法律声明

□ 本课件包括演示文稿、示例、代码、题库、视频和声音等内容，小象学院和主讲老师拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意及内容，我们保留一切通过法律手段追究违反者的权利。

□ 课程详情请咨询

- ■ 微信公众号：小象
- ■ 新浪微博：ChinaHadoop

回归

小象学院 ChinaHadoop.cn

邹博

主要内容

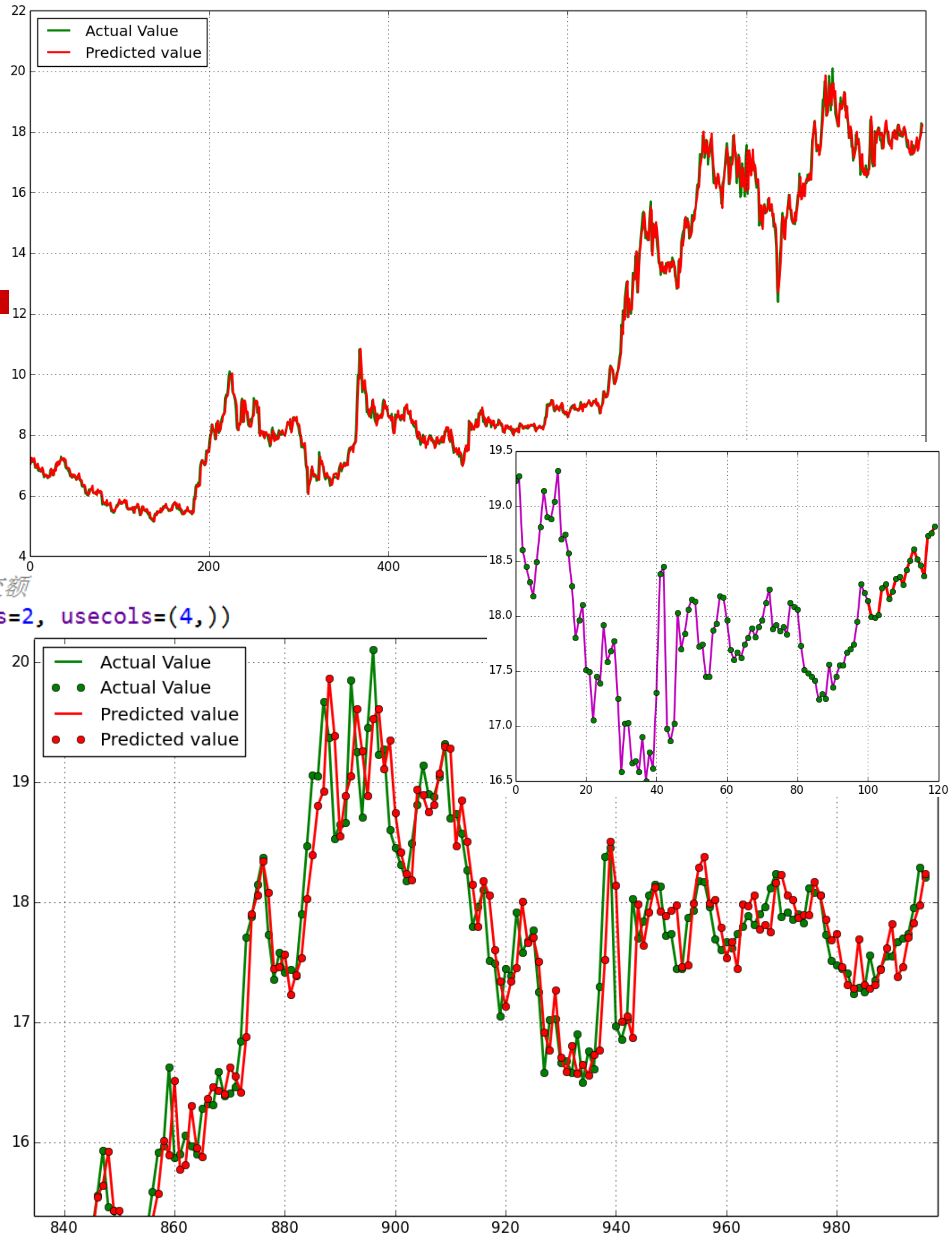
- □ 线性回归
  - ■ 高斯分布
  - ■ 最大似然估计MLE
  - ■ 最小二乘法的本质
- □ Logistic回归
  - ■ 分类问题的首选算法
- □ 工具
  - ■ 梯度下降算法
  - ■ 极大似然估计

