南京航空航天大學
NANJING UNIVERSITY OF AERONAUTICS AND ASTRONAUTICS

机器学习基础

李明磊

南京航空航天大学 电子信息工程学院
E-mail: minglei_li@nuaa.edu.cn

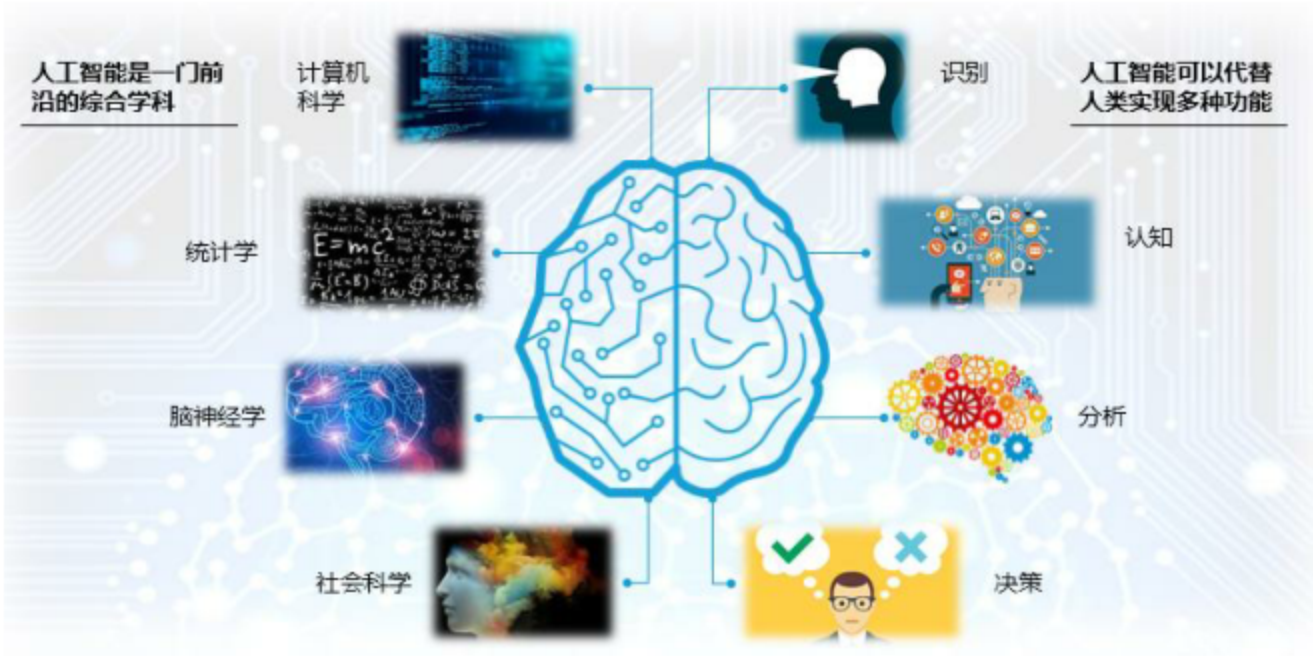
目录
CATALOG

机器学习简介

什么是人工智能→机器学习→神经网络→深度学习

https://www.bilibili.com/video/BV1vJ41147QU/?spm_id_from=333.337.search-card.all.click&vd_source=7d1d23b0822743a868a1515c73cf4bc5

人工智能（AI）



人工智能（Artificial Intelligence，AI）是指使计算机像人一样拥有智能能力
AI代替人类实现识别、认知，分析和决策等多种功能
主要的应用包括了智能翻译、机器对话、辅助驾驶、机器人场景理解和决策等

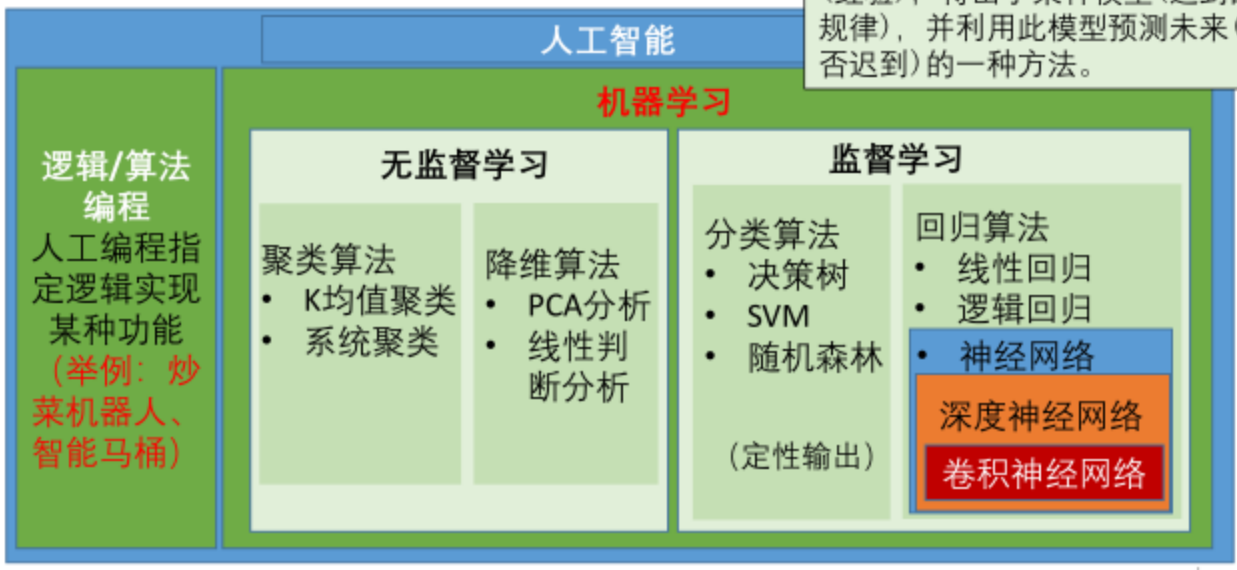
参考：腾讯 AI Lab 张潼主任带你轻松 get AI 新知识

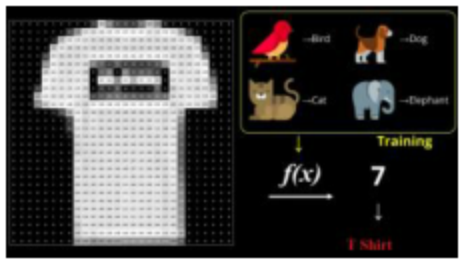
人工智能（AI）

人工智能（Artificial Intelligence，AI）是指使计算机像人一样拥有智能能力
AI代替人类实现识别、认知，分析和决策等多种功能
主要的应用包括了智能翻译、机器对话、辅助驾驶、机器人场景理解和决策等

人工智能（AI）

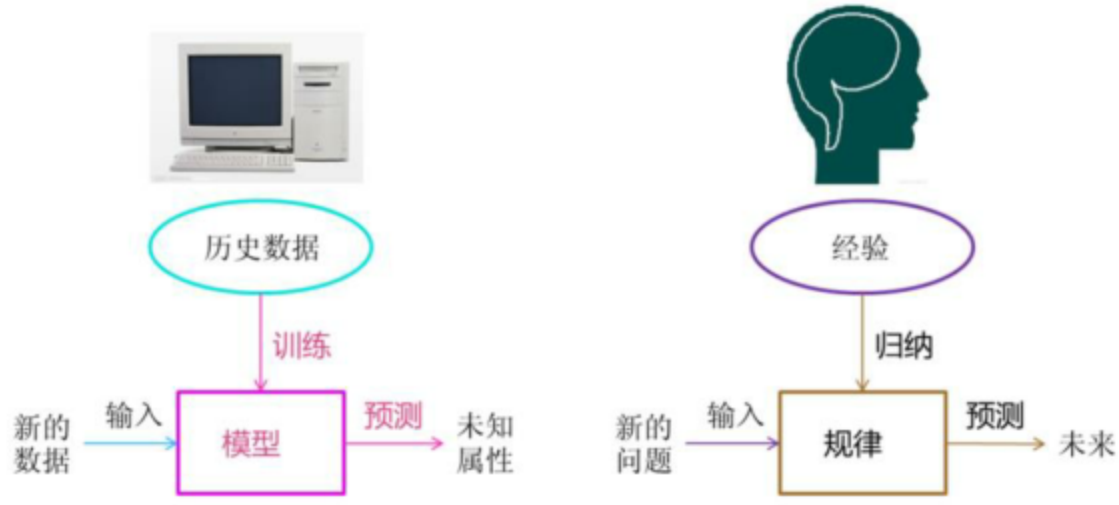
机器学习是计算机利用已有的数据（经验），得出了某种模型（迟到的规律），并利用此模型预测未来（是否迟到）的一种方法。



