回归

七月算法 邹博

2015年10月25日

引理：向量的导数

- A为m×n的矩阵， x为n×1的列向量，
- 记 $\vec{y} = A \cdot \vec{x}$
- 思考： $\dfrac{\partial \vec{y}}{\partial \vec{x}} = ?$

向量导数的推导

- 令

$$A=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \quad \vec{x}=\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \quad A\cdot\vec{x}=\begin{pmatrix} a_{11}x_1+a_{12}x_2+\cdots+a_{1n}x_n \\ a_{21}x_1+a_{22}x_2+\cdots+a_{2n}x_n \\ \vdots \\ a_{m1}x_1+a_{m2}x_2+\cdots+a_{mn}x_n \end{pmatrix}$$

- 从而，

$$\frac{\partial\vec{y}}{\partial\vec{x}}=\frac{\partial A\vec{x}}{\partial\vec{x}}=\begin{bmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{bmatrix}=A^T$$

结论与直接推广

- 向量偏导公式：

$$\frac{\partial A\vec{x}}{\partial \vec{x}} = A^T$$

$$\frac{\partial A^T\vec{x}}{\partial \vec{x}} = A$$

$$\frac{\partial A\vec{x}}{\partial \vec{x}^T} = A$$

引理：标量对向量的导数

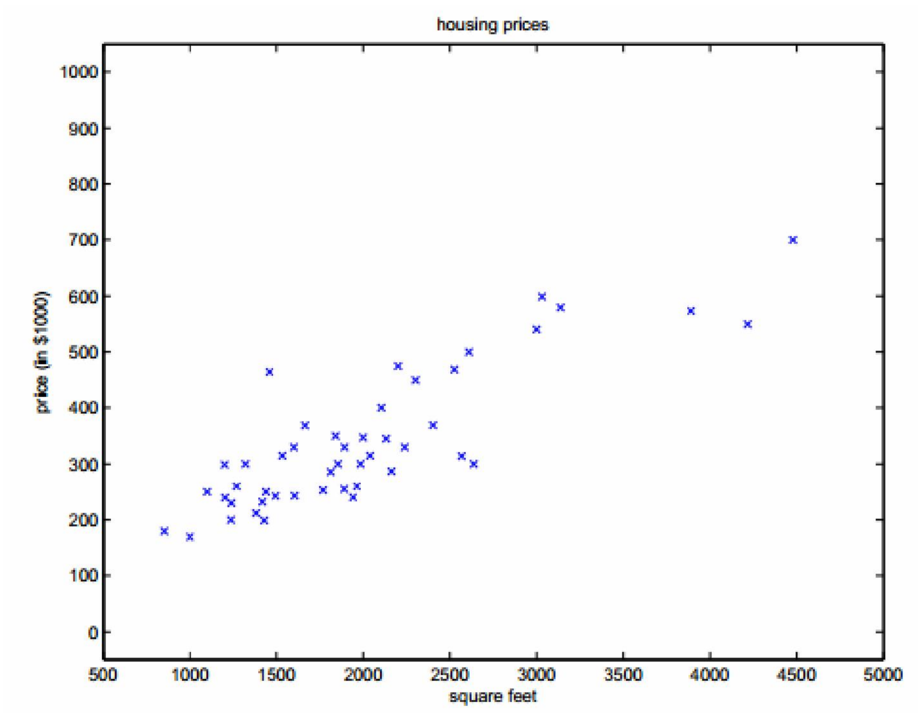
- A为n×n的矩阵， x为n×1的列向量，
- 记 $y = \vec{x}^T \cdot A \cdot \vec{x}$
- 同理可得：

$$\frac{\partial y}{\partial \vec{x}} = \frac{\partial\left(\vec{x}^T \cdot A \cdot \vec{x}\right)}{\partial \vec{x}} = \left(A^T + A\right) \cdot \vec{x}$$

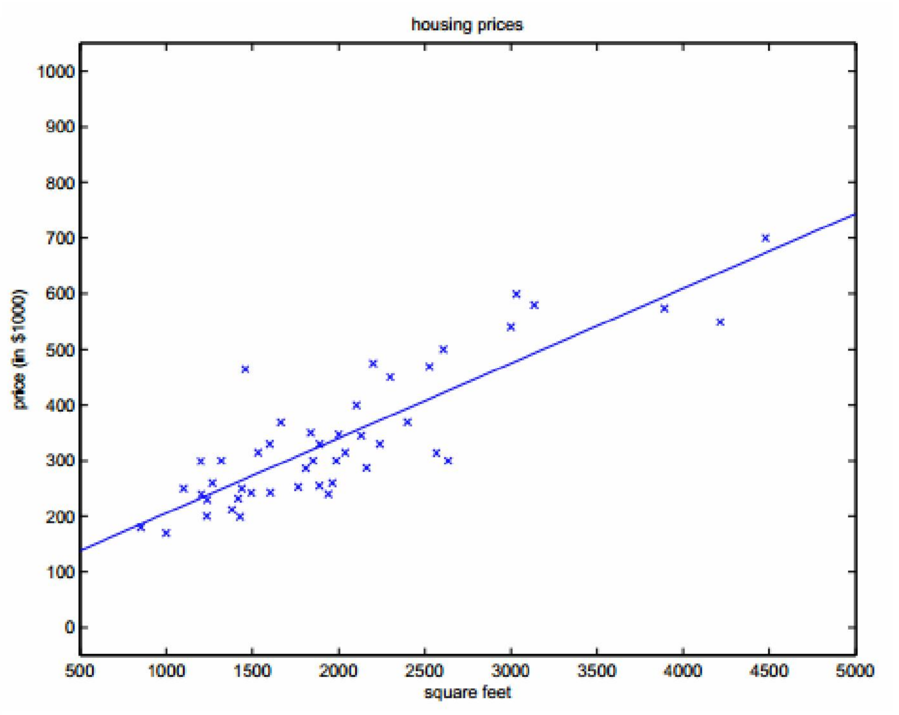
- 若A为对称阵，则有 $\dfrac{\partial\left(\vec{x}^T A \vec{x}\right)}{\partial \vec{x}} = 2A\vec{x}$

线性回归

- y=ax+b



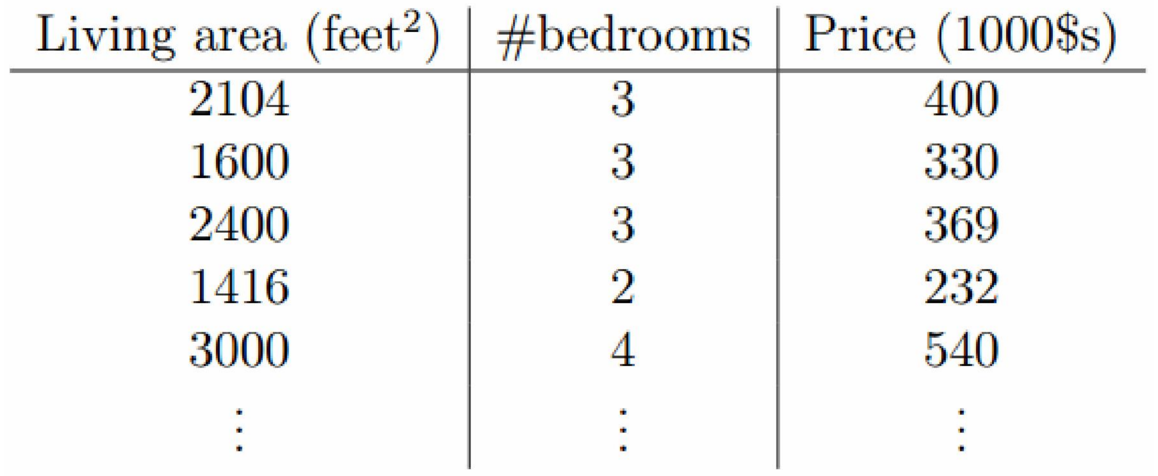
housing prices
price (in $1000)
square feet



