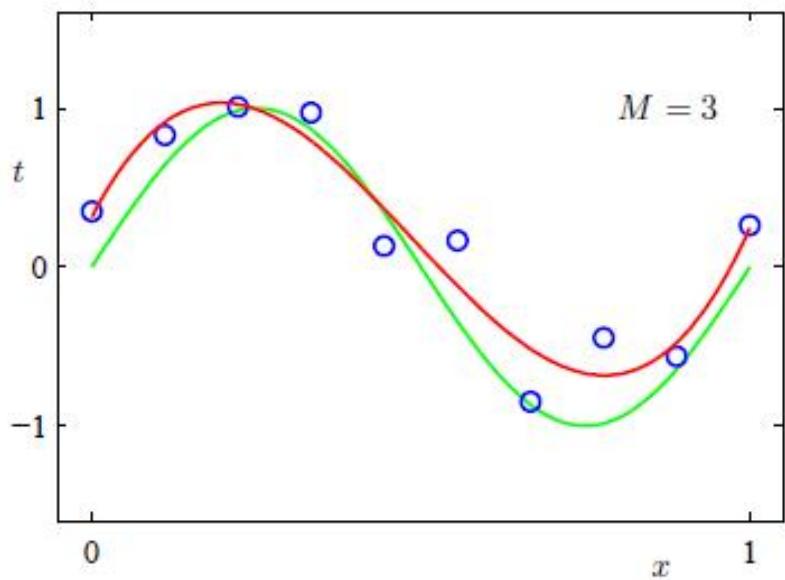
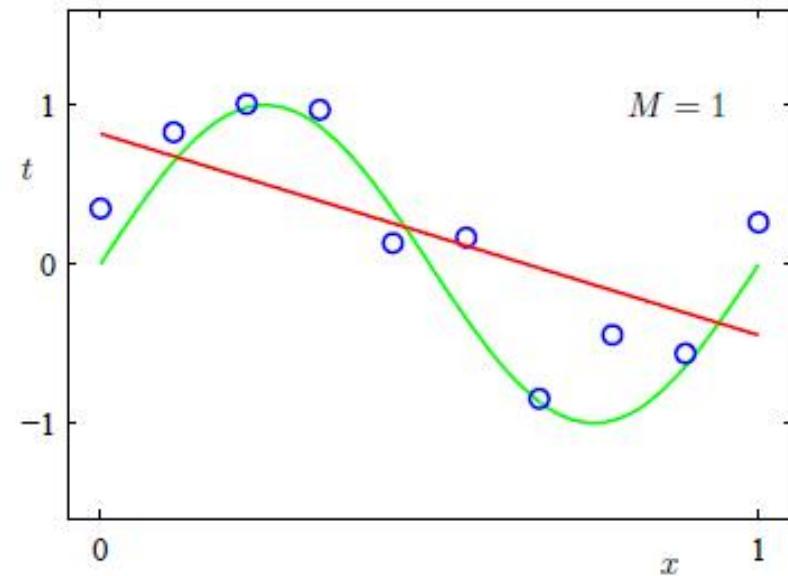
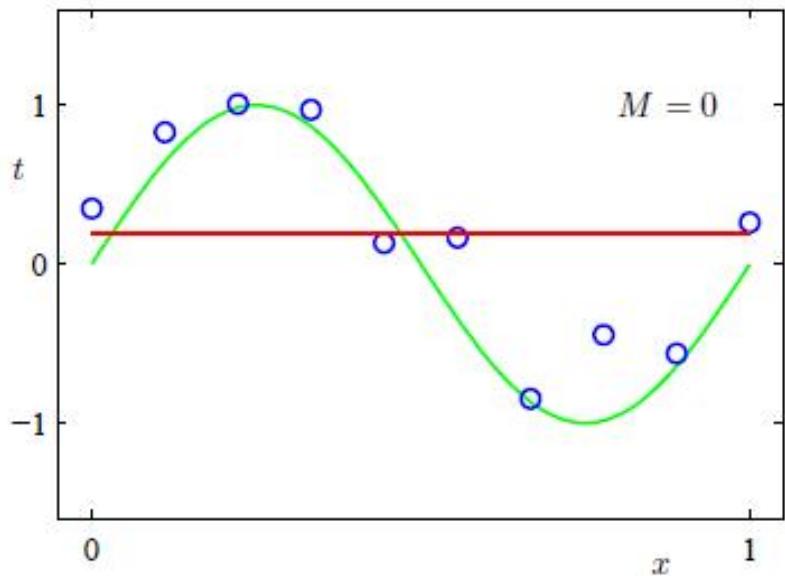