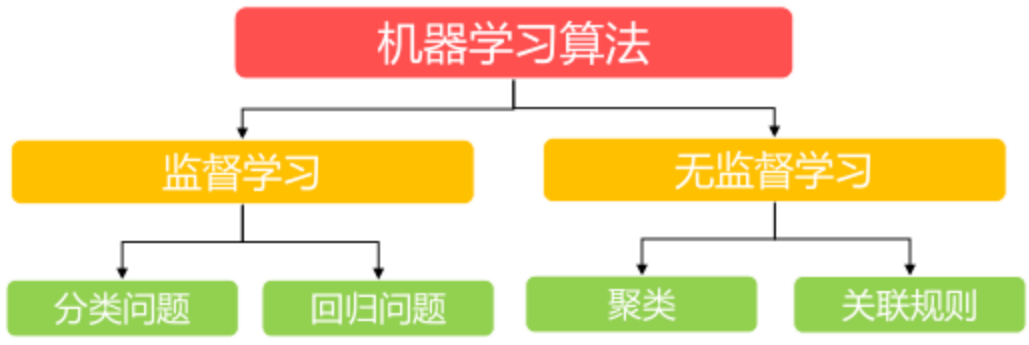


机器学习（ML）

机器学习（Machine Learning）是一种让计算机利用数据而不是指令来进行各种工作的方法。



机器学习的分类



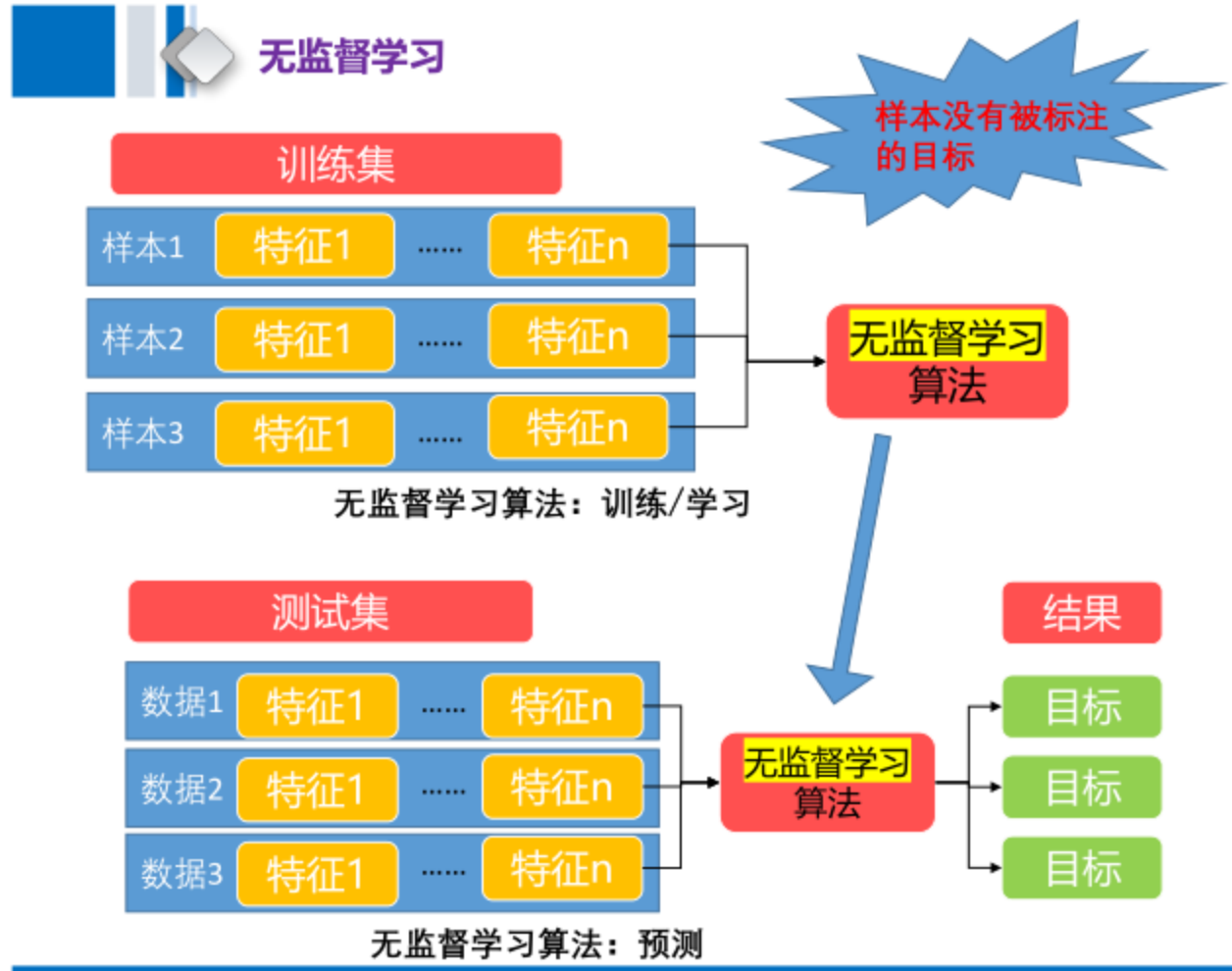
- **监督学习是利用标记数据进行训练，可以用于分类、回归等任务。**
- **无监督学习则是利用未标记数据进行训练，可以用于聚类、异常检测等任务。**

- 监督学习从给定的训练数据集中学习一个函数（模型），处理新数据时，根据这个函数（模型）预测结果；
- “训练数据”有一个明确的标识或结果，如“垃圾邮件”、“非垃圾邮件”；
- 在建立模型时，监督式学习建立一个学习过程，将预测结果与“测试数据”的实际结果进行比较，不断调整预测模型，直到模型的预测结果达到一个预期的准确率。常见的监督学习算法包括回归分析和统计分类。
- 回归分析分为线性回归和逻辑回归。

- 无监督式学习中，数据并不被特别标识，学习模型是为了推断出数据的一些内在结构；
- 常见的应用场景包括关联规则的学习以及聚类等。常见算法例如k-means算法等；

- 监督学习和无监督学习的区别：训练数据集的目标或类型是否被标注。

无监督学习



无监督学习算法：训练/学习

无监督学习算法：预测

监督学习

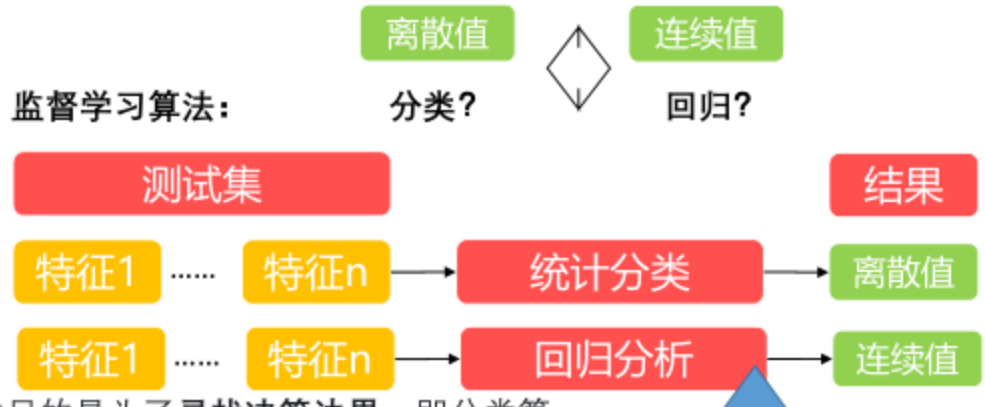
身高	发长	抽烟	性别
1.88	1.4cm	是	男
1.66	15.3cm	否	女
1.78	22.6cm	否	女

身高	发长	抽烟	性别
1.66	0cm	是	男
1.76	5.1cm	否	男
1.55	14.4cm	否	女

监督学习算法：训练/学习

监督学习算法：预测

监督学习



分类的目的是为了**寻找决策边界**，即分类算法得到是一个决策面，用于对数据集中的数据进行分类。

回归的目的是为了**找到最优拟合**，通过回归算法得到是一个最优拟合线，这个线条可以最好的接近数据集中的各个点。

不管是分类，还是回归，其本质是一样的，**都是对输入做出预测**，并且都是**监督学习**。

分类问题输出的值是定性的，回归问题输出的值是定量的

常常将回归问题和分类问题互相转化

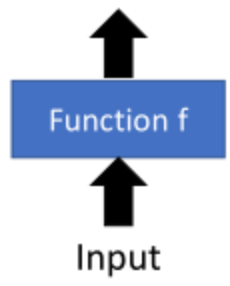
分类问题回归化：逻辑回归；回归问题分类化：年龄预测问题——>年龄段分类问题

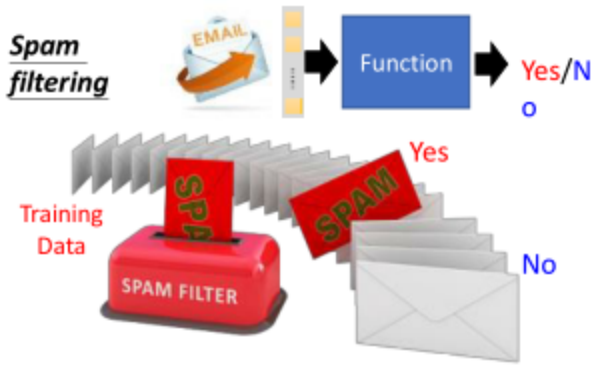
这都是为了处理实际问题时的方便之举，背后损失的是数学上的严谨性

监督学习 分类Classification

- Binary Classification

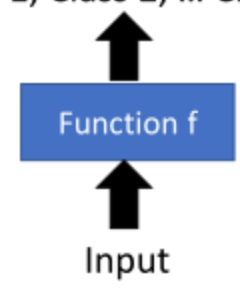
Yes or No





- Multi-class Classification

Class 1, Class 2, … Class N



回归算法简介

回归(Regression) Output a scalar

- Linear回归
- logistic回归

- Stock Market Forecast

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

- Self-driving Car

$f($  $)=$ 方向盘角度

- Recommendation

$f($ 使用者 A 商品B $)=$ 购买可能性

回归算法

监督学习

- 回归算法有两个重要的子类：线性回归和逻辑回归；
- 线性回归是拟合出一个线性函数，最佳匹配所有的数据。

  其处理的是数值问题，最后预测出的结果是数字，例如房价。
- 逻辑回归与线性回归非常类似，但处理的问题属于分类问题。

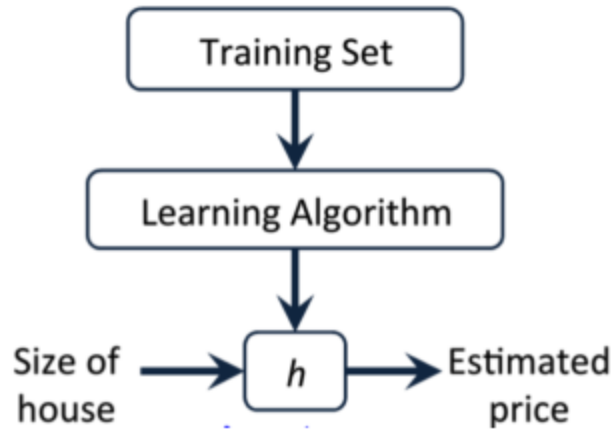
  逻辑回归预测结果是离散的分类，例如判断这封邮件是否是垃圾邮件，以及用户是否会点击此广告等等。

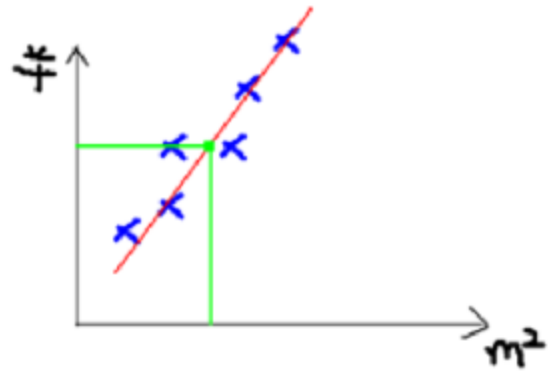
逻辑回归核心是一个回归函数，但是最终是用于分类。两方面原因：
1. 训练数据用的是带有分类标签的数据
2.训练过程是在寻找最佳“拟合”模型，用了非线性变换函数做“拟合”和预测。
这个过程等价于找逻辑回归内部区分正负（>0正样本，<0负样本）的线性函数f(x)。
非线性变换函数的变量是一个内部函数，这个内部函数是样本分类边界f(x)=0。整个过程就是回归了这个分类边界！

回归算法简介

线性回归

面积(m^2)	销售价钱（万元）
123	250
150	320
87	160
102	220
…	…





回归算法简介

线性回归

Training Set

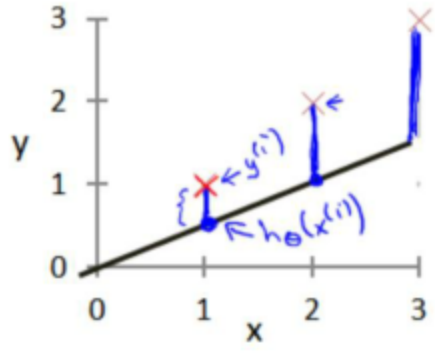
Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
...	...

m = 47

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

θ_i's: Parameters

How to choose θ_i's ?