housing prices
price (in $1000)
square feet

多个变量的情形

- 考虑两个变量

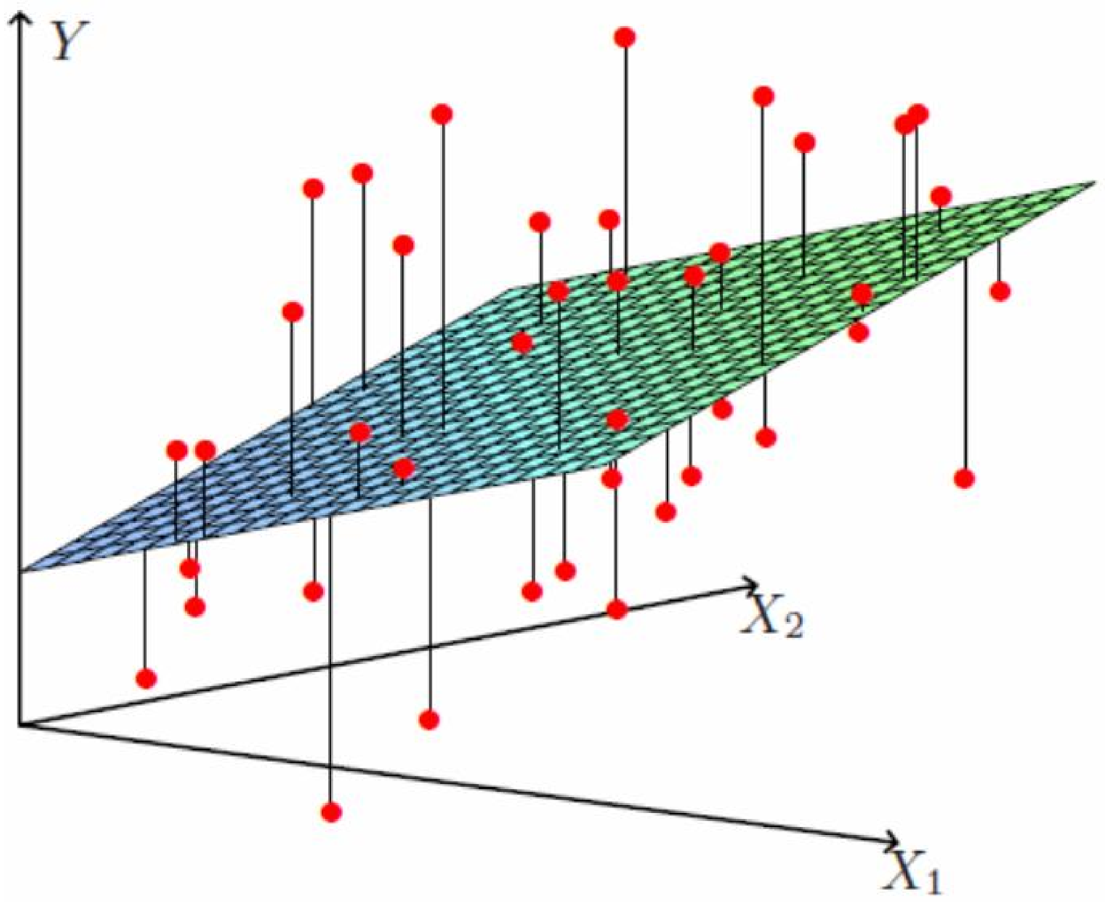


Living area (feet^2)	#bedrooms	Price (1000\$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮

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

$$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x$$

多个变量的情形



最小二乘的目标函数

- m为样本个数，则一个比较“符合常理”的误差函数为：

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

- 符合常理
  - 最小二乘建立的目标函数，即是在噪声为均值为0的高斯分布下，极大似然估计的目标函数

使用极大似然估计解释最小二乘

$$y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)}$$

the $\epsilon^{(i)}$ are distributed IID (independently and identically distributed) according to a Gaussian distribution (also called a Normal distribution) with mean zero and some variance σ^2

似然函数

$$p(\epsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\epsilon^{(i)})^2}{2\sigma^2}\right)$$

$$p(y^{(i)}|x^{(i)};\theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$$

$$\begin{aligned} L(\theta) &= \prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)};\theta) \\ &= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \end{aligned}$$

对数似然

$$
\begin{aligned}
\ell(\theta) &= \log L(\theta) \\
&= \log \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)}-\theta^T x^{(i)})^2}{2\sigma^2}\right) \\
&= \sum_{i=1}^{m} \log \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)}-\theta^T x^{(i)})^2}{2\sigma^2}\right) \\
&= m \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2} \cdot \frac{1}{2} \sum_{i=1}^{m} (y^{(i)}-\theta^T x^{(i)})^2
\end{aligned}
$$

θ 的解析式的求解过程

- 将M个N维样本组成矩阵X：
  - 其中，X的每一行对应一个样本，共M个样本
  - X的每一列对应样本的一个维度，共N维
    - 其实还有额外的一维常数项，全为1
- 目标函数 $J(\theta)=\frac{1}{2}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right)-y^{(i)}\right)^2=\frac{1}{2}(X\theta-y)^T(X\theta-y)$
- 梯度：

$$\nabla_\theta J(\theta)=\nabla_\theta\left(\frac{1}{2}(X\theta-y)^T(X\theta-y)\right)=\nabla_\theta\left(\frac{1}{2}\left(\theta^T X^T-y^T\right)(X\theta-y)\right)$$

$$=\nabla_\theta\left(\frac{1}{2}\left(\theta^T X^T X\theta-\theta^T X^T y-y^T X\theta+y^T y\right)\right)$$

$$=\frac{1}{2}\left(2X^T X\theta-X^T y-\left(y^T X\right)^T\right)=X^T X\theta-X^T y\xrightarrow{\text{求驻点}}0$$

最小二乘意义下的参数最优解

- 参数的解析式

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

- 若$X^T X$不可逆或防止过拟合，增加λ扰动

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

- “简便”方法记忆结论

$$X\theta = y \Rightarrow X^T X\theta = X^T y$$

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

加入λ扰动后

- X^TX半正定：对于任意的非零向量u

$$uX^TXu=(Xu)^TXu\xrightarrow{令v=Xu}v^Tv\geq 0$$

- 所以，对于任意的实数λ>0，$X^TX+\lambda I$ 正定，从而可逆。保证回归公式一定有意义。

$$\theta=\left(X^TX+\lambda I\right)^{-1}X^Ty$$

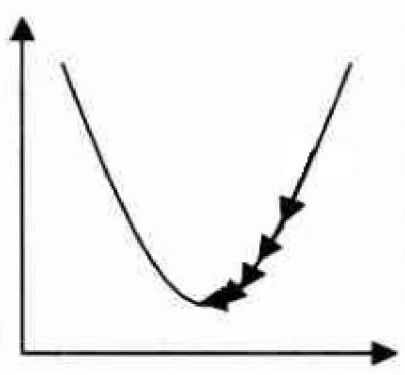
若X^TX阶过高

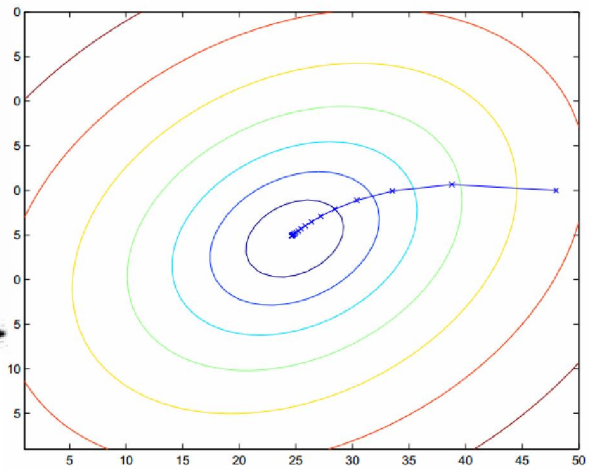
- 实际中，若X^TX阶过高，仍然需要使用梯度下降的方式计算数值解。

广义逆矩阵（伪逆）$A^{+}=\left(A^{T}A\right)^{-1}A^{T}$

- 若A为非奇异矩阵,则线性方程组Ax=b的解为$x=A^{-1}b$ 其中A的A的逆矩阵A^{-1}满足$A^{-1}A=AA^{-1}=I$ (I为单位矩阵)。若A是奇异阵或长方阵, x=A^{+}b。A^{+}叫做A的伪逆阵。
- 1955年R.Penrose证明了对每个m×n阶矩阵A,都存在惟一的n×m阶矩阵X，满足：①AXA=A；②XAX=X；③(AX)* = I；④(XA)* = I。通常称X为A的穆尔-彭罗斯广义逆矩阵,简称M-P逆，记作A^{+}。
- 在矛盾线性方程组Ax = b的最小二乘解中, x=A^{+}b是范数最小的一个解。
  - 在奇异值分解SVD的问题中，将继续该话题的讨论。

梯度下降算法





- 初始化θ(随机初始化)
- 迭代，新的θ能够使得J(θ)更小
- 如果J(θ)能够继续减少，返回(2)