L2 regularization

# L1与L2正则项的比较

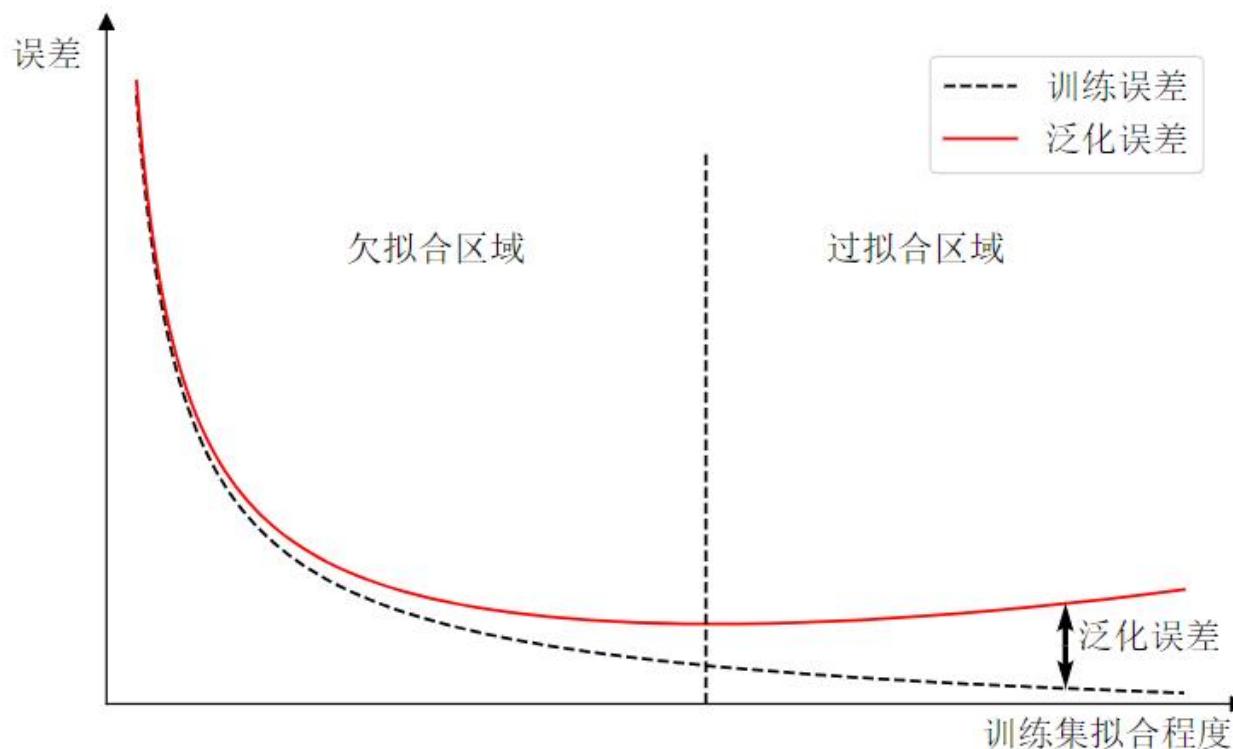


# 欠拟合和过拟合



# 欠拟合和过拟合

- **训练误差**: 模型在训练集上的误差
- **泛化误差**: 模型在测试集上的误差