Hypothesis:

$h_\theta(x) = \theta_0 + \theta_1 x$

Parameters:

θ_0, θ_1

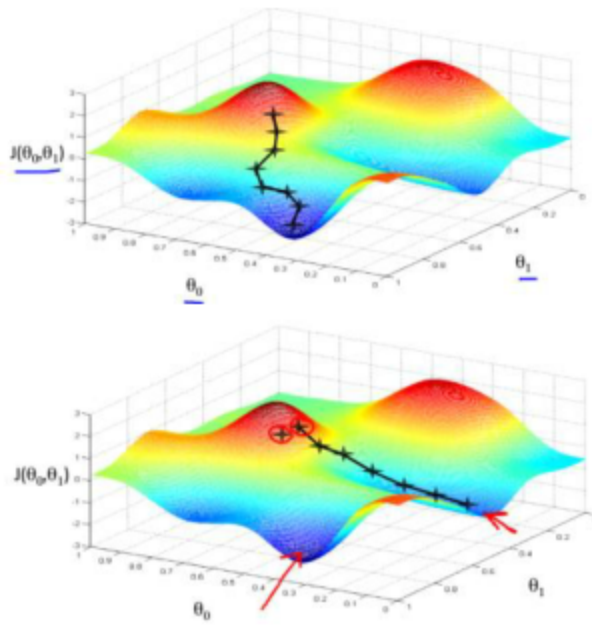
Cost Function:

$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

目标 ⟹ 选择出可以使得建模误差的平方和能够最小的模型参数

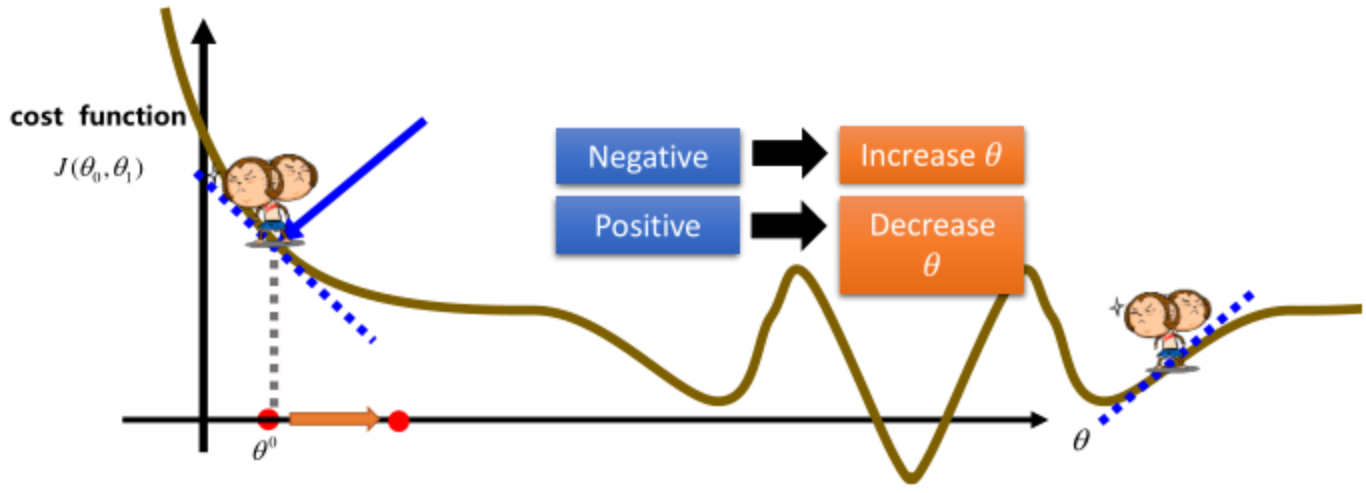
回归算法简介



梯度下降法

开始时我们随机选择一个参数的组合$(\theta_0, \theta_1, \cdots, \theta_n)$计算代价函数，然后我们寻找下一个能让代价函数值下降最多的参数组合。持续这么做直到得到一个局部最小值（local-minimum）。

线性回归



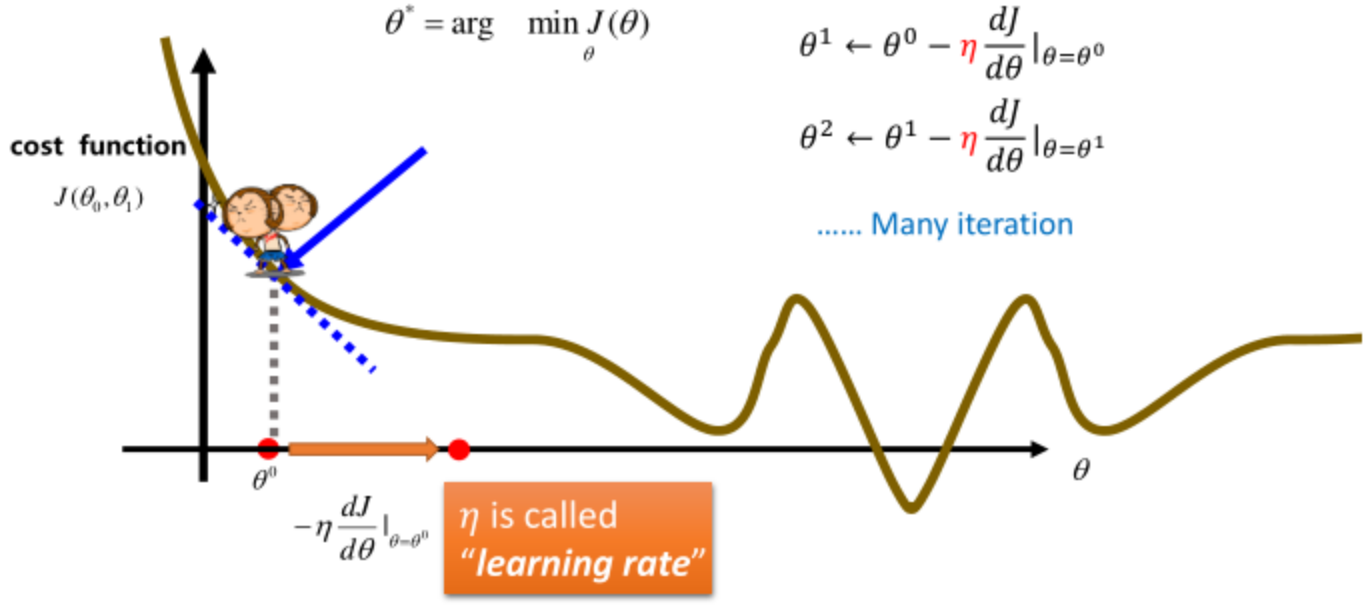
线性回归

$$\theta^* = \arg \min_{\theta} J(\theta)$$

$$\theta^1 \leftarrow \theta^0 - \eta \frac{dJ}{d\theta}|_{\theta=\theta^0}$$

$$\theta^2 \leftarrow \theta^1 - \eta \frac{dJ}{d\theta}|_{\theta=\theta^1}$$

..... Many iteration



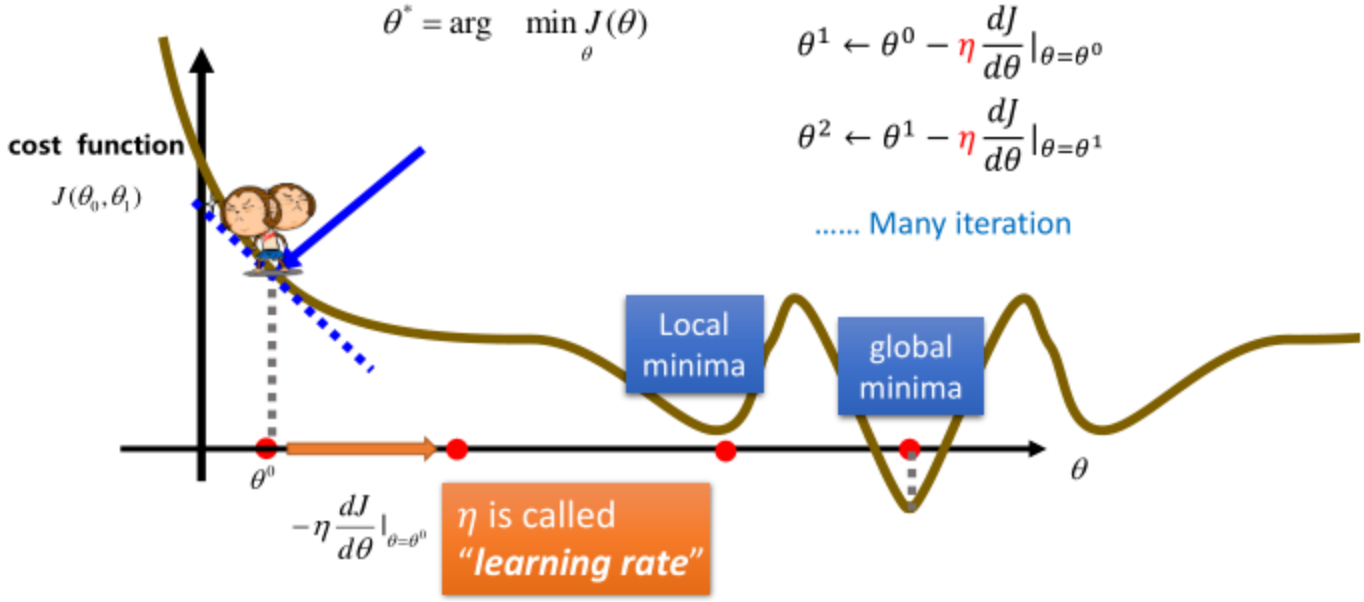
线性回归

$$\theta^* = \arg \min_{\theta} J(\theta)$$

$$\theta^1 \leftarrow \theta^0 - \eta \frac{dJ}{d\theta}|_{\theta=\theta^0}$$

$$\theta^2 \leftarrow \theta^1 - \eta \frac{dJ}{d\theta}|_{\theta=\theta^1}$$

..... Many iteration



线性回归

Gradient descent algorithm

repeat until convergence {

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

(for $j = 1$ and $j = 0$)

}

Linear Regression Model

$$h_\theta(x) = \theta_0 + \theta_1 x$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

线性回归

$$\frac{\partial}{\partial\theta_j}J(\theta_0\theta_1)=\frac{\partial}{\partial\theta_j}\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2$$

j=0 时：

$$\frac{\partial}{\partial\theta_0}J(\theta_0\theta_1)=\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})$$

j=1 时：

$$\frac{\partial}{\partial\theta_1}J(\theta_0\theta_1)=\frac{1}{m}\sum_{i=1}^{m}((h_\theta(x^{(i)})-y^{(i)})\cdot x^{(i)})$$

Repeat {

$$\theta_0:=\theta_0-\alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})$$

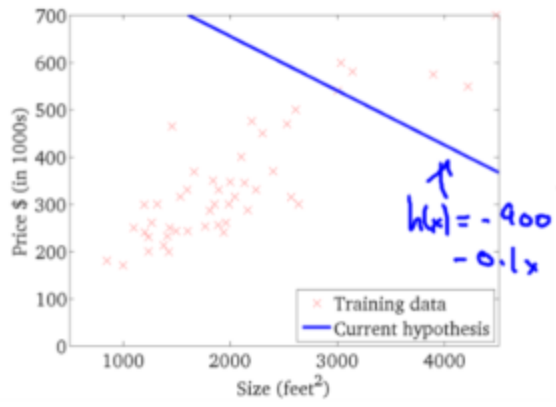
$$\theta_1:=\theta_1-\alpha\frac{1}{m}\sum_{i=1}^{m}((h_\theta(x^{(i)})-y^{(i)})\cdot x^{(i)})$$

}

线性回归

$h_\theta(x)$

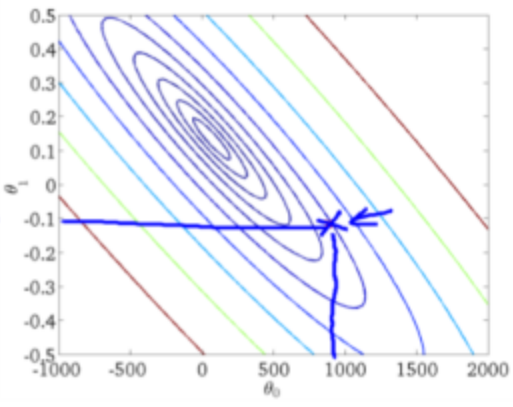
(for fixed θ_0, θ_1, this is a function of x)