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

- α称为学习率/步长

梯度方向

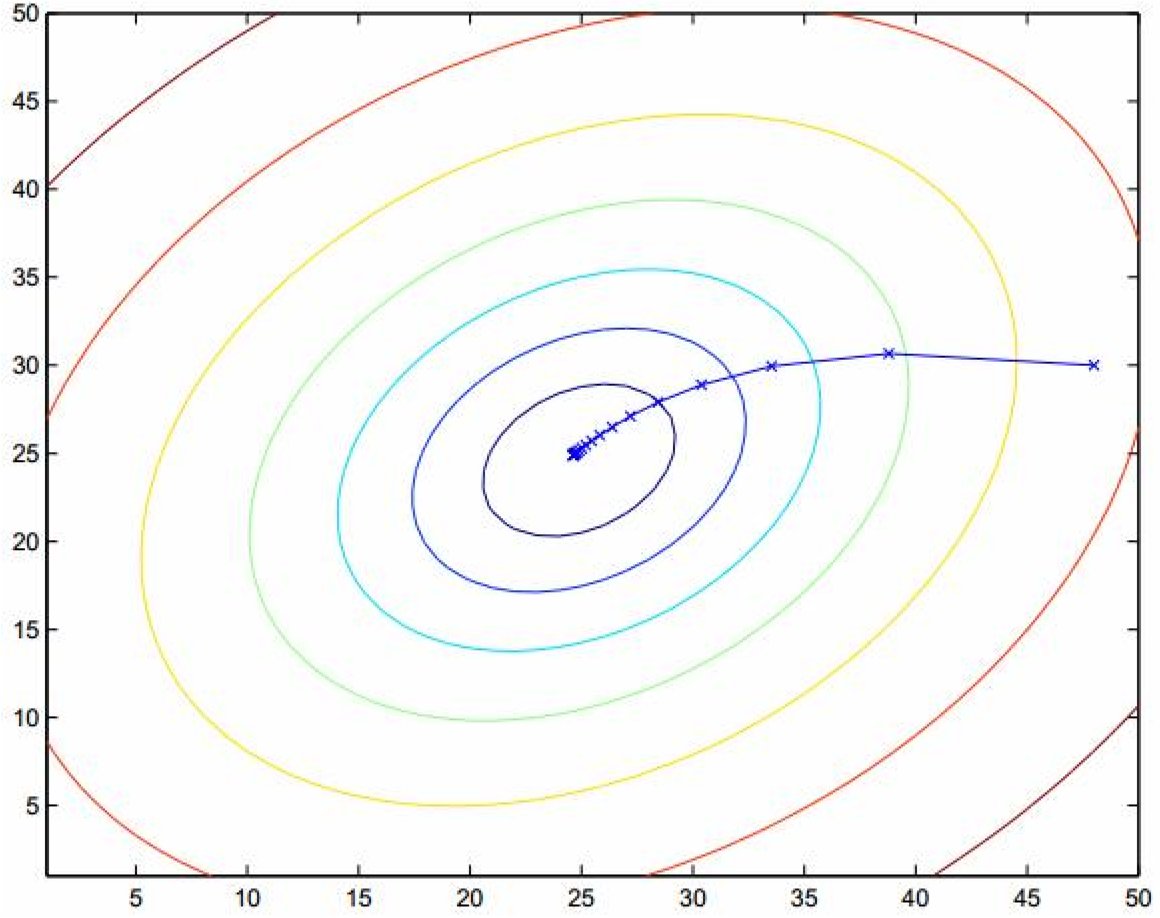
$$\begin{aligned}
\frac{\partial}{\partial\theta_j}J(\theta) &= \frac{\partial}{\partial\theta_j}\frac{1}{2}\left(h_\theta(x)-y\right)^2 \\
&= 2\cdot\frac{1}{2}\left(h_\theta(x)-y\right)\cdot\frac{\partial}{\partial\theta_j}\left(h_\theta(x)-y\right) \\
&= \left(h_\theta(x)-y\right)\cdot\frac{\partial}{\partial\theta_j}\left(\sum_{i=0}^{n}\theta_i x_i - y\right) \\
&= \left(h_\theta(x)-y\right)x_j
\end{aligned}$$

批处理梯度下降算法

```
Repeat until convergence {
    $\theta_j := \theta_j + \alpha \sum_{i=1}^{m} \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}$
}
```

gradient descent. Note that, while gradient descent can be susceptible to local minima in general, the optimization problem we have posed here for linear regression has only one global, and no other local, optima; thus gradient descent always $\underline{\text{converges}}$ (assuming the learning rate α is not too large) $\underline{\text{to the global minimum.}}$ Indeed, J is a $\boxed{\text{convex}}$ quadratic function.

批处理梯度下降图示



随机梯度下降算法

```
Loop {
    for i=1 to m, {
        $\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}$
    }
}
```

This algorithm is called **stochastic gradient descent** (also **incremental gradient descent**). Whereas batch gradient descent has to scan through the entire training set before taking a single step—a costly operation if m is large—stochastic gradient descent can start making progress right away, and continues to make progress with each example it looks at. Often, stochastic gradient descent gets θ "close" to the minimum much faster than batch gradient descent. (Note however that it may never "converge" to the minimum, and the parameters θ will keep oscillating around the minimum of $J(\theta)$; but in practice most of the values near the minimum will be reasonably good approximations to the true minimum.[2]) For these reasons, particularly when the training set is large, stochastic gradient descent is often preferred over batch gradient descent.

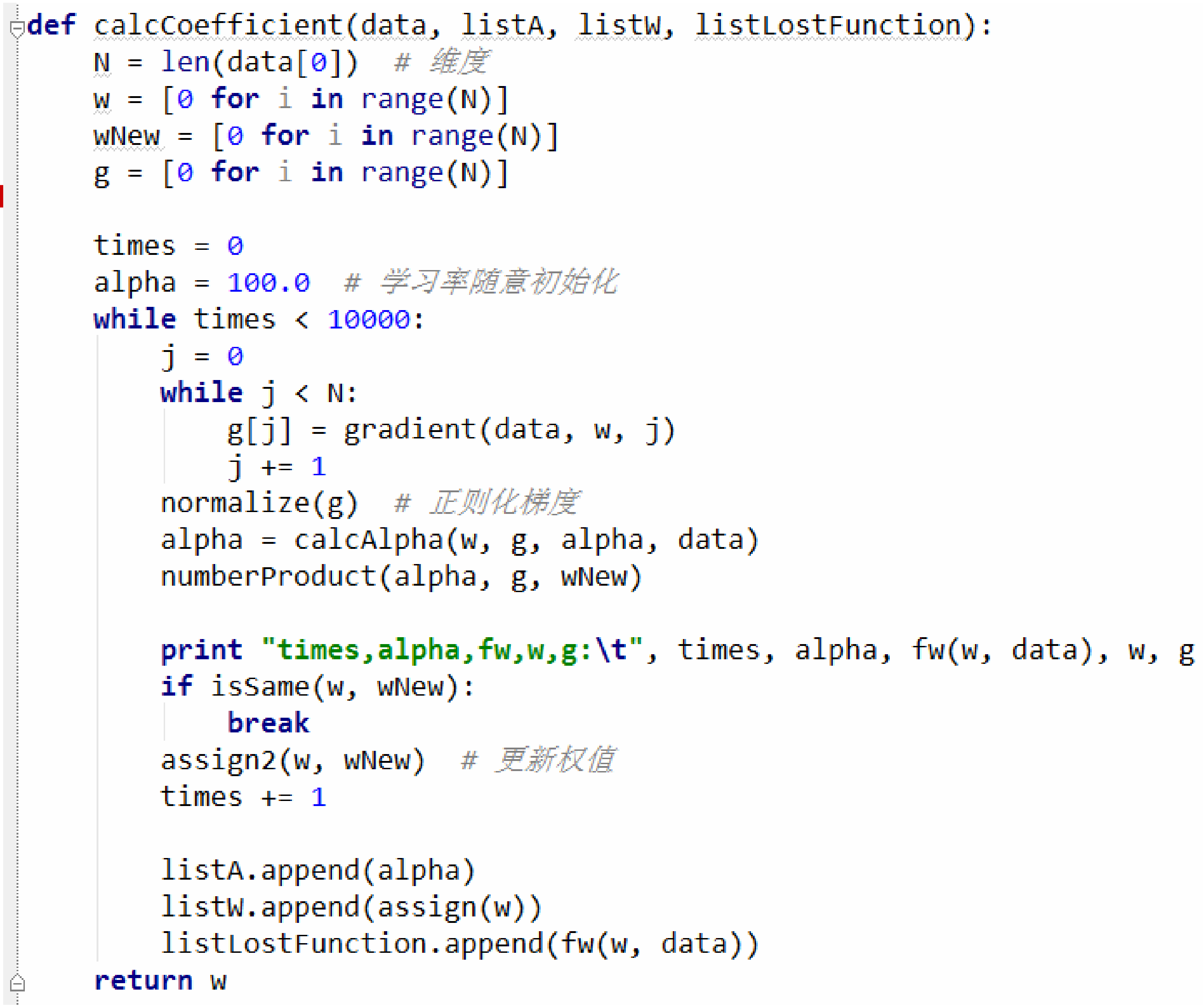
mini-batch

- 如果不是每拿到一个样本即更改梯度，而是若干个样本的平均梯度作为更新方向，则是mini-batch梯度下降算法。

```
Repeat until convergence {
    $\theta_j := \theta_j + \alpha \sum_{i=1}^{m} \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}$
}
```