股价预测

□ 方法：自回归

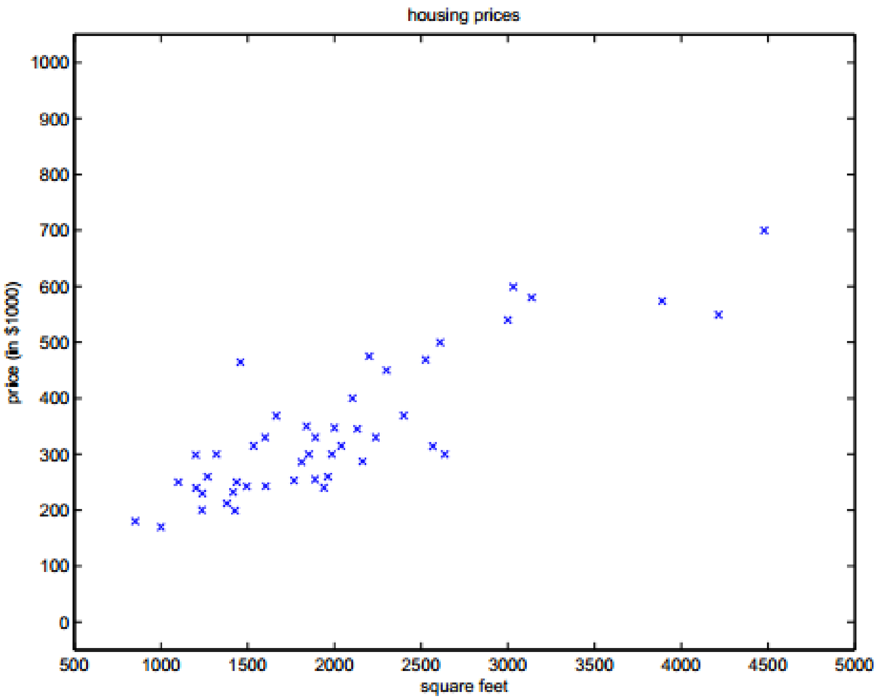
□ 参数：100阶

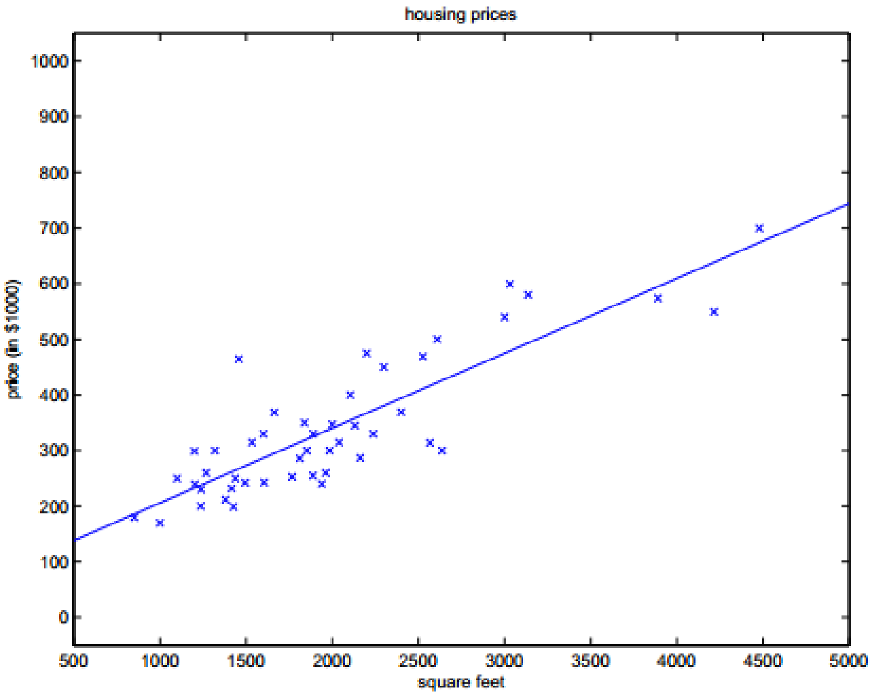
```
if __name__ == "__main__":
    #   日期      开盘      最高      最低      收盘      成交量        成交额
    price = np.loadtxt('sh_600000.txt', delimiter='\t', skiprows=2, usecols=(4,))
    print "原始价格：\n", price
    n = 100           # 阶数
    y = price[n:]
    m = len(y)        # 样本个数
    print "预测价格：\n", y
    x = np.zeros((m, n+1))
    for i in range(m):
        x[i] = np.hstack((price[i:i+n], 1))
    print "自变量：\n", x
    theta = np.linalg.lstsq(x, y)[0]      # theta为回归系数
    print theta
    show(theta, x, y)

    # 预测
    pn = 20      # 预测未来多少天
    x = price[-n:]
    y = np.hstack((price[-n:], np.zeros(pn)))
    for i in range(pn):
        y[n+i] = np.dot(theta, np.hstack((y[i:n+i], 1)))
    show_predict(y, pn)
```