变量空间

$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

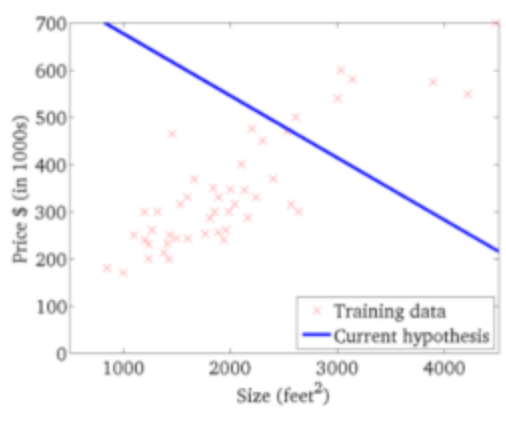


参数空间

线性回归

$h_\theta(x)$

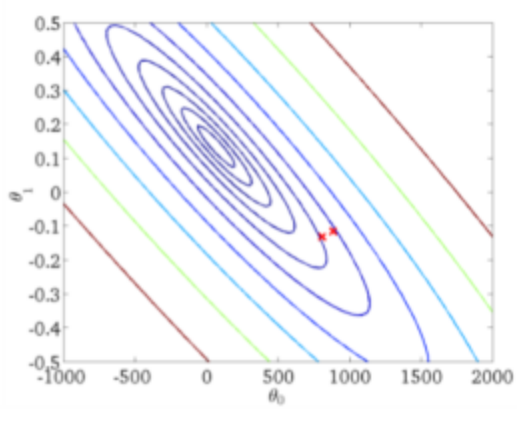
(for fixed θ_0, θ_1, this is a function of x)



变量空间

$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

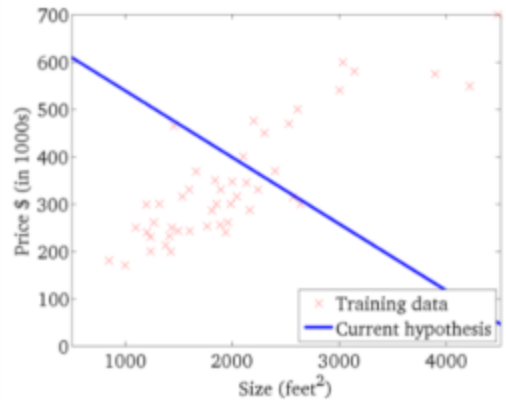


参数空间

线性回归

$h_\theta(x)$

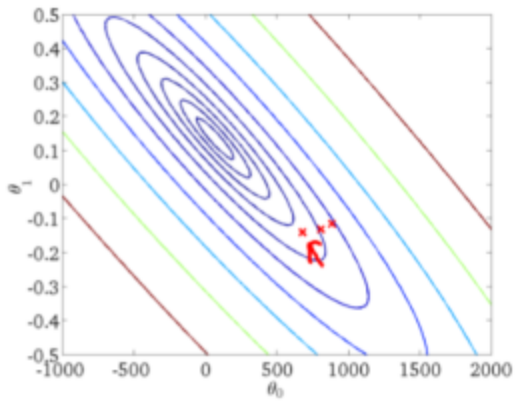
(for fixed θ_0, θ_1, this is a function of x)



变量空间

$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

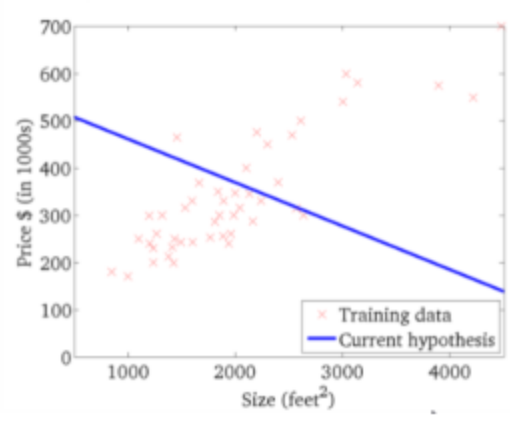


参数空间

线性回归

$h_\theta(x)$

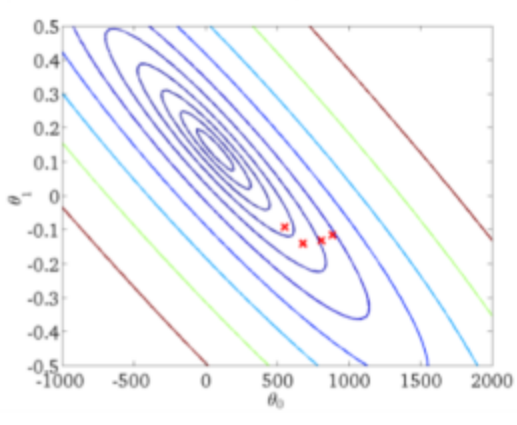
(for fixed θ_0, θ_1, this is a function of x)



变量空间

$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

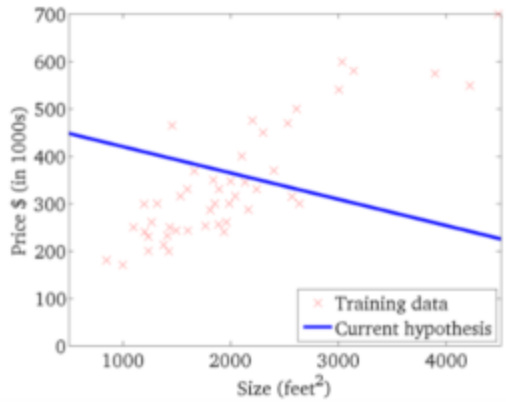


参数空间

线性回归

$h_\theta(x)$

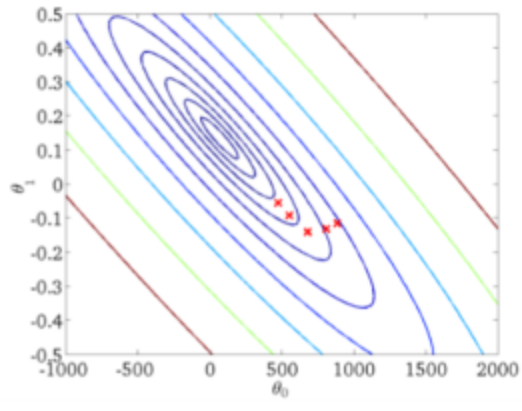
(for fixed θ_0, θ_1, this is a function of x)



变量空间

$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

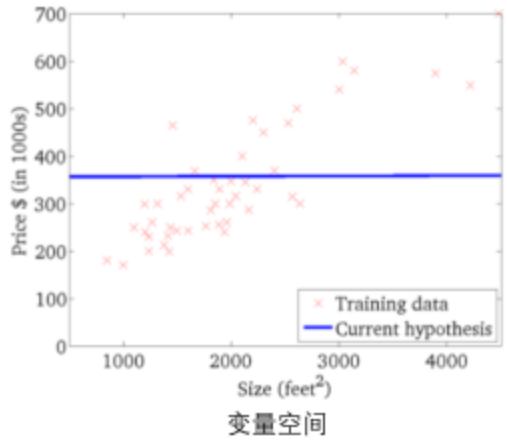


参数空间

线性回归

$h_\theta(x)$

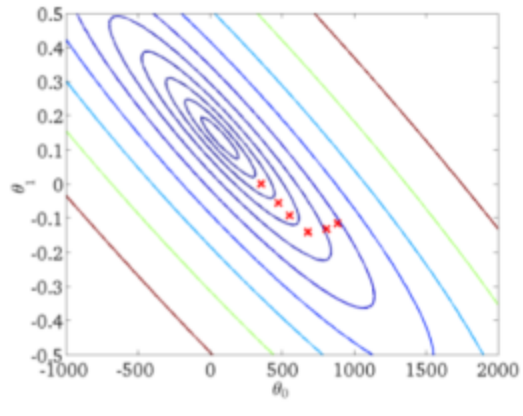
(for fixed θ_0, θ_1, this is a function of x)



变量空间

$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

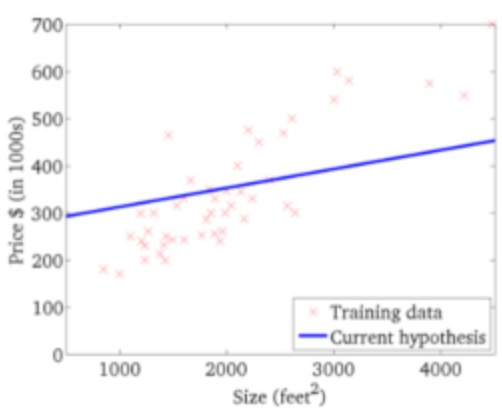


参数空间

线性回归

$h_\theta(x)$

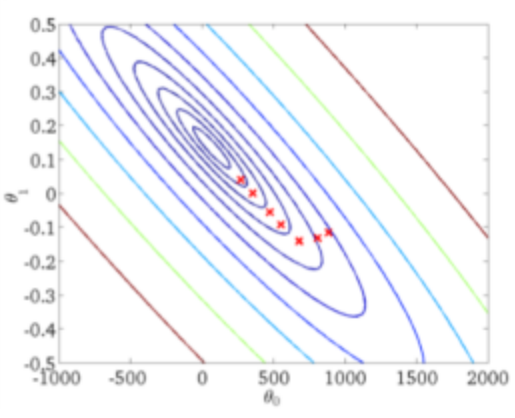
(for fixed θ_0, θ_1, this is a function of x)



变量空间

$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

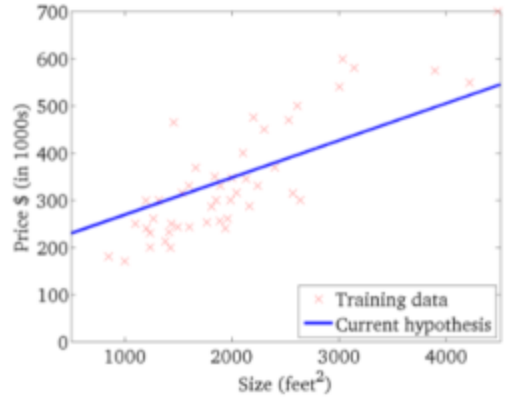


参数空间

线性回归

$h_\theta(x)$

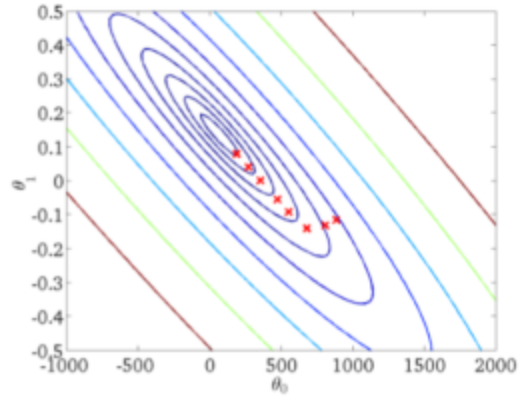
(for fixed θ_0, θ_1, this is a function of x)



变量空间

$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

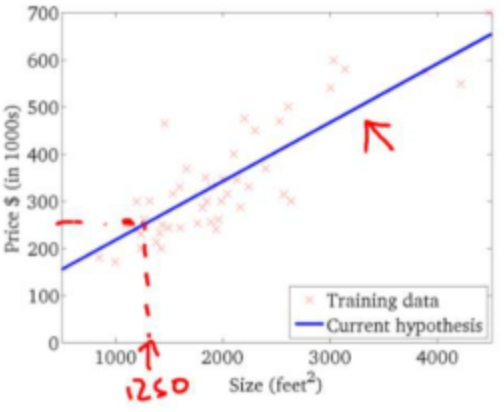


参数空间

线性回归

$h_\theta(x)$

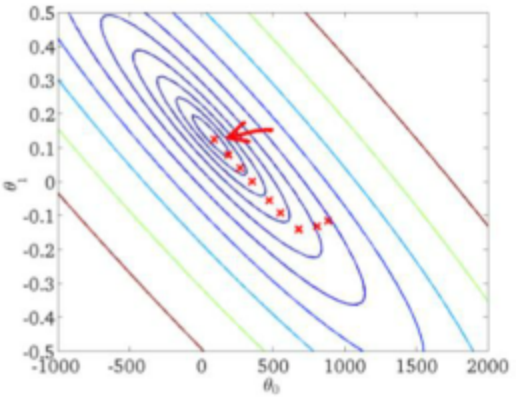
(for fixed θ_0, θ_1, this is a function of x)