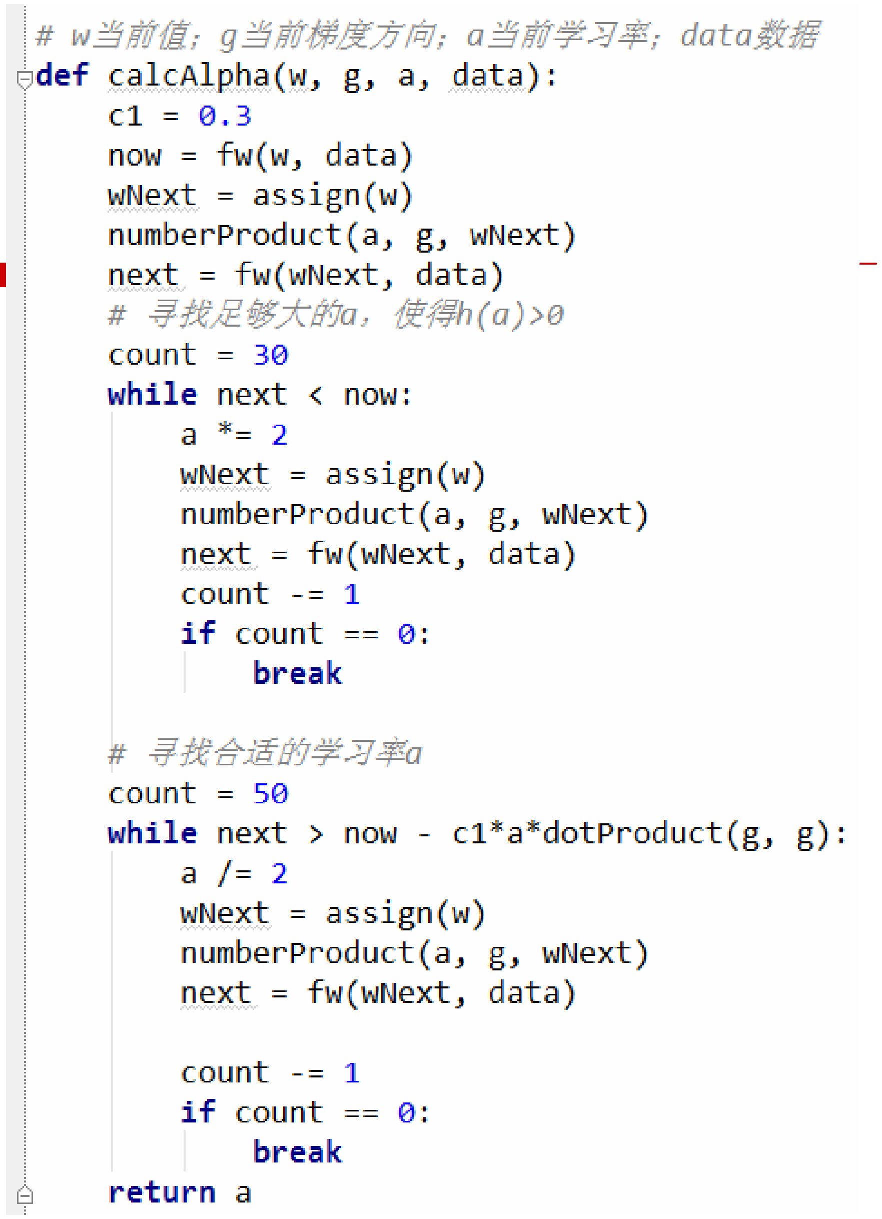
```
Loop {
    for i=1 to m, {
        $\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}$
    }
}
```

回归Code



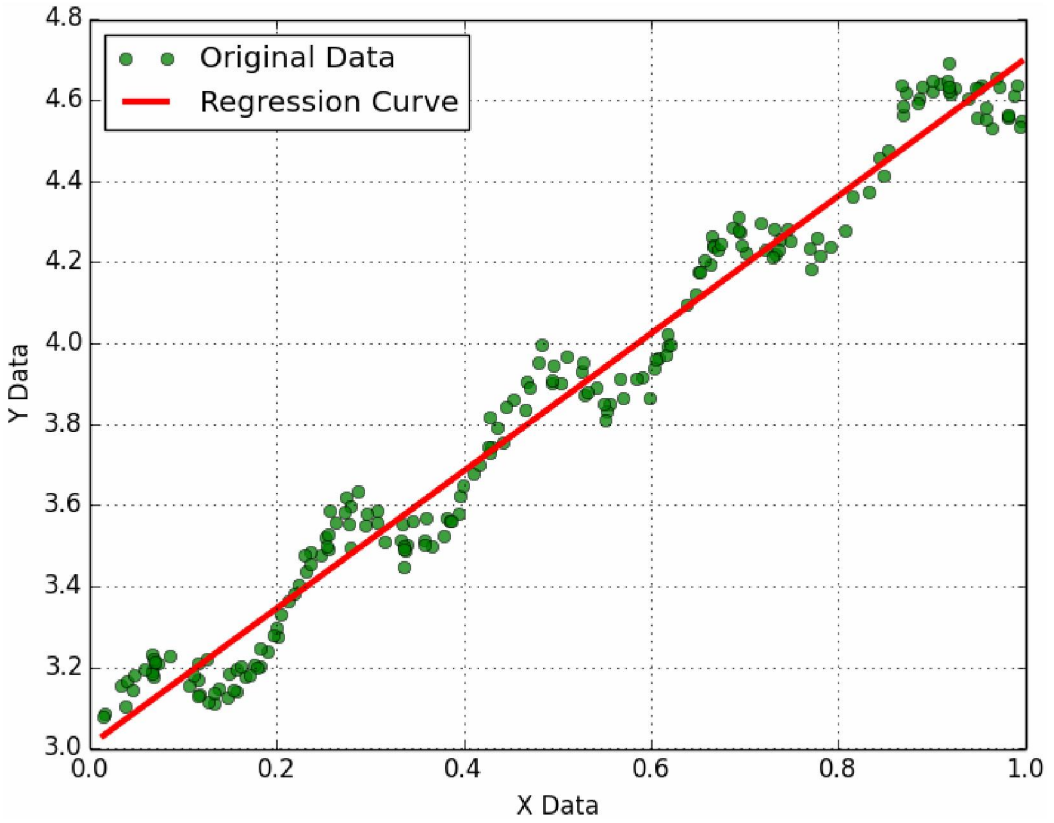
```python
def calcCoefficient(data, listA, listW, listLostFunction):
    N = len(data[0])  # 维度
    w = [0 for i in range(N)]
    wNew = [0 for i in range(N)]
    g = [0 for i in range(N)]

    times = 0
    alpha = 100.0  # 学习率随意初始化
    while times < 10000:
        j = 0
        while j < N:
            g[j] = gradient(data, w, j)
            j += 1
        normalize(g)  # 正则化梯度
        alpha = calcAlpha(w, g, alpha, data)
        numberProduct(alpha, g, wNew)

        print "times,alpha,fw,w,g:\t", times, alpha, fw(w, data), w, g
        if isSame(w, wNew):
            break
        assign2(w, wNew)  # 更新权值
        times += 1

        listA.append(alpha)
        listW.append(assign(w))
        listLostFunction.append(fw(w, data))
    return w
```