线性回归

- □ y=ax+b

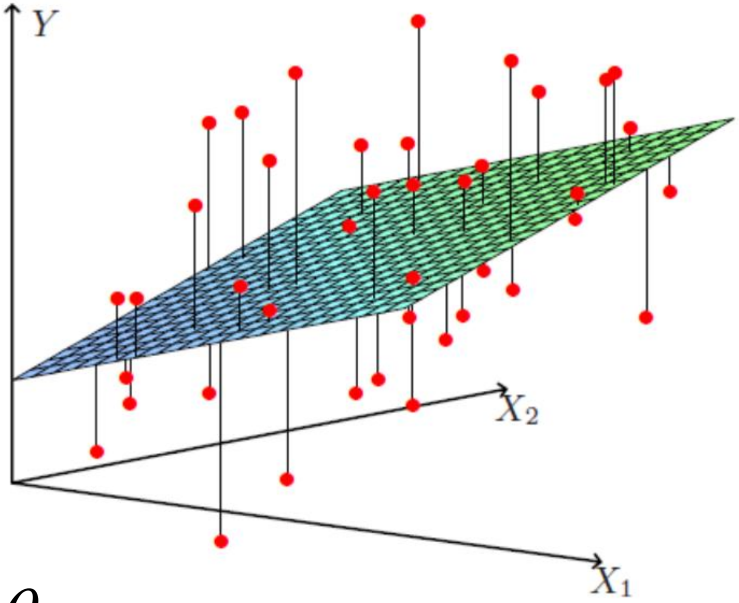

housing prices
price (in $1000)
1000
900
800
700
600
500
400
300
200
100
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
square feet


housing prices
price (in $1000)
1000
900
800
700
600
500
400
300
200
100
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
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
⋮	⋮	⋮

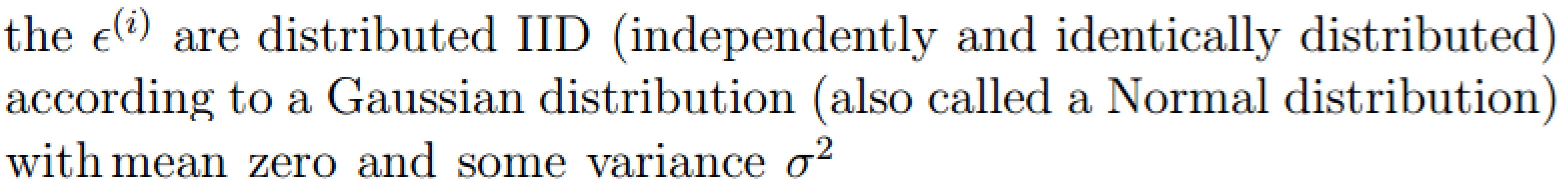


$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

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

使用极大似然估计解释最小二乘

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



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

theta由各种因素构成，其加和根据中心极限定理服从高斯分布

- □ 误差$\varepsilon^{(i)}(1 \leq i \leq m)$是独立同分布的，服从均值为0，方差为某定值$\sigma^2$的高斯分布。
  - 原因：中心极限定理