变量空间

$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



参数空间

3分钟3D动画演示梯度下降优化算法_哔哩哔哩_bilibili

线性回归

Size (feet²)	Price ($1000)
x	y
2104	460
1416	232
1534	315
852	178
...	...

$$h_\theta(x) = \theta_0 + \theta_1 x$$

线性回归

Size (feet2)	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

模型中的特征为 $(x_1, x_2, \cdots, x_n)$，n代表特征的数量，$x^{(i)}$代表第i个训练实例，是特征矩阵中的第i行，是一个向量。例如，

$$x^{(2)} = \begin{pmatrix} 1416 \\ 3 \\ 2 \\ 40 \end{pmatrix}$$

线性回归

Size (feet2)	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

$x_j^{(i)}$ 代表特征矩阵中第i行的第j个特征，也就是第i个训练实例的第j个特征。

$$x_3^{(2)} = 2, x_1^{(4)} = 852$$

线性回归

支持多变量的假设h表示为：$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n$

这个公式中有n+1个参数和n个变量，为了使得公式能够简化一些，引入 $x_0 = 1$ ，则公式转化为：$h_\theta = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

此时模型中的参数是一个n+1维的向量，任何一个训练实例也都是n+1维的向量，特征矩阵X的维度是m*(n+1)。

因此公式可以简化为 $h_\theta(x) = \theta^T X$,其中上标 T 代表矩阵转置。

线性回归

正则化方程是通过求解下面的方程来找出使得代价函数最小的参数的:

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

假设我们的训练集特征矩阵为X（包含了$x_0 = 1$），并且我们的训练集结果为向量y，则利用正规方程解出向量:

$$\theta = \left(X^T X\right)^{-1} X^T y$$

其中，上标T代表矩阵转置，上标-1 代表矩阵的逆。

线性回归

Examples: $m = 4$.

x_0	Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix}$$

线性回归

X(0)	X(1)	X(2)	X(3)	X(4)	y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

$$\theta=\left(X^TX\right)^{-1}X^Ty \Rightarrow \left(\begin{pmatrix}1&1&1&1\\2104&1416&1534&852\\5&3&3&2\\1&2&2&1\\45&40&30&36\end{pmatrix}\times\begin{pmatrix}1&2104&5&1&45\\1&1416&3&2&40\\1&1534&3&2&30\\1&852&2&1&36\end{pmatrix}\right)^{-1}\times\begin{pmatrix}1&1&1&1\\2104&1416&1534&852\\5&3&3&2\\1&2&2&1\\45&40&30&36\end{pmatrix}\times\begin{pmatrix}460\\232\\315\\178\end{pmatrix}$$

梯度下降	正规方程
需要选择学习率 α	不需要
需要多次迭代	一次运算得出
当特征数量 n 大时也能较好适用	需要计算$(X^TX)^{-1}$ 如果特征数量 n 较大则运算代价大，因为矩阵逆的计算时间复杂度为 $O(n^3)$，通常来说当 n 小于 10000 时还是可以接受的
适用于各种类型的模型	只适用于线性模型，不适合逻辑回归模型等其他模型

回归算法——逻辑回归

如何理解逻辑回归（logistic regression）？ - 知乎 (zhihu.com)

回归算法——逻辑回归

分类问题

邮件分类：垃圾邮件/正常邮件?

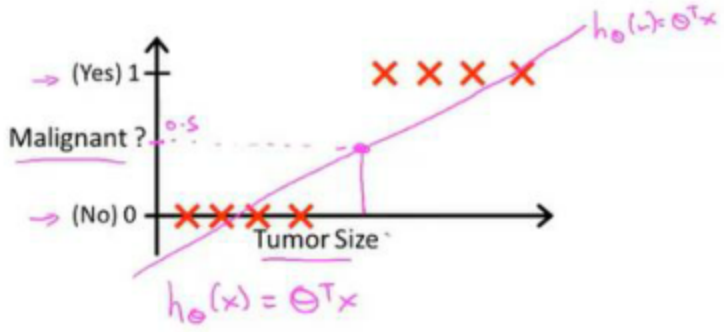
在线转账：欺诈（Yes/No）

肿瘤诊断： Malignant(恶性)/Benign(良性)

→ **Logistic Regression**

回归算法——逻辑回归

如果直接使用线性回归



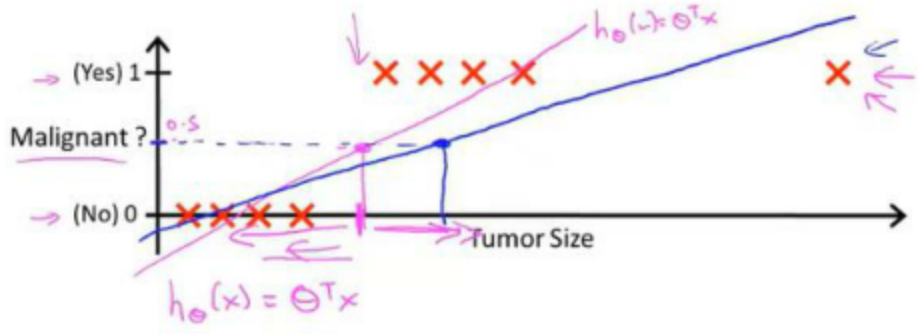
→ Threshold classifier output $h_\theta(x)$ at 0.5:

→ If $h_\theta(x) \geq 0.5$, predict "y = 1"

If $h_\theta(x) < 0.5$, predict "y = 0"

回归算法——逻辑回归

如果直接使用线性回归



- Threshold classifier output $h_\theta(x)$ at 0.5:
  - If $h_\theta(x) \geq 0.5$, predict "y = 1"
  - If $h_\theta(x) < 0.5$, predict "y = 0"

引入逻辑回归

如果直接使用线性回归

Classification: y = 0 or 1

预测值可以是>1或<0

$h_\theta(x)$ can be > 1 or < 0

逻辑回归的原理是用逻辑函数把线性回归的结果从(-∞,∞)映射到(0,1)

Logistic Regression: $0 \leq h_\theta(x) \leq 1$

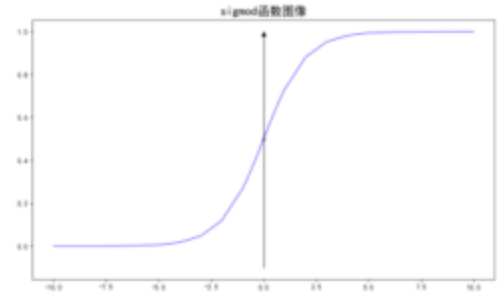
伯努利分布(通俗解释):

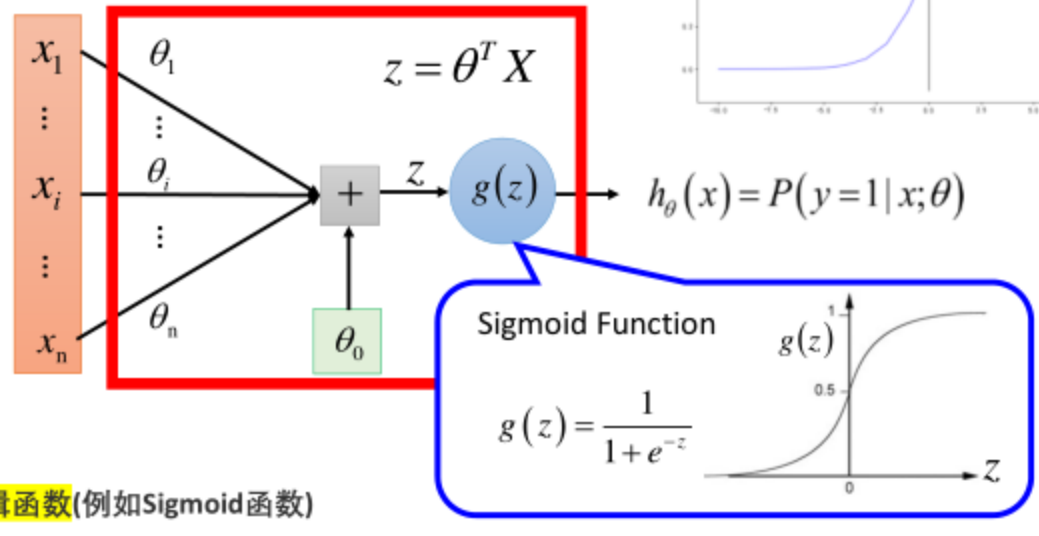
一件事情，只有两种可能的结果。

伯努利分布描述了其中一种结果的概率为a，

另一种结果的概率为100%-a。

逻辑回归



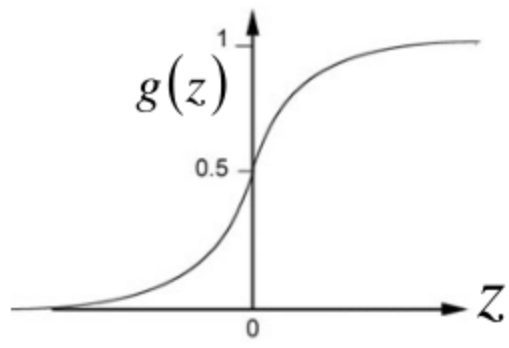


逻辑函数(例如Sigmoid函数)

把线性回归函数的结果**Z**，放到**Sigmoid**函数中去，就构造了**逻辑回归函数**。

逻辑回归

Sigmoid Function $g(z) = \frac{1}{1+e^{-z}}$



在逻辑回归中，我们预测：

当 h_θ 大于等于 0.5 时，预测 y=1

当 h_θ 小于 0.5 时，预测 y=0

根据上面绘制出的 S 形函数图像，我们知道当

z=0 时 g(z)=0.5

z>0 时 g(z)>0.5

z<0 时 g(z)<0.5

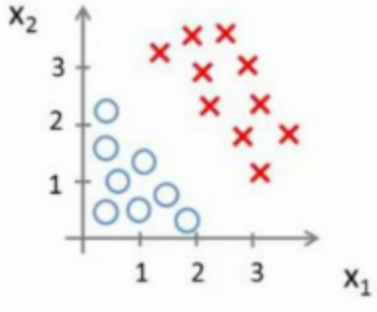
又 $z=\theta^T X$，即：

$\theta^T X$ 大于等于 0 时，预测 y=1

$\theta^T X$ 小于 0 时，预测 y=0

逻辑回归

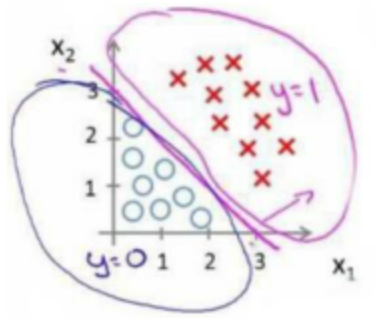
Decision Boundary



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

-3

并且参数θ是向量[-3 1 1]。则当$-3+x_1+x_2$大于等于0，即x_1+x_2大于等于3时，模型将预测y=1。



绘制直线 $x_1+x_2=3$，这条线便是我们模型的分界线，将预测为1的区域和预测为0的区域分隔开。

逻辑回归

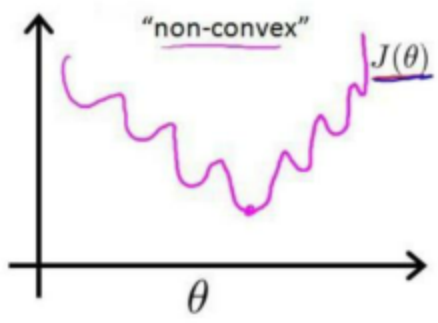
Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})\}$

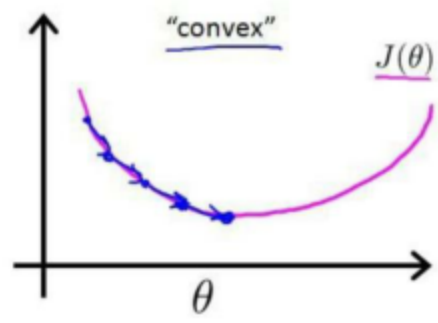
m examples $\quad x \in \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix} \quad x_0 = 1, y \in \{0, 1\}$

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

How to choose parameters θ ?