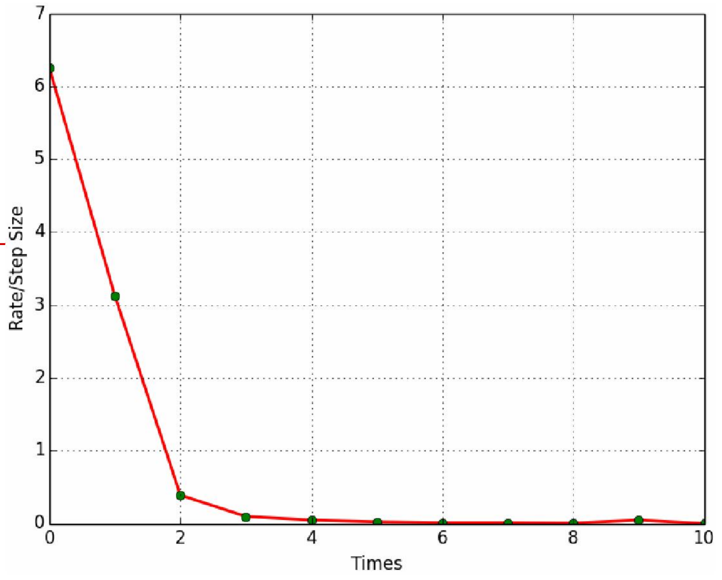
附：学习率Code

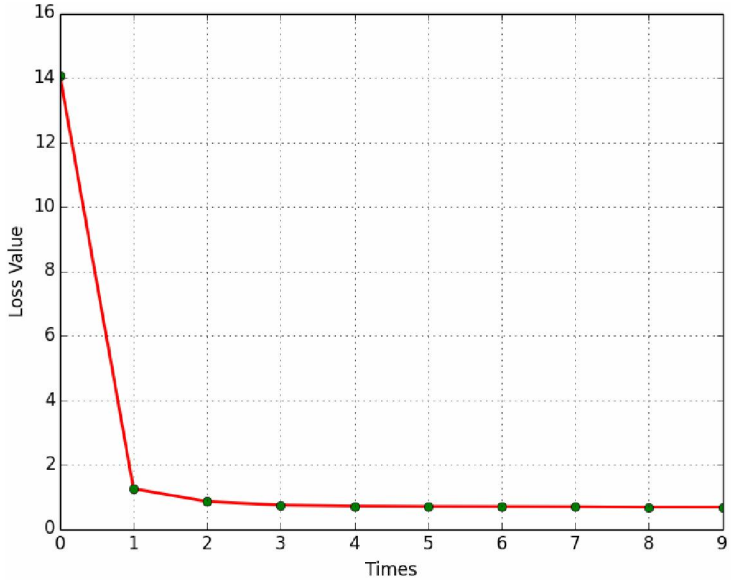


```python
# w当前值；g当前梯度方向；a当前学习率；data数据
def calcAlpha(w, g, a, data):
    c1 = 0.3
    now = fw(w, data)
    wNext = assign(w)
    numberProduct(a, g, wNext)
    next = fw(wNext, data)
    # 寻找足够大的a，使得h(a)>0
    count = 30
    while next < now:
        a *= 2
        wNext = assign(w)
        numberProduct(a, g, wNext)
        next = fw(wNext, data)
        count -= 1
        if count == 0:
            break

    # 寻找合适的学习率a
    count = 50
    while next > now - c1*a*dotProduct(g, g):
        a /= 2
        wNext = assign(w)
        numberProduct(a, g, wNext)
        next = fw(wNext, data)

        count -= 1
        if count == 0:
            break
    return a
```