- **泛化差距**: 训练误差和泛化误差之间的差距



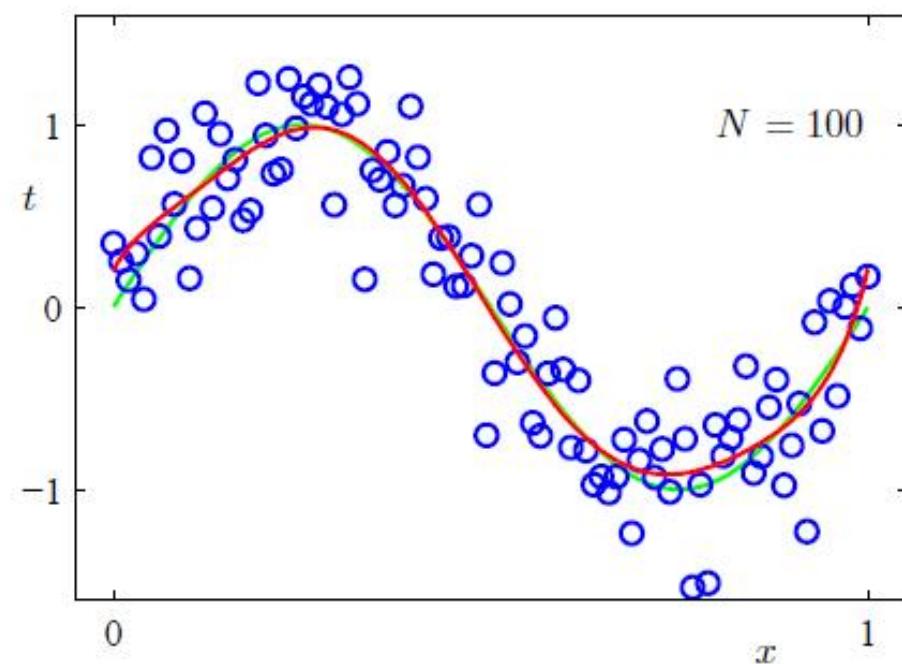
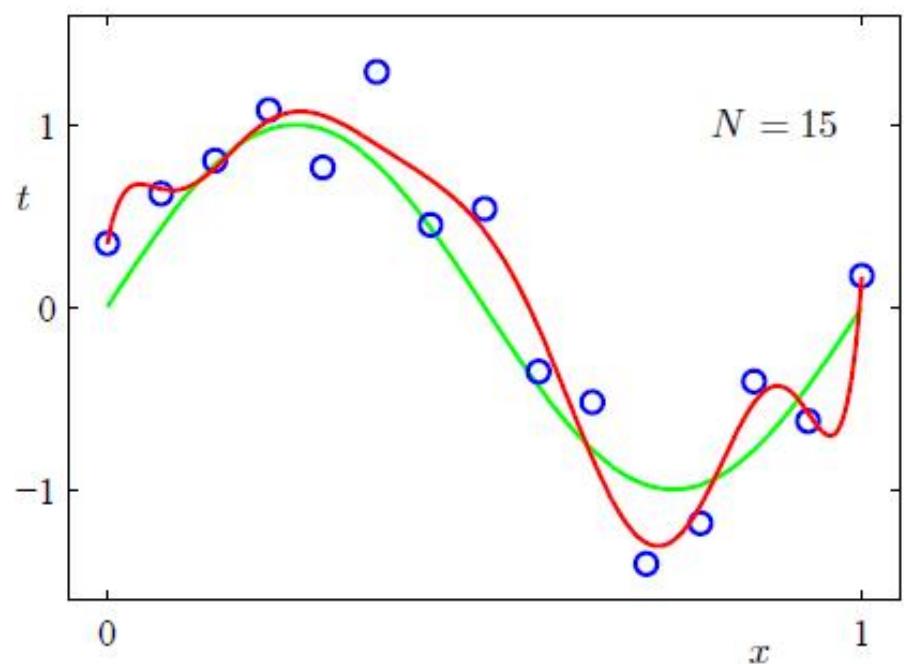
# 欠拟合的解决

---

- 增加模型的复杂度
- 提升模型的建模能力
- .....