逻辑回归



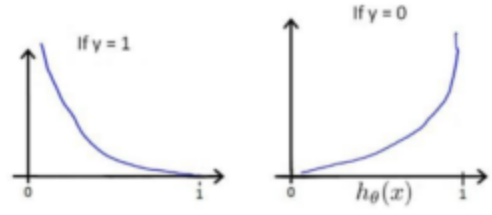


$$h_\theta(x) = \frac{1}{1+e^{-\theta^T X}}$$

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\mathrm{Cost}\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right)$$

$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$



逻辑回归

$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

$$Cost(h_\theta(x), y) = -y \times \log(h_\theta(x)) - (1-y) \times \log(1 - h_\theta(x))$$

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right) + \left(1-y^{(i)}\right)\log\left(1-h_\theta\left(x^{(i)}\right)\right)]$$

Repeat {

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

(simultaneously update all θ_j)

}

Repeat {

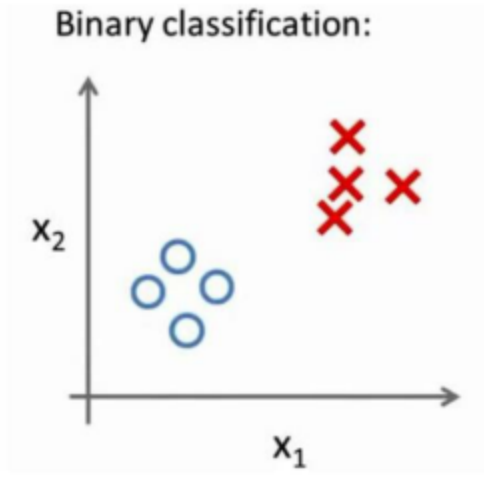
$$\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

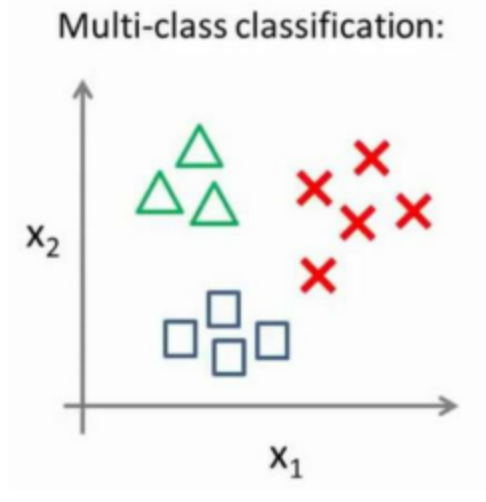
(simultaneously update all θ_j)

}

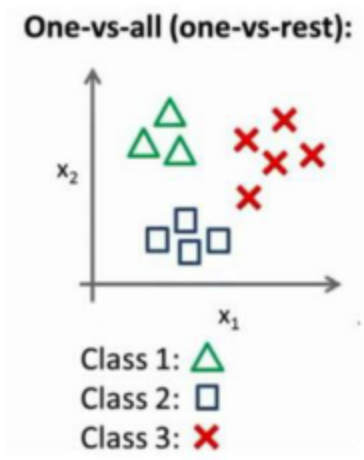
形式与线性回归相似！

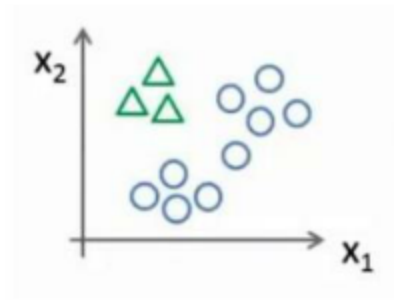
逻辑回归

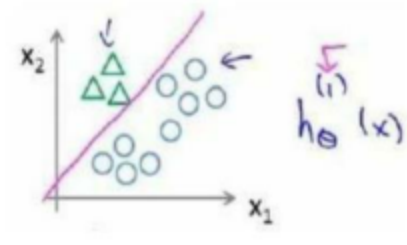


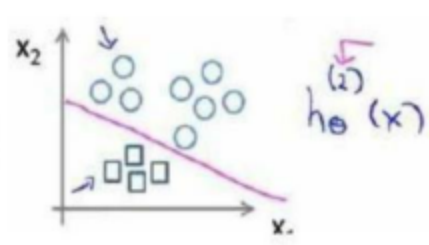


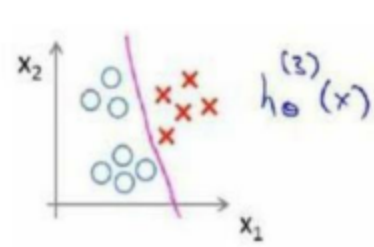
逻辑回归











做预测时，将所有的分类机都运行一遍，然后对每一个输入变量，都选择最高可能性的输出变量。

	Linear Regression	***Logistic Regression***
Step 1:	$h_\theta(x) = \sum_i \theta_i x_i + \theta_0$ Output: any value	$h_\theta(x) = g(\sum_i \theta_i x_i + \theta_0)$ Output: between 0 and 1
Step 2:	Training data: $(x^n, \hat{y}^n)$ $\hat{y}^n$: a real number $J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}(h_\theta(x^i) - y^{(i)})^2$	Training data: $(x^n, \hat{y}^n)$ $\hat{y}^n$: 1 for class 1, 0 for class 2 $J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)})$
Step 3:	Logistic regression: $\theta_j \leftarrow \theta_i - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)} - y^{(i)})x_j^{(i)}$ Linear regression: $\theta_j \leftarrow \theta_i - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)} - y^{(i)})x_j^{(i)}$	

How's the results?

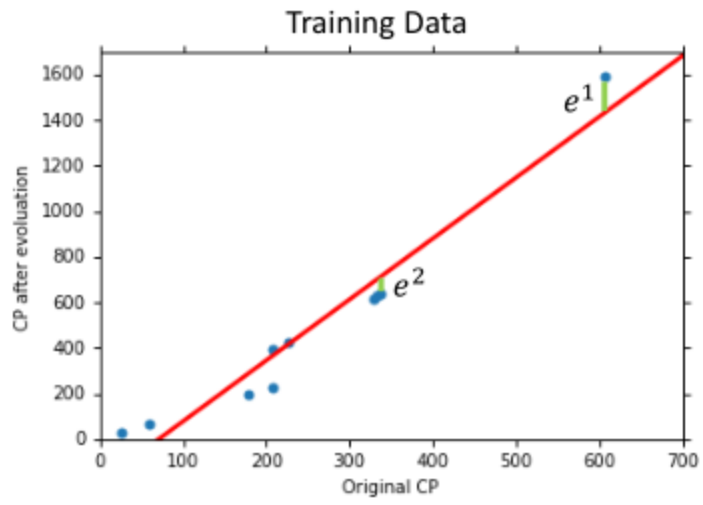
$y = b + w \cdot x_{cp}$

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)

Selecting another Model

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

Best Function

b = -10.3

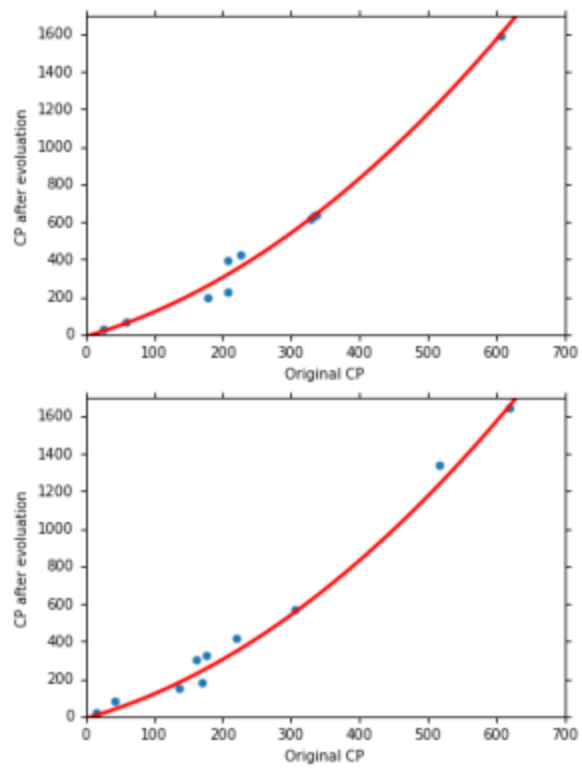
w_1 = 1.0, w_2 = 2.7 x 10^{-3}

Average Error = 15.4

Testing:

Average Error = 18.4

Better! Could it be even better?



Selecting another Model

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

Best Function

b = 6.4, w_1 = 0.66

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

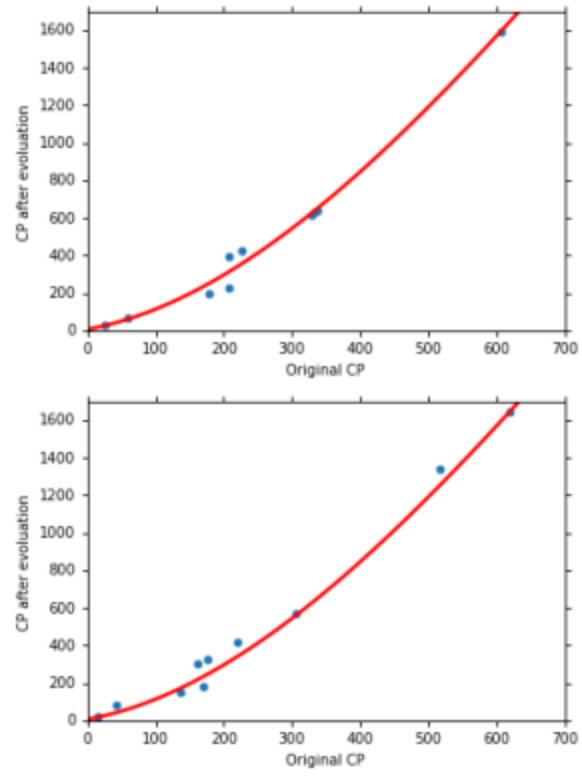
$w_3 = -1.8 \times 10^{-6}$

Average Error = 15.3

Testing:

Average Error = 18.1

Slightly better.
How about more complex model?