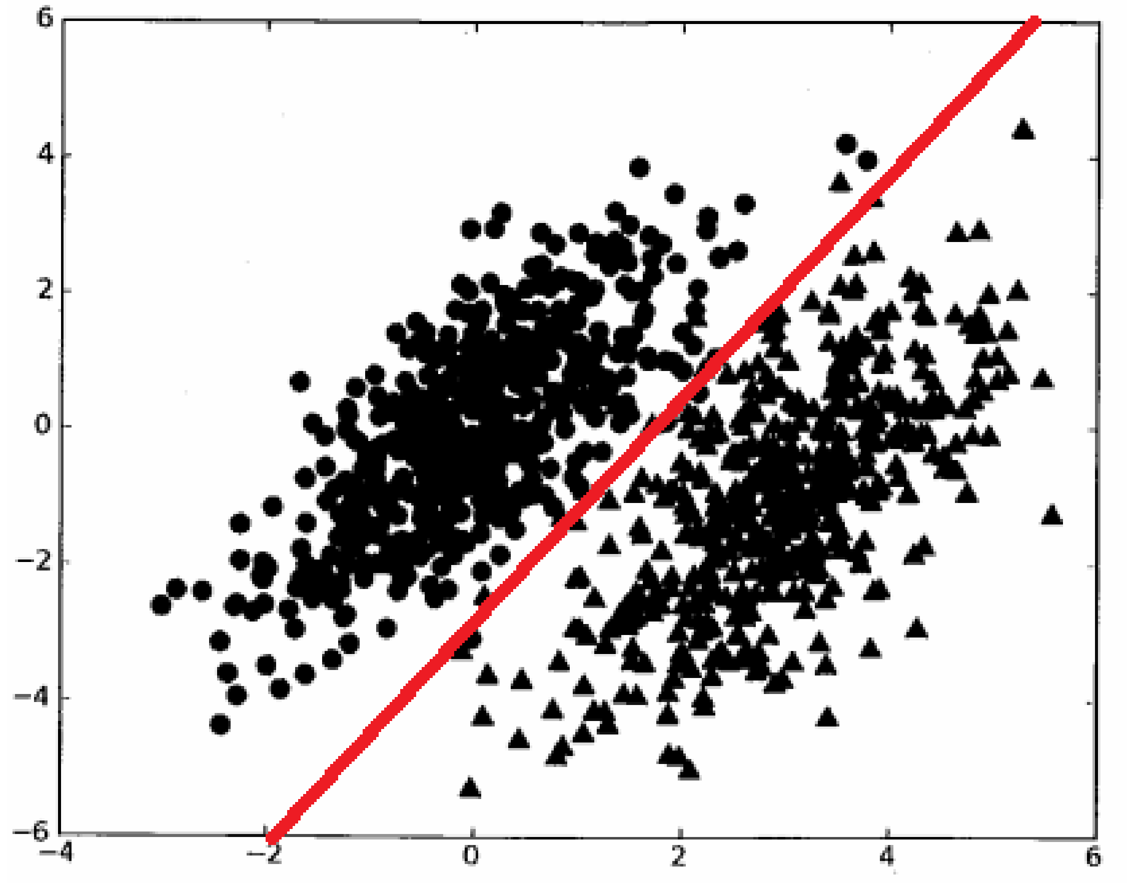
线性回归、rate、Loss



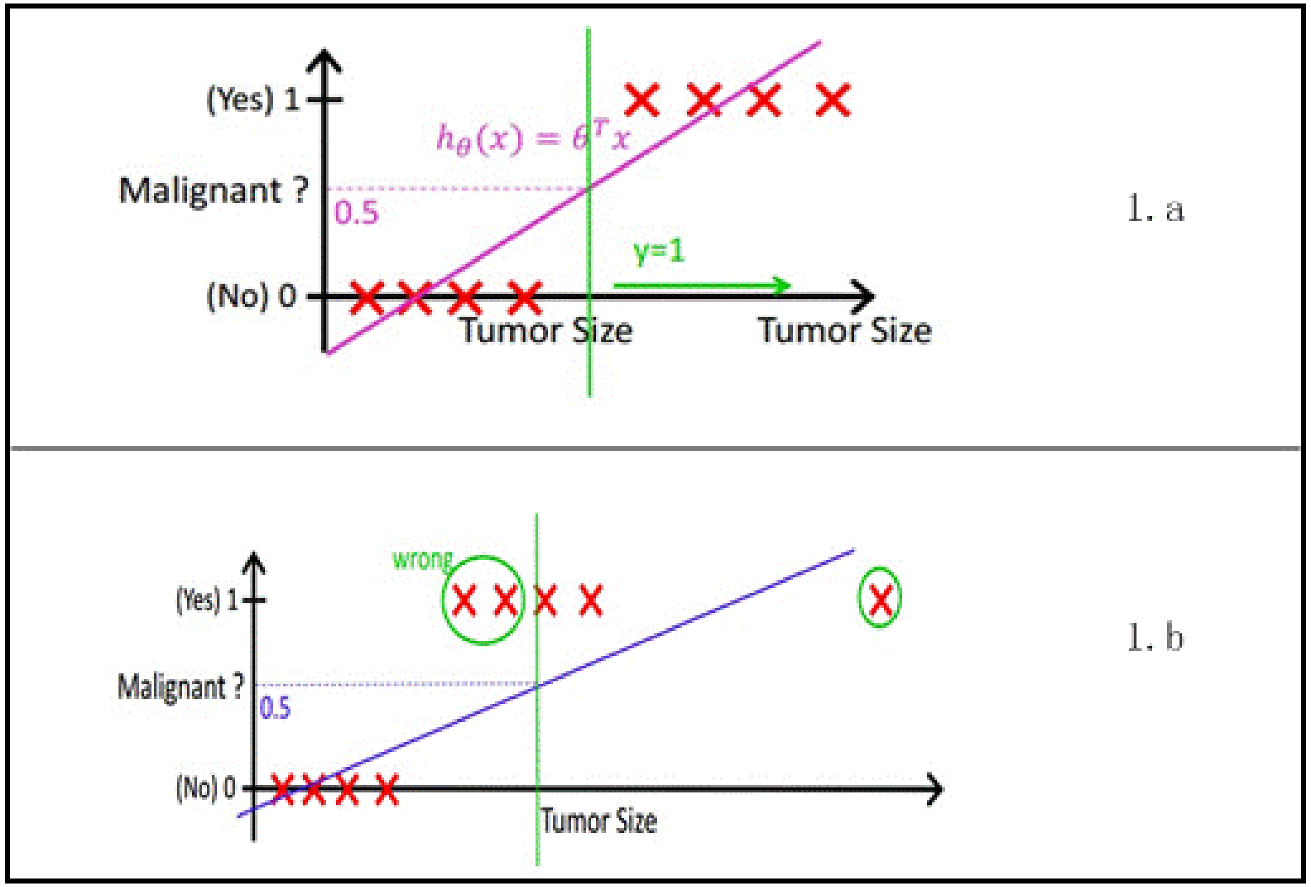




用回归解决分类问题，如何？



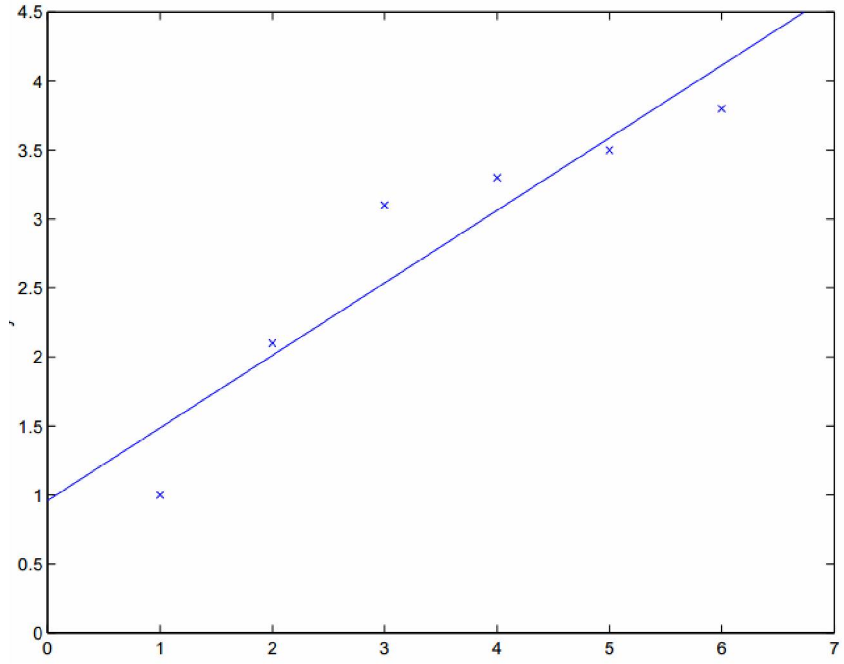
最简单的例子：一维回归

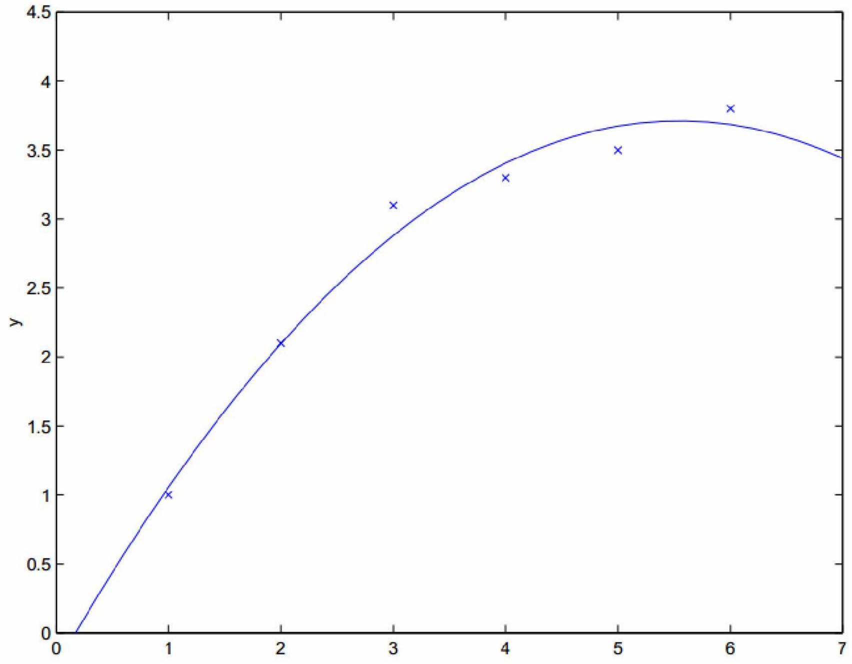


(Yes) 1
Malignant ?
(No) 0
$h_\theta(x) = \theta^T x$
0.5
y=1
Tumor Size
Tumor Size
1. a
wrong
(Yes) 1
Malignant ?
0.5
(No) 0
Tumor Size
1. b

线性回归的进一步分析

- 可以对样本是非线性的，只要对参数θ线性

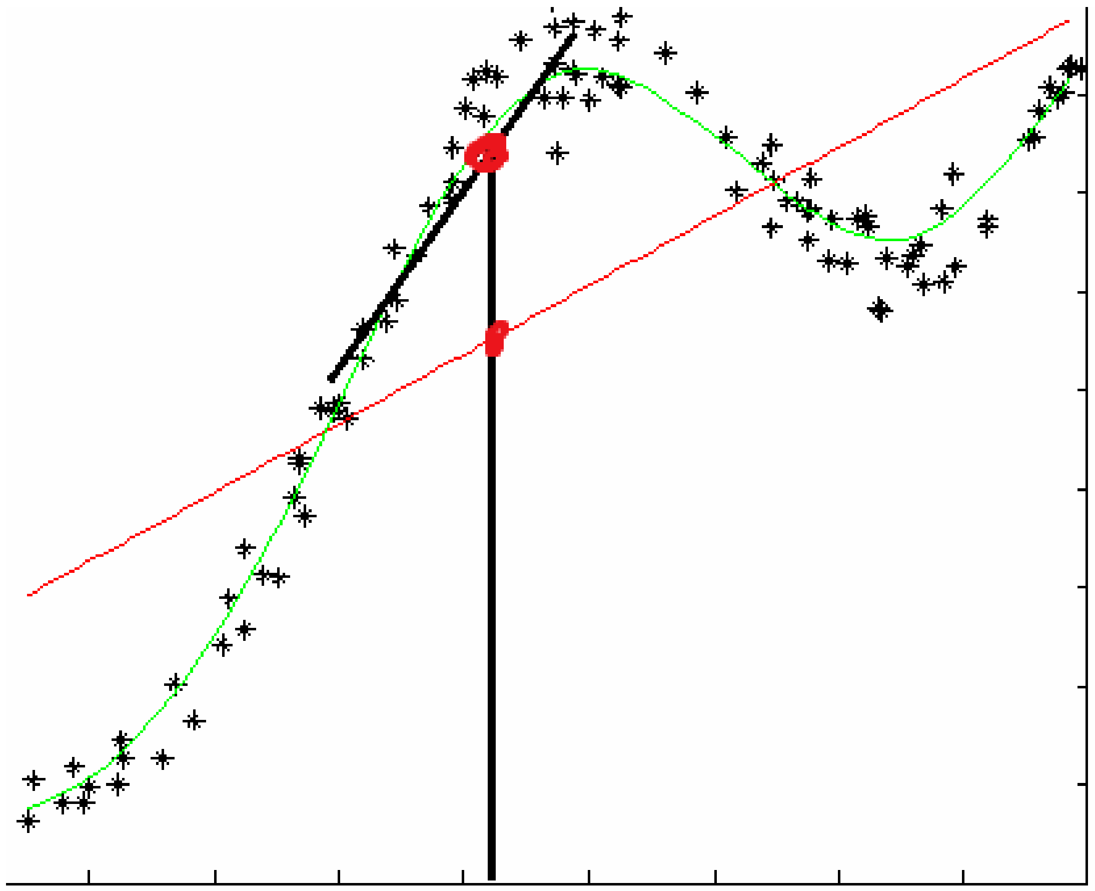




$$y = \theta_0 + \theta_1 x + \theta_2 x^2$$

局部加权回归

- 黑色是样本点
- 红色是线性回归曲线
- 绿色是局部加权回归曲线



局部加权线性回归

- LWR：Locally Weighted linear Regression

1. Fit θ to minimize $\sum_i (y^{(i)} - \theta^T x^{(i)})^2$
2. Output $\theta^T x$.

1. Fit θ to minimize $\sum_i w^{(i)} (y^{(i)} - \theta^T x^{(i)})^2$
2. Output $\theta^T x$.

权值的设置

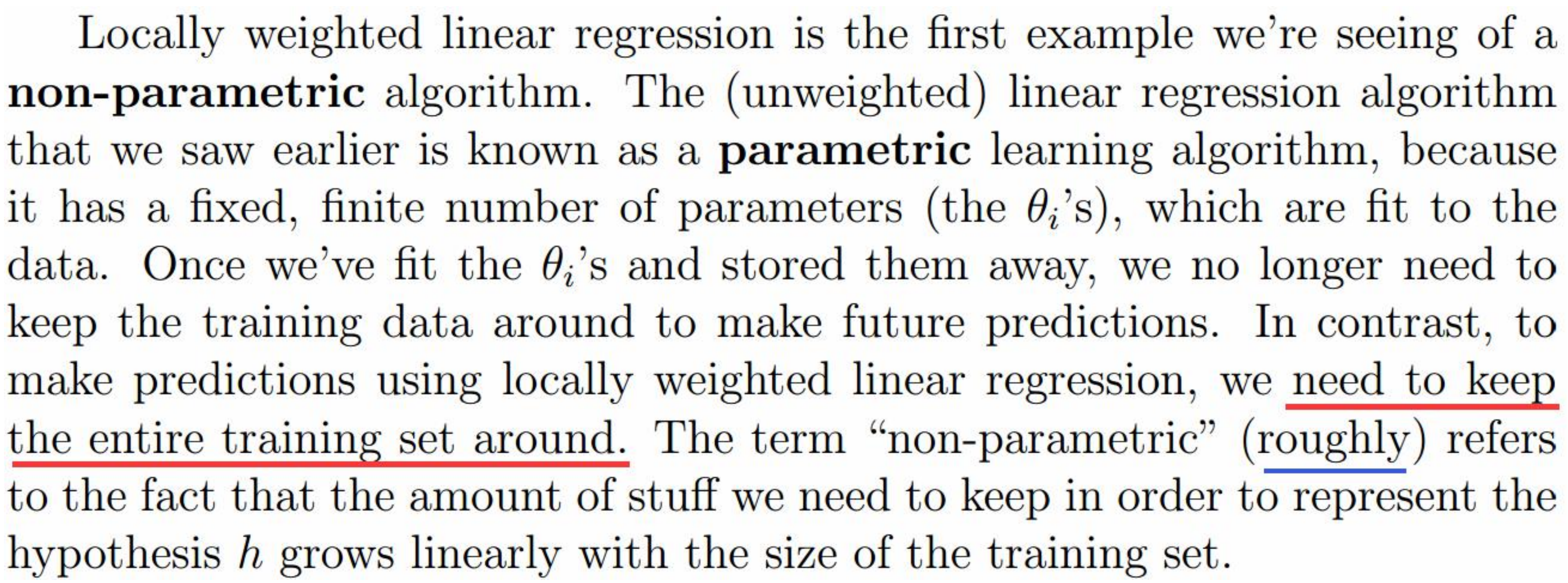
- ω 的一种可能的选择方式(高斯核函数):