# 过拟合的解决

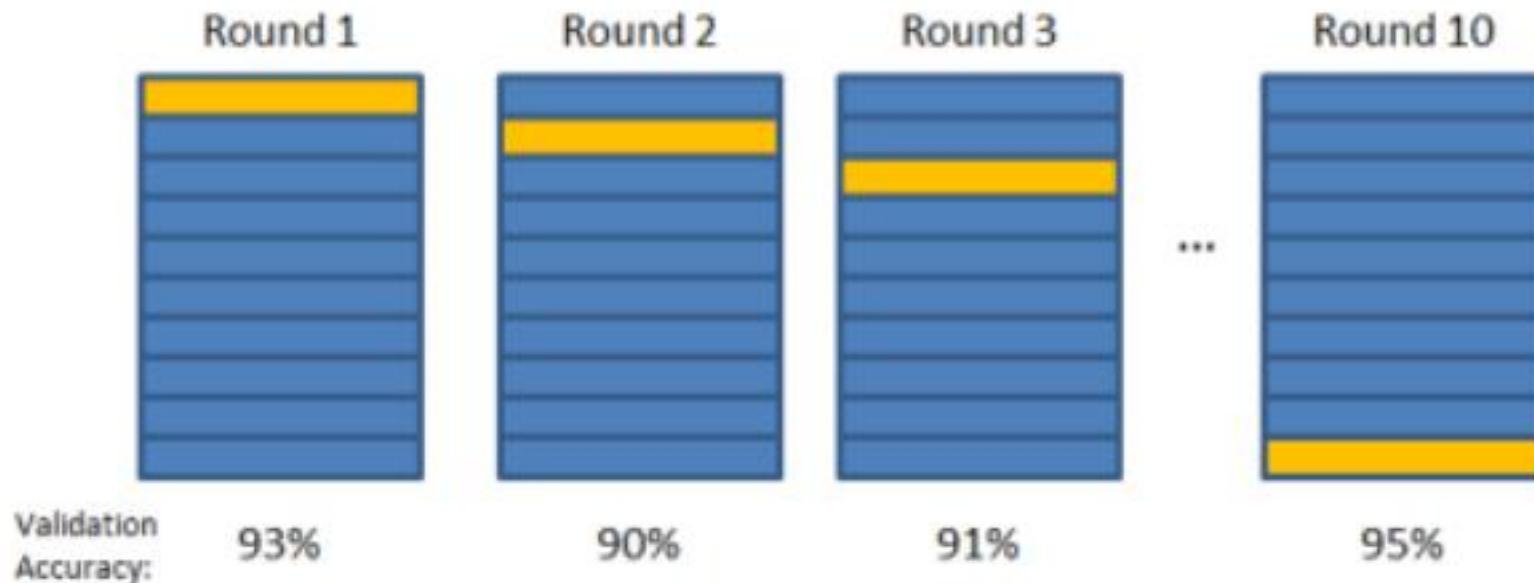
- 方法1：增加训练数据



# 过拟合的解决

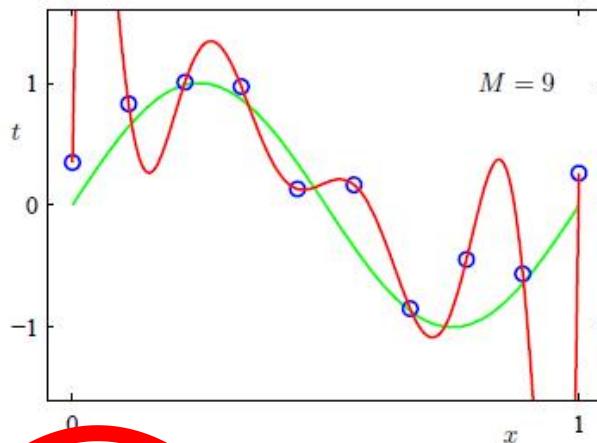
- 方法2：交叉验证
  - K-fold Cross validation

 Validation Set  
 Training Set



# 过拟合

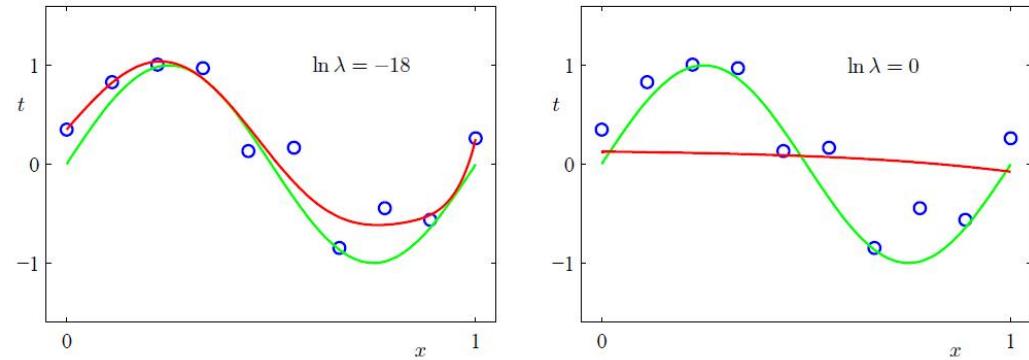
- 模型过拟合的表现



	$M = 0$	$M = 1$	$M = 3$	$M = 9$
$w_0^*$	0.19	0.82	0.31	0.35
$w_1^*$		-1.27	7.99	232.37
$w_2^*$			-25.43	-5321.83
$w_3^*$			17.37	48568.31
$w_4^*$				-231639.30
$w_5^*$				640042.26
$w_6^*$				-1061800.52
$w_7^*$				1042400.18
$w_8^*$				-557682.99
$w_9^*$				125201.43

# 过拟合的解决

## • 方法3：正则化



给  $M = 9$  的多项式添加正则项，并调节正则项的系数  $\lambda$

	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^*$	0.35	0.35	0.13
$w_1^*$	232.37	4.74	-0.05
$w_2^*$	-5321.83	-0.77	-0.06
$w_3^*$	48568.31	-31.97	-0.05
$w_4^*$	-231639.30	-3.89	-0.03
$w_5^*$	640042.26	55.28	-0.02
$w_6^*$	-1061800.52	41.32	-0.01
$w_7^*$	1042400.18	-45.95	-0.00
$w_8^*$	-557682.99	-91.53	0.00
$w_9^*$	125201.43	72.68	0.01

# 谢 谢

E-mail: [jywang@buaa.edu.cn](mailto:jywang@buaa.edu.cn)

Weibo: [@王静远BUAA](#)