中心极限定理的意义

- □ 实际问题中，很多随机现象可以看做众多因素的独立影响的综合反应，往往近似服从正态分布。
  - 城市耗电量：大量用户的耗电量总和
  - 测量误差：许多观察不到的、微小误差的总和

  - 注：应用前提是多个随机变量的和，有些问题是乘性误差，则需要鉴别或者取对数后再使用。

似然函数 $y^{(i)} = \theta^T x^{(i)} + \varepsilon^{(i)}$

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

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

由于是独立同分布所以可以概率是联乘。

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

高斯的对数似然与最小二乘

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

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

θ的解析式的求解过程

- □ 将M个N维样本组成矩阵X：
  - ■ X的每一行对应一个样本，共M个样本(measurements)
  - ■ X的每一列对应样本的一个维度，共N维(regressors)
    - □ 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：

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

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

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

最小二乘意义下的参数最优解

n * 1　　n*1

□ 参数的解析式

n*n　　n*m　　m*1

X是m*n

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

□ 若X^TX不可逆或防止过拟合，增加λ扰动

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

□ “简便”方法记忆结论

$$X\theta=y\Rightarrow X^TX\theta=X^Ty$$

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

加入λ扰动后

- □ X^TX半正定：对于任意的非零向量u

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

- □ 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。

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

Ridge传入的参数就是lambda

线性回归的复杂度惩罚因子

- 线性回归的目标函数为：

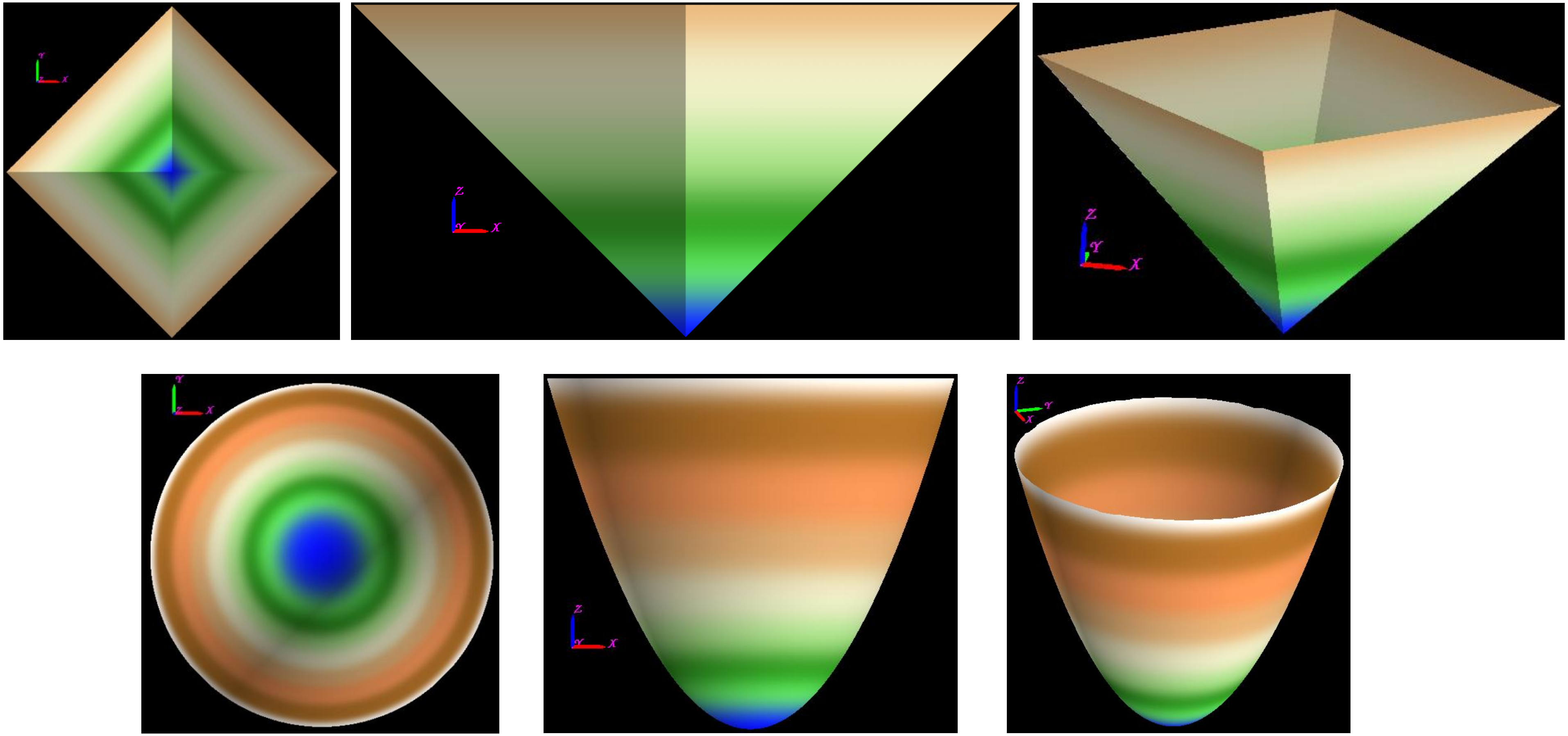
$$J\left(\vec{\theta}\right)=\frac{1}{2}\sum_{i=1}^{m}\left(h_{\vec{\theta}}\left(x^{(i)}\right)-y^{(i)}\right)^{2}$$