$$w^{(i)} = \exp\left(-\frac{(x^{(i)} - x)^2}{2\tau^2}\right)$$

- τ 称为带宽，它控制着训练样本随着与$x^{(i)}$距离的衰减速率。
- 多项式核函数

$$\kappa(\boldsymbol{x}_1, \boldsymbol{x}_2) = (\langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle + R)^d$$

参数算法 – 非参数学习算法

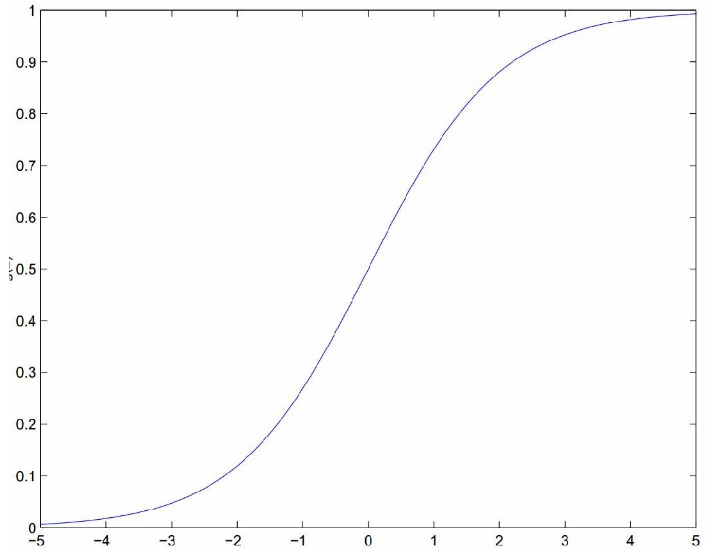


Locally weighted linear regression is the first example we're seeing of a **non-parametric** algorithm. The (unweighted) linear regression algorithm that we saw earlier is known as a **parametric** learning algorithm, because it has a fixed, finite number of parameters (the θ_i's), which are fit to the data. Once we've fit the θ_i's and stored them away, we no longer need to keep the training data around to make future predictions. In contrast, to make predictions using locally weighted linear regression, we need to keep the entire training set around. The term "non-parametric" (roughly) refers to the fact that the amount of stuff we need to keep in order to represent the hypothesis h grows linearly with the size of the training set.

Logistic回归

- Logistic函数

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



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

Logistic函数的导数

$$\begin{aligned}
g'(z) &= \frac{d}{dz}\frac{1}{1+e^{-z}} \\
&= \frac{1}{(1+e^{-z})^2}\left(e^{-z}\right) \\
&= \frac{1}{(1+e^{-z})}\cdot\left(1-\frac{1}{(1+e^{-z})}\right) \\
&= g(z)(1-g(z))
\end{aligned}$$

Logistic回归参数估计

- 假定：
$$
\begin{aligned}
P(y=1 \mid x;\theta) &= h_\theta(x) \\
P(y=0 \mid x;\theta) &= 1-h_\theta(x)
\end{aligned}
$$

$$p(y \mid x;\theta) = \left(h_\theta(x)\right)^{y}\left(1-h_\theta(x)\right)^{1-y}$$

$$
\begin{aligned}
L(\theta) &= p(\vec{y} \mid X;\theta) \\
&= \prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)};\theta) \\
&= \prod_{i=1}^{m} \left(h_\theta(x^{(i)})\right)^{y^{(i)}}\left(1-h_\theta(x^{(i)})\right)^{1-y^{(i)}}
\end{aligned}
$$

对数似然函数

$$\begin{aligned}\ell(\theta) &= \log L(\theta) \\ &= \sum_{i=1}^{m} y^{(i)} \log h(x^{(i)}) + (1-y^{(i)}) \log(1-h(x^{(i)}))\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial\theta_j}\ell(\theta) &= \left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right)\frac{\partial}{\partial\theta_j} g(\theta^T x) \\ &= \left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right) g(\theta^T x)(1-g(\theta^T x)\frac{\partial}{\partial\theta_j}\theta^T x \\ &= \left(y(1-g(\theta^T x)) - (1-y)g(\theta^T x)\right) x_j \\ &= (y - h_\theta(x))\, x_j\end{aligned}$$

参数的迭代

- Logistic回归参数的学习规则：

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

- 比较上面的结果和线性回归的结论的差别：
  - 它们具有相同的形式！

```
Repeat until convergence {
        $\theta_j := \theta_j + \alpha \sum_{i=1}^{m} \left( y^{(i)} - h_\theta(x^{(i)}) \right) x_j^{(i)}$
}
```

```
Loop {
    for i=1 to m, {
        $\theta_j := \theta_j + \alpha \left( y^{(i)} - h_\theta(x^{(i)}) \right) x_j^{(i)}$
    }
}
```