Selecting another Model

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

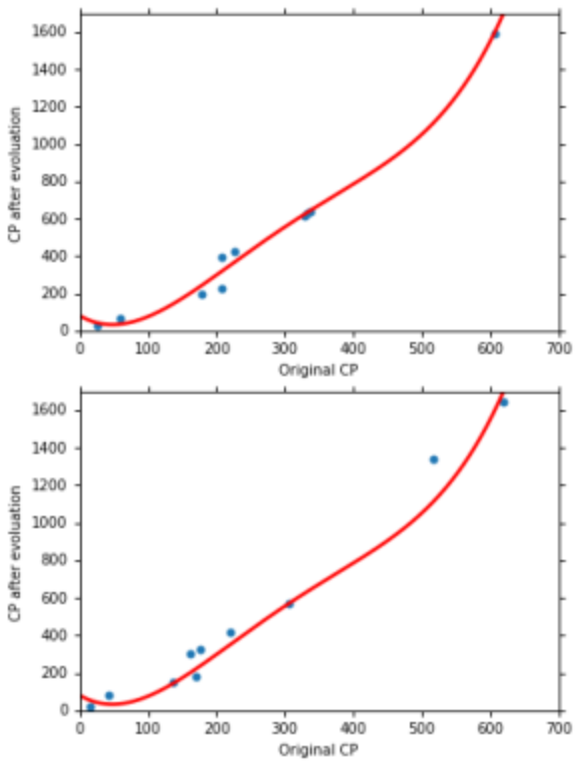
Best Function

Average Error = 14.9

Testing:

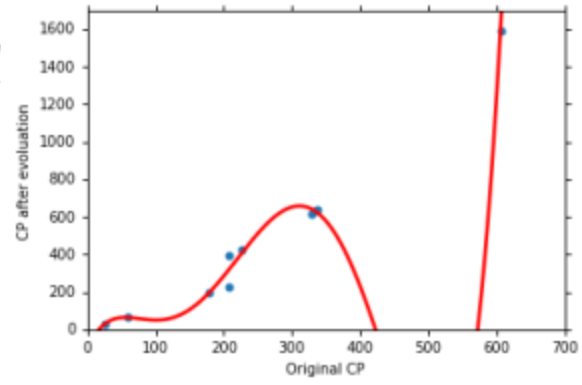
Average Error = 28.8

The results become worse ...



Selecting another Model

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



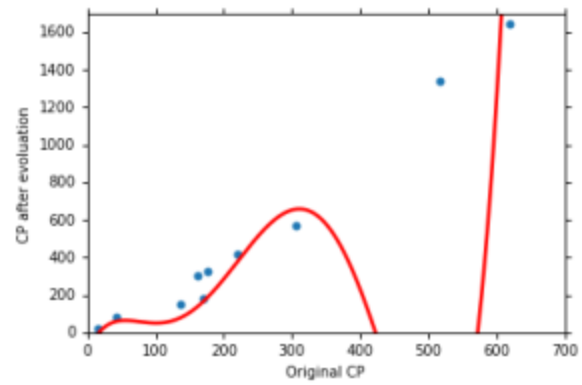
Best Function

Average Error = 12.8

Testing:

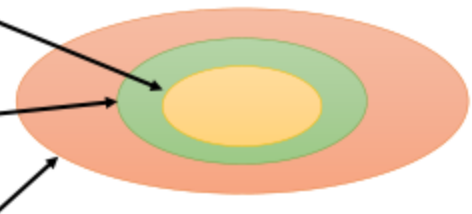
Average Error = 232.1

The results are so bad.



Model Selection

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



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

过拟合与正则化

处理过拟合问题的方法：

- 1. 丢弃一些不能帮助我们正确预测的特征。

可以是手工选择保留哪些特征，或者使用一些模型选择的算法来帮忙（例如PCA）

- 2. 正则化。

保留所有的特征，但是减少参数的大小（magnitude）。

过拟合与正则化

$$h_\theta(x)=\theta_0+\theta_1x_1+\theta_2x_2^2+\theta_3x_3^3+\theta_4x_4^4$$

正是那些高次项导致了过拟合的产生，所以如果我们能让这些高次项的系数接近于 0 的话，我们就能很好的拟合了。

所以我们要做的就是在一定程度上减小这些参数 θ 的值，这就是正则化的基本方法。我们决定要减少θ_3和θ_4的大小，我们要做的便是修改代价函数，在其中θ_3和θ_4设置一点惩罚。这样做的话，我们在尝试最小化代价时也需要将这个惩罚纳入考虑中，并最终导致选择较小一些的θ_3和θ_4。修改后的代价函数如下：

$$\min_\theta \frac{1}{2m}\sum_{i=1}^{m}((h_\theta(x^{(i)})-y^{(i)})^2+1000\theta_3^2+10000\theta_4^2)$$

通过这样的代价函数选择出的θ_3和θ_4对预测结果的影响就比之前要小许多。

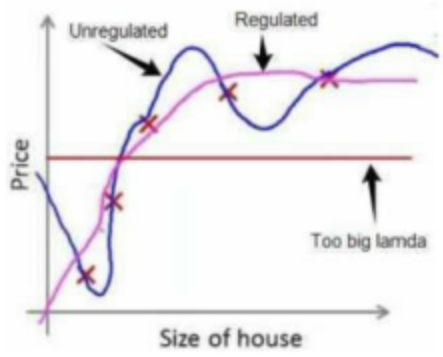
过拟合与正则化

假如我们有非常多的特征，我们并不知道其中哪些特征我们要惩罚，我们将对所有的特征进行惩罚，并且让代价函数最优化的方法来选择这些惩罚的程度。这样的结果是得到了一个较为简单的能防止过拟合问题的假设：

$$J(\theta)=\frac{1}{2m}\left[\sum_{i=1}^{m}((h_\theta(x^{(i)}-y^{(i)})^2+\lambda\sum_{j=1}^{n}\theta_j^2)\right]$$

其中λ又称为正则化参数（Regularization Parameter）。

过拟合与正则化



$$J(\theta)=\frac{1}{2m}\left[\sum_{i=1}^{m}((h_\theta(x^{(i)}-y^{(i)})^2+\lambda\sum_{j=1}^{n}\theta_j^2)\right]$$

如果选择的正则化参数 λ 过大，则会把所有的参数都最小化了，导致模型变成

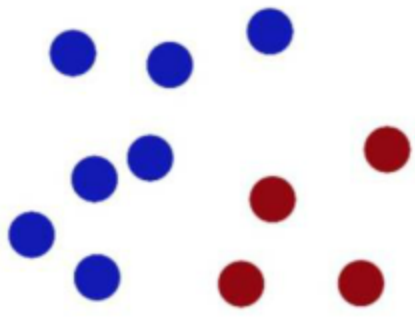
$$h_\theta(x)=\theta_0$$

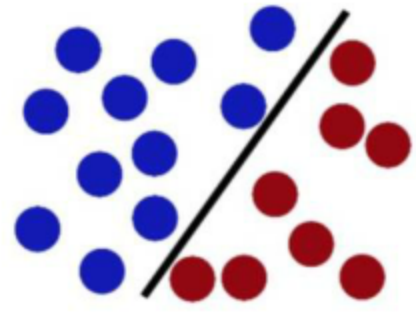
也就是上图中红色直线所示的情况，造成欠拟合。

支持向量机
SVM

支持向量机——引入

桌子上有规律放了两种颜色的球，如何用一根棍分开它们？要求：尽量在放更多球之后，仍然适用。

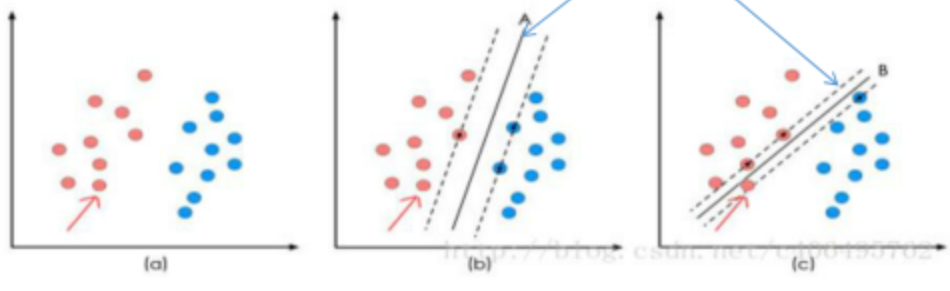




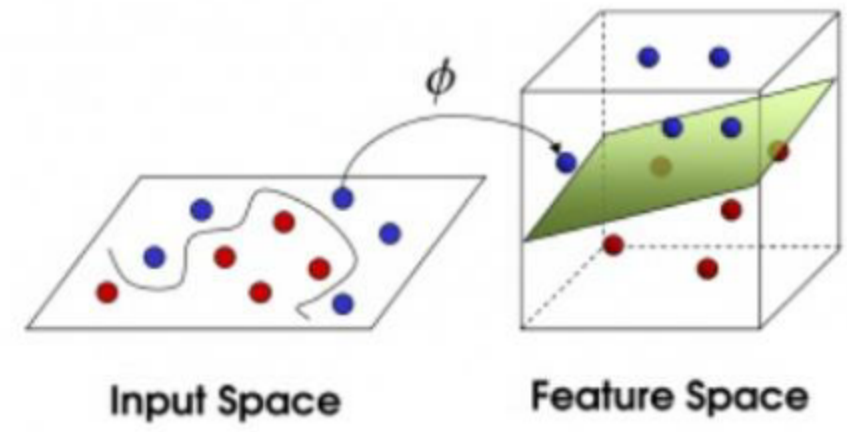
决策面

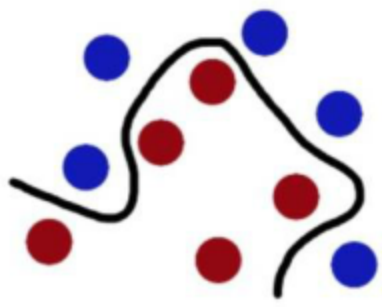
目标函数——分类间隔

优化对象——决策面

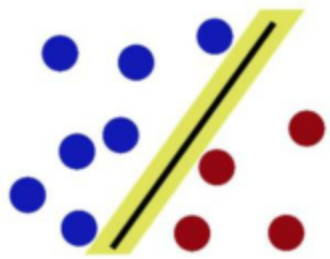


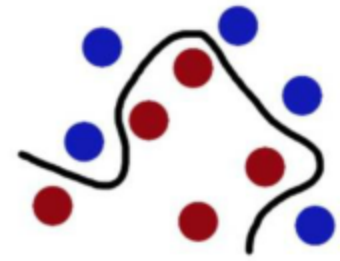
支持向量机——引入





支持向量机——引入





当一个分类问题，数据是线性可分的，也就是用一条线性直线，就可以将两种小球分开的时候，只要将分割线的位置放在让小球距离线的距离最大化的位置即可，寻找这个最大间隔的过程，就叫做最优化。

但是，现实往往是很残酷的，一般的数据是线性不可分的，也就是找不到一条线将两种小球很好的分类。这个时候，我们就需要将小球拍起，用一张纸代替线将小球进行分类。想要让数据飞起，我们需要的东西就是核函数(kernel)，用于切分小球的纸，就是超平面。

支持向量机——线性SVM

$$y = ax + b$$

$$x_2 = ax_1 + b$$

$$ax_1 - x_2 + b = 0$$

$$\begin{bmatrix} a & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + b = 0$$

$$\omega^T x + \gamma = 0$$

$$\begin{cases} \omega = [\omega_1, \quad \omega_2]^T \\ x = [x_1, \quad x_2]^T \end{cases}$$

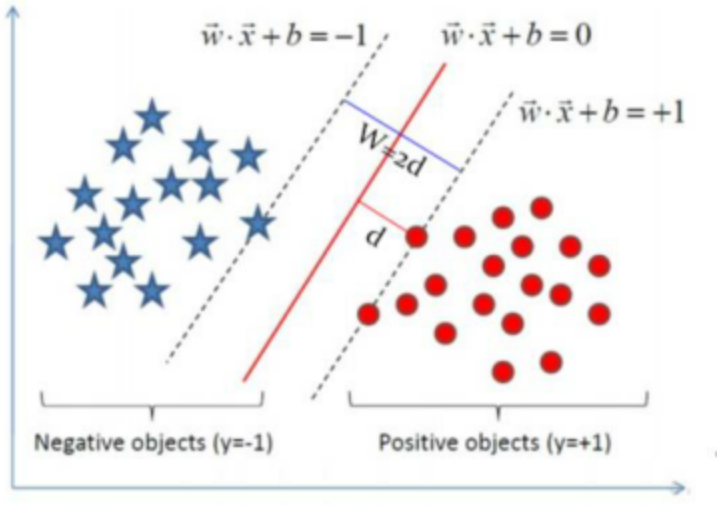
$$\omega^T x + \gamma = 0$$

$$\begin{cases} \omega = [\omega_1, \omega_2, \cdots, \omega_n]^T \\ x = [x_1, x_2, \cdots, x_n]^T \end{cases}$$