- 将目标函数增加平方和损失：算出的结果竟然和上一页的结论一样。

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

- 本质即为假定参数θ服从高斯分布。

$|w|$与$|w|^2$

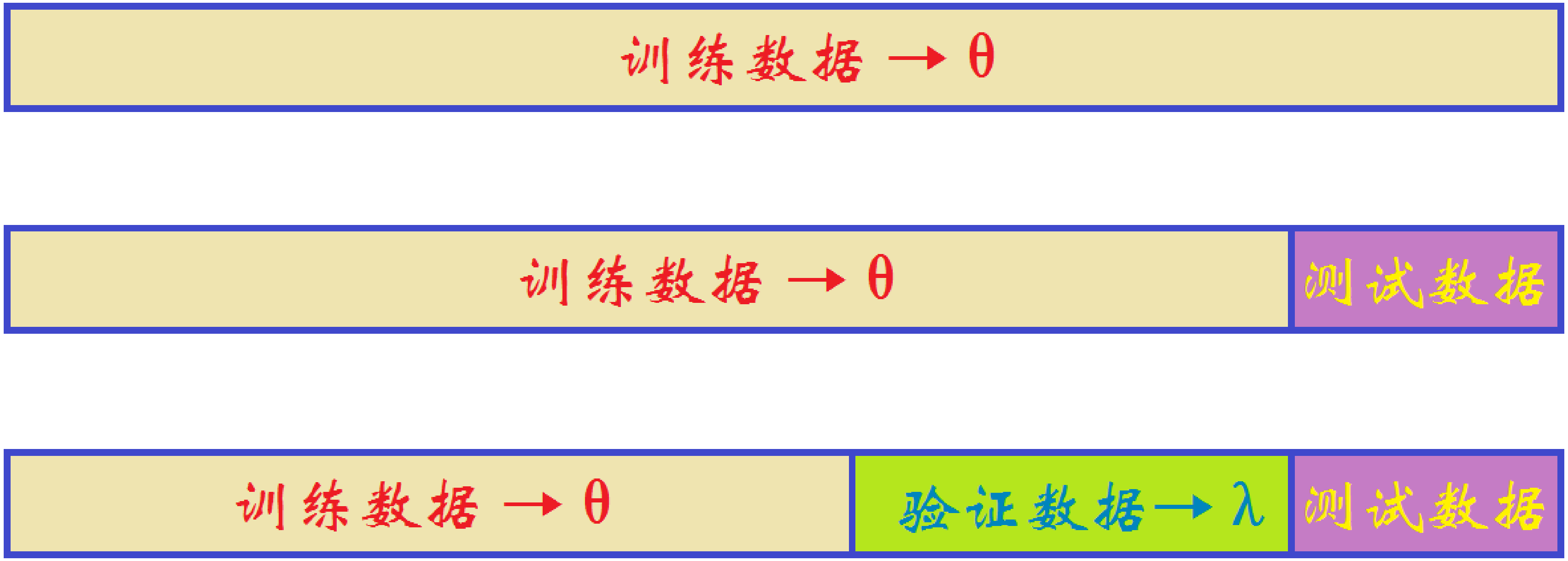


可以将一些theta为0，相当于将一些特征去掉了。

L1-norm如何处理梯度？

- □ 目标函数： $J\left(\vec{\theta}\right)=\frac{1}{2}\sum_{i=1}^{m}\left(h_{\vec{\theta}}\left(x^{(i)}\right)-y^{(i)}\right)^{2}+\lambda\sum_{j=1}^{n}\left|\theta_j\right|$
- □ 给定： $f(x;\alpha)=x+\frac{1}{\alpha}\log\left(1+\exp\left(-\alpha x\right)\right),\ x\geq 0$
- □ 近似： $|x|\approx f(x;\alpha)+f(-x;\alpha)=\frac{1}{\alpha}\log\left(1+\exp\left(-\alpha x\right)+1+\exp\left(\alpha x\right)\right)$