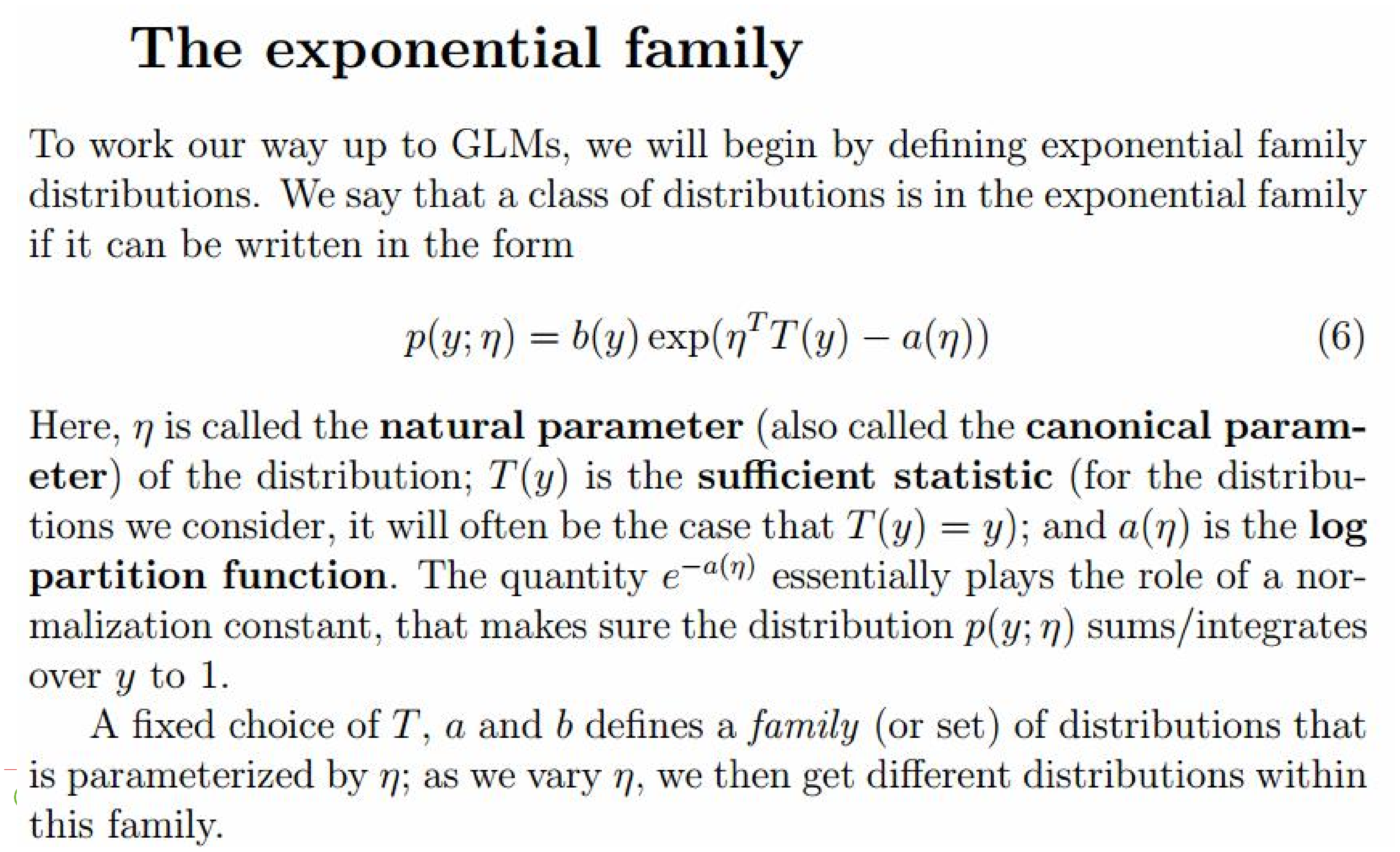
对数线性模型

- 一个事件的几率odds，是指该事件发生的概率与该事件不发生的概率的比值。
- 对数几率：logit函数

$$
\begin{aligned}
P(y=1 \mid x;\theta) &= h_\theta(x) \\
P(y=0 \mid x;\theta) &= 1-h_\theta(x)
\end{aligned}
$$

$$\log it(p) = \log\frac{p}{1-p} = \log\frac{h_\theta(x)}{1-h_\theta(x)} = \log\left(\frac{\dfrac{1}{1+e^{-w^T x}}}{\dfrac{e^{-w^T x}}{1+e^{-w^T x}}}\right) = w^T x$$

复习：指数族



The exponential family

To work our way up to GLMs, we will begin by defining exponential family distributions. We say that a class of distributions is in the exponential family if it can be written in the form

$$p(y;\eta) = b(y)\exp(\eta^T T(y) - a(\eta)) \tag{6}$$

Here, η is called the **natural parameter** (also called the **canonical parameter**) of the distribution; $T(y)$ is the **sufficient statistic** (for the distributions we consider, it will often be the case that $T(y) = y$); and $a(\eta)$ is the **log partition function**. The quantity $e^{-a(\eta)}$ essentially plays the role of a normalization constant, that makes sure the distribution $p(y;\eta)$ sums/integrates over y to 1.

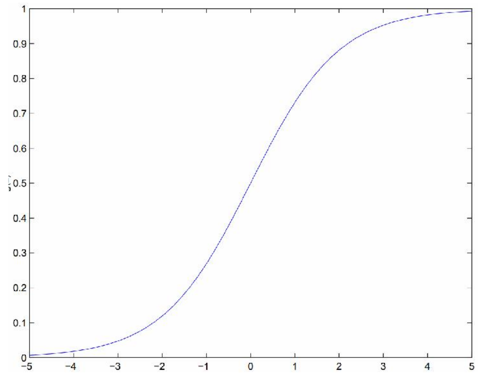
A fixed choice of T, a and b defines a *family* (or set) of distributions that is parameterized by η; as we vary η, we then get different distributions within this family.

复习：指数族

- 指数族概念的目的，是为了说明广义线性模型Generalized Linear Models
- 凡是符合指数族分布的随机变量，都可以用GLM回归分析

广义线性模型GLM

- 因变量y不再只是正态分布，而是扩大为指数族中的任一分布；
- 解释变量x的线性组合不再直接用于解释因变量y的均值u，而是通过一个联系函数g来解释g(u)；这里，要求g连续单调可导
  - 如Logistic回归中的 $g(z) = \dfrac{1}{1+e^{-z}}$



连接函数

- 连接函数：单调可导
- 恒等：g(u)=u，线性模型即使在正态分布下的恒等连接的广义线性模型，
- 对数：g(u)=ln(u)，因为对数的逆是指数，因此它可以将原本线性关系转变成乘积关系；
- Logit：g(u)=ln(u/1-u)，它的特点为可将预测值控制在0~1之间，对于因变量y为比率时适合使用

其他连接函数

- 如：可以将Logistic函数做拉伸变换，得到新的连接函数

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

线性方程的最小二乘问题

- 原问题 $$\begin{aligned}&\text{minimize} \quad x^T x, \quad x \in \mathbf{R}^n \\ &\text{subject to} \quad Ax = b\end{aligned}$$
- Lagrange函数
$$L(x,v) = x^T x + v^T (Ax - b)$$
- Lagrange对偶函数
$$g(v) = -\frac{1}{4} v^T A A^T v - b^T v$$
  - 对L求x的偏导，带入L，得到g
  - 对g求v的偏导，求g的极大值，作为原问题的最小值

求L的对偶函数 $L(x,\nu)=x^Tx+\nu^T(Ax-b)$

$$\frac{\partial L}{\partial x}=\frac{\partial\left(x^Tx+\nu^T(Ax-b)\right)}{\partial x}=2x+A^T\nu\overset{令}{=}0\Rightarrow x^*=-\frac{1}{2}A^T\nu$$

$$L(x,\nu)=x^Tx+\nu^T(Ax-b)$$

$$=\left(-\frac{1}{2}A^T\nu\right)^T\left(-\frac{1}{2}A^T\nu\right)+\nu^T\left(A\left(-\frac{1}{2}A^T\nu\right)-b\right)$$

$$=\frac{1}{4}\nu^TAA^T\nu-\frac{1}{2}\nu^TAA^T\nu-\nu^Tb$$

$$=-\frac{1}{4}\nu^TAA^T\nu-\nu^Tb$$

$$\overset{\Delta}{=}g(\nu)$$

求对偶函数的极大值 $g(\nu) = -\frac{1}{4}\nu^T AA^T \nu - \nu^T b$

$$\frac{\partial g}{\partial \nu} = \frac{\partial\left(-\frac{1}{4}\nu^T AA^T\nu - \nu^T b\right)}{\partial \nu} = -\frac{1}{2}AA^T\nu - b \overset{令}{=} 0$$

$$\Rightarrow AA^T\nu = -2b$$

$$\Rightarrow A^T AA^T\nu = -2A^T b$$

$$\Rightarrow A^T\nu = -2\left(A^T A\right)^{-1} A^T b$$

$$\Rightarrow -\frac{1}{2}A^T\nu = \left(A^T A\right)^{-1} A^T b$$

$$\Rightarrow x^* = \left(A^T A\right)^{-1} A^T b$$

极小值点 $x^* = \left(A^T A\right)^{-1} A^T b$

- 极小值：$\min\left(x^T x\right)$

$$= \left(\left(A^T A\right)^{-1} A^T b\right)^T \left(\left(A^T A\right)^{-1} A^T b\right)$$

$$= b^T A\left(A^T A\right)^{-1}\left(A^T A\right)^{-1} A^T b$$

$$= b^T A\left(A^T A\right)^{-2} A^T b$$

- 极小值点的结论，和通过线性回归计算得到的结论是完全一致的。
  - 线性回归问题具有强对偶性。

强对偶条件

- 若要对偶函数的最大值即为原问题的最小值，考察需要满足的条件：

$$\begin{aligned}
f_0(x^\star) &= g(\lambda^\star, \nu^\star) \\
&= \inf_x \left(f_0(x) + \sum_{i=1}^{m} \lambda_i^\star f_i(x) + \sum_{i=1}^{p} \nu_i^\star h_i(x) \right) \\
&\le f_0(x^\star) + \sum_{i=1}^{m} \lambda_i^\star f_i(x^\star) + \sum_{i=1}^{p} \nu_i^\star h_i(x^\star) \\
&\le f_0(x^\star).
\end{aligned}$$

Karush-Kuhn-Tucker (KKT)条件

$$\begin{aligned} f_i(x^\star) &\le 0, \quad i = 1, \ldots, m \\ h_i(x^\star) &= 0, \quad i = 1, \ldots, p \\ \lambda_i^\star &\ge 0, \quad i = 1, \ldots, m \\ \lambda_i^\star f_i(x^\star) &= 0, \quad i = 1, \ldots, m \end{aligned}$$

$$\nabla f_0(x^\star) + \sum_{i=1}^m \lambda_i^\star \nabla f_i(x^\star) + \sum_{i=1}^p \nu_i^\star \nabla h_i(x^\star) = 0$$

参考文献

- Prof. Andrew Ng, Machine Learning, Stanford University
- 统计学习方法，李航著，清华大学出版社，2012年
- Convex Optimization, Stephen Boyd,Lieven Vandenberghe, Cambridge University Press, 2004
  - 中译本：王书宁，许鋆，黄晓霖 译，凸优化，清华大学出版社，2013

我们在这里

7 七月算法 http://www.julyedu.com/

- 视频/课程/社区

- 七月题库APP：Android/iOS
  - http://www.julyapp.com/
- 微博
  - @研究者July
  - @七月题库
  - @邹博_机器学习
- 微信公众号
  - julyedu

感谢大家！

恳请大家批评指正！