线性“决策面”方程

二维空间 → n维空间

支持向量机——线性SVM



“分类间隔”方程

二维情况

$Ax+By+C=0 \qquad P(x_0,y_0)$

$$d=\left|\frac{Ax_0+By_0+C}{\sqrt{A^2+B^2}}\right|$$

通适的线性情况

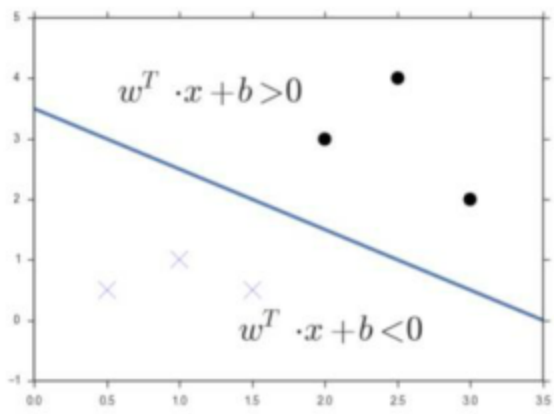
$$d=\frac{|\omega^T x+\gamma|}{\|\omega\|}$$

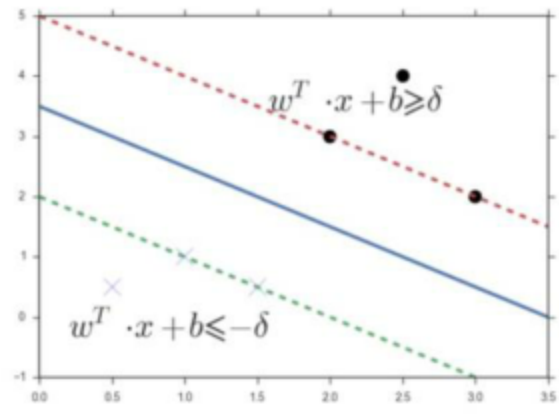
$$\omega=[\omega_1,\omega_2]^T \qquad \|\omega\|=\sqrt{\omega_1^2+\omega_2^2}$$

约束条件

- 如何判断超平面是否将样本点正确分类?
- 知道想求距离d的最大值，首先需要找到支持向量上的点，怎么在众多的点中选出支持向量上的点呢?

支持向量机——线性SVM





约束条件

- 如何判断超平面是否将样本点正确分类?
- 知道想求距离d的最大值，首先需要找到支持向量上的点，怎么在众多的点中选出支持向量上的点呢?

支持向量机——线性SVM

$$\begin{cases} \omega^T x+\gamma \geq 1 & \cdots \bullet \\ \omega^T x+\gamma \leq -1 & \cdots \times \end{cases} \qquad y_i=\begin{cases} +1 & x_i=\bullet \\ -1 & x_i=\times \end{cases}$$

$$\begin{cases} \omega^T x_i+\gamma \geq 1 & \forall y_i=1 \\ \omega^T x_i+\gamma \leq -1 & \forall y_i=-1 \end{cases}$$

$$y_i(\omega^T x_i+\gamma)\geq 1 \quad \forall x_i$$ 约束条件

支持向量机——线性SVM

$$d=\frac{|\omega^T x+\gamma|}{\|\omega\|}$$

$$|\omega^T x_i+\gamma|=1 \quad \forall \text{支持向量上的样本点} x_i$$

$$d=\frac{|\omega^T x+\gamma|}{\|\omega\|}=\frac{1}{\|\omega\|}$$

目标 在限制条件$y_i(\omega^T x_i+\gamma)\geq 1$下，找到合适的参数$(\omega,\gamma)$，使得$\frac{1}{2}\|\omega\|^2$最小

支持向量机——线性SVM

无约束优化问题 $\min f(x)$

使用费马大定理(Fermat)，即求取函数f(x)的导数，然后令其为零，可以求得候选最优值，再在这些候选值中验证；如果是凸函数，可以保证是最优解。

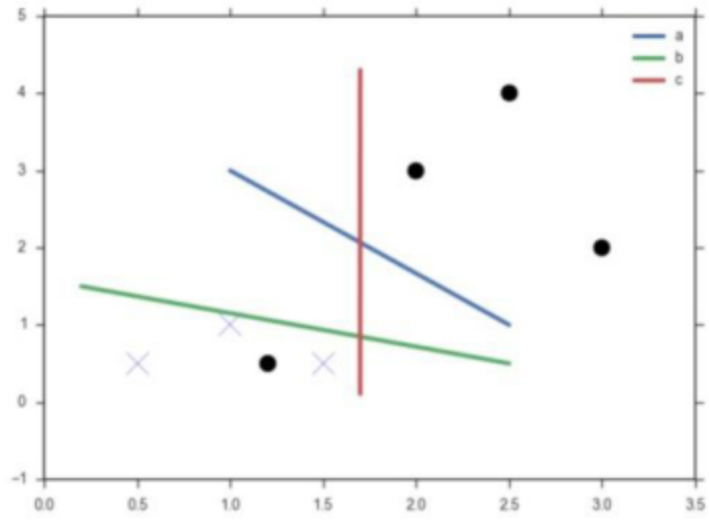
有等式约束的优化问题 $\min f(x)$ $s.t.\quad h_i(x)=0, i=1,2,\cdots,n$

使用拉格朗日乘子法，即把等式约束$h_i(x)$用一个系数与f(x)写为一个式子，通过拉格朗日函数对各个变量求导，令其为零，可以求得候选值集合，然后验证求得最优值。

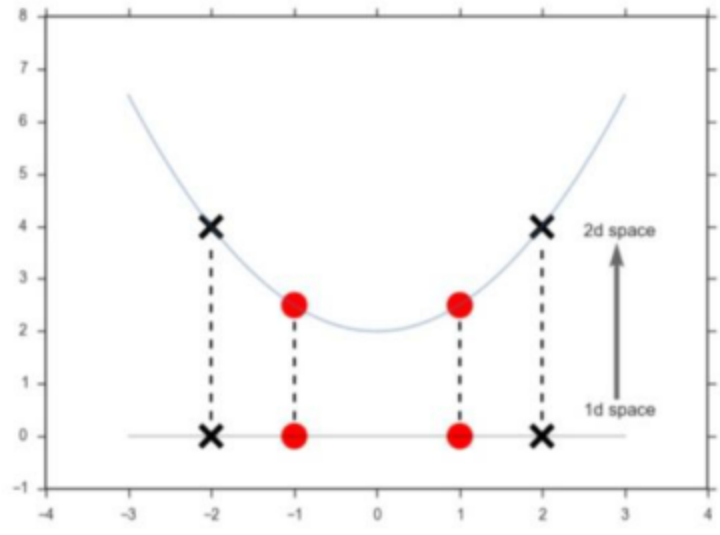
有不等式约束的优化问题 $\min f(x)$ $s.t.\quad g_i(x)\le 0, i=1,2,\cdots,n$ $h_j(x)=0, j=1,2,\cdots,m$

使用KKT条件。我们把所有的等式、不等式约束与f(x)写为一个式子，通过一些条件，可以求出最优值的必要条件，这个条件称为KKT条件。

支持向量机——非线性SVM



支持向量机——非线性SVM



支持向量机——非线性SVM

$$y = \omega^T x + \gamma = \sum_{i=1}^{n} \alpha_i y_i x_i^T x + \gamma \quad \Longrightarrow \quad y = \omega^T \phi(x) + \gamma = \sum_{i=1}^{n} \alpha_i y_i \phi(x_i)^T \phi(x_j) + \gamma$$

$$\mathrm{K}(x_i, x_j) = \left\langle \phi(x_i), \phi(x_j) \right\rangle = \phi(x_i)^T \phi(x_j)$$

即，x_i和x_j在高维空间的内积等于它们在原始样本空间中通过函数$\mathrm{K}(x_i, x_j)$计算的函数值。这里的$\mathrm{K}(x_i, x_j)$就是核函数。

$$y = \sum_{i=1}^{n} \alpha_i y_i \mathrm{K}(x_i, x_j) + \gamma$$

支持向量机——非线性SVM

令 $x=[x_1,x_2,x_3]^T, y=[y_1,y_2,y_3]^T$，我们定义

$\phi(x)=[x_1x_1,x_1x_2,x_1x_3,x_2x_1,x_2x_2,x_2x_3,x_3x_1,x_3x_2,x_3x_3]$ 将原始数据从三维空间映射到九维空间中，让我们来计算 $\phi(1,2,3)\cdot\phi(4,5,6)$：

$$
\begin{aligned}
\phi(1,2,3) &= [1,2,3,2,4,6,3,6,9]^T \\
\phi(4,5,6) &= [16,20,24,20,25,30,24,30,36]^T \\
\phi(1,2,3)\cdot\phi(4,5,6) &= 1\times16+2\times20+3\times24+2\times20+4\times25+6\times30+3\times24+6\times30+9\times36 \\
&= 16+40+72+40+100+180+72+180+324 \\
&= 1024
\end{aligned}
$$

支持向量机——非线性SVM

$$
\begin{aligned}
\phi(x)\cdot\phi(y) &= [x_1x_1,x_1x_2,x_1x_3,x_2x_1,x_2x_2,x_2x_3,x_3x_1,x_3x_2,x_3x_3]^T\cdot[y_1y_1,y_1y_2,y_1y_3,y_2y_1,y_2y_2,y_2y_3,y_3y_1,y_3y_2,y_3y_3] \\
&= x_1y_1x_1y_1+x_1y_1x_2y_2+x_1y_1x_3y_3+x_2y_2x_1y_1+x_2y_2x_2y_2+x_2y_2x_3y_3 \\
&+ x_3y_3x_1y_1+x_3y_3x_2y_2+x_3y_3x_3y_3 \\
&= (x_1y_1+x_2y_2+x_3y_3)^2 \\
&= (x^Ty)^2 \\
&= K(x,y)
\end{aligned}
$$

$$K(x,y)=K((1,2,3),(4,5,6))=(1\times4+2\times5+3\times6)^2=(32)^2=1024$$

相比于从低维映射到高维空间再进行矢量积运算，核函数大大简化了计算的过程，使得向更高维转化变为了可能，我们不需要知道低维空间的数据是怎样映射到高维空间的，我们只需要知道结果是怎么计算出来的。

支持向量机——非线性SVM

在实际应用中，通常人们会从一些常用的核函数里选择（根据样本数据的不同，选择不同的参数，实际上就得到了不同的核函数）。下面给出常用的核函数：

名称	表达式	参数
线性核	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j$	即处理线性可分的情形,这样使得它们在形式统一起来
多项式核	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j)^d$	$d \geqslant 1$ 为多项式的次数
高斯核RBF	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right)$	$\sigma > 0$ 为高斯核的带宽(width)
拉普拉斯核	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{\sigma}\right)$	$\sigma > 0$
Sigmoid 核	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tanh(\beta \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j + \theta)$	tanh 为双曲正切函数, $\beta > 0, \theta < 0$

参考资料

1、斯坦福大学吴恩达机器学习系列课程

https://www.bilibili.com/video/BV164411b7dx?from=search&seid=12860939662404817484

2、台湾大学李宏毅机器学习课程

https://www.bilibili.com/video/BV13x411v7US?from=search&seid=12860939662404817484

3、3BluelBrown

https://space.bilibili.com/88461692?spm_id_from=333.788.b_765f7570696e666f.1