- □ 梯度： $\nabla|x|\approx\frac{1}{1+\exp\left(-\alpha x\right)}-\frac{1}{1+\exp\left(\alpha x\right)}$
- □ 二阶导： $\nabla^2|x|\approx\frac{2\alpha\exp\left(\alpha x\right)}{\left(1+\exp\left(\alpha x\right)\right)^2}$
- □ 实践中，对于一般问题，如取： $\alpha=10^6$

机器学习与数据使用



训练数据 → θ

训练数据 → θ　测试数据

训练数据 → θ　验证数据 → λ　测试数据

- 交叉验证
  - 如：十折交叉验证

广义逆矩阵(伪逆)

- 若A为非奇异矩阵，则线性方程组Ax=b的解为 $x=(A^TA)^{-1}A^T\cdot b$ ，从方程解的直观意义上，可以定义： $A^+=(A^TA)^{-1}A^T$
- 若A为可逆方阵，$A^+=(A^TA)^{-1}A^T$ 即为 A^{-1}

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

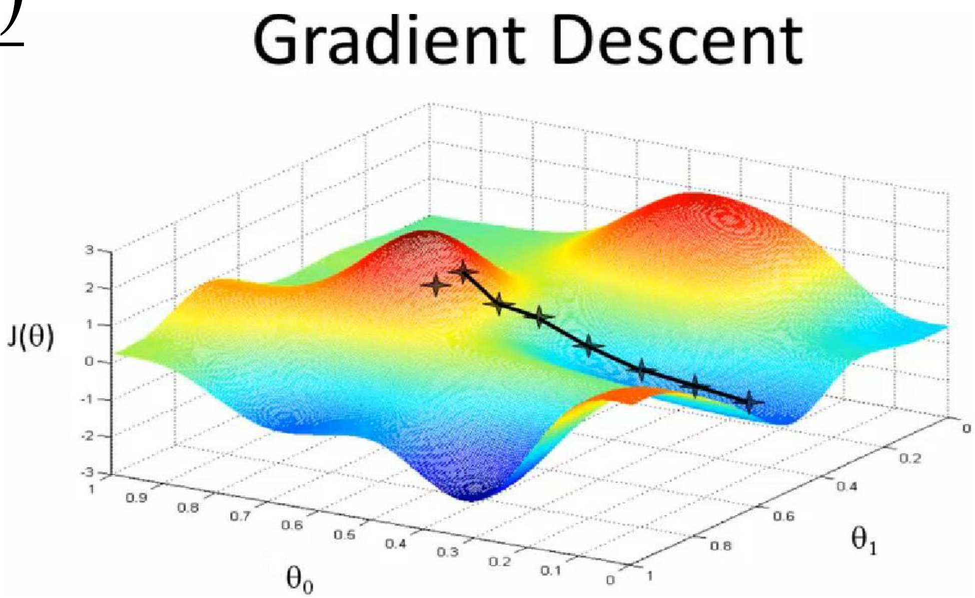
- 当A为矩阵(非方阵)时，称A$^+$称为A的广义逆(伪逆)。
  - 奇异值分解SVD

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

- 初始化θ(随机初始化)
- 沿着负梯度方向迭代，更新后的θ使J(θ)更小

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

  - α：学习率、步长

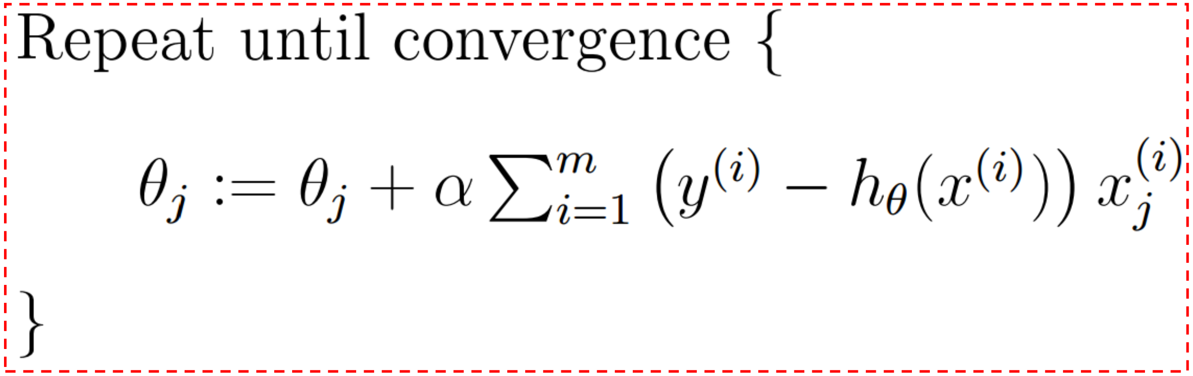


梯度方向

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

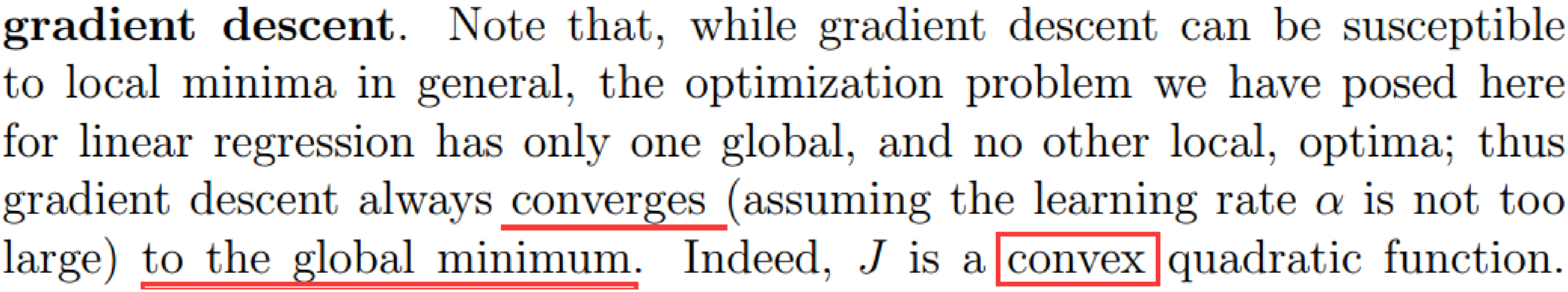
拿到所有样本之后再下降

批量梯度下降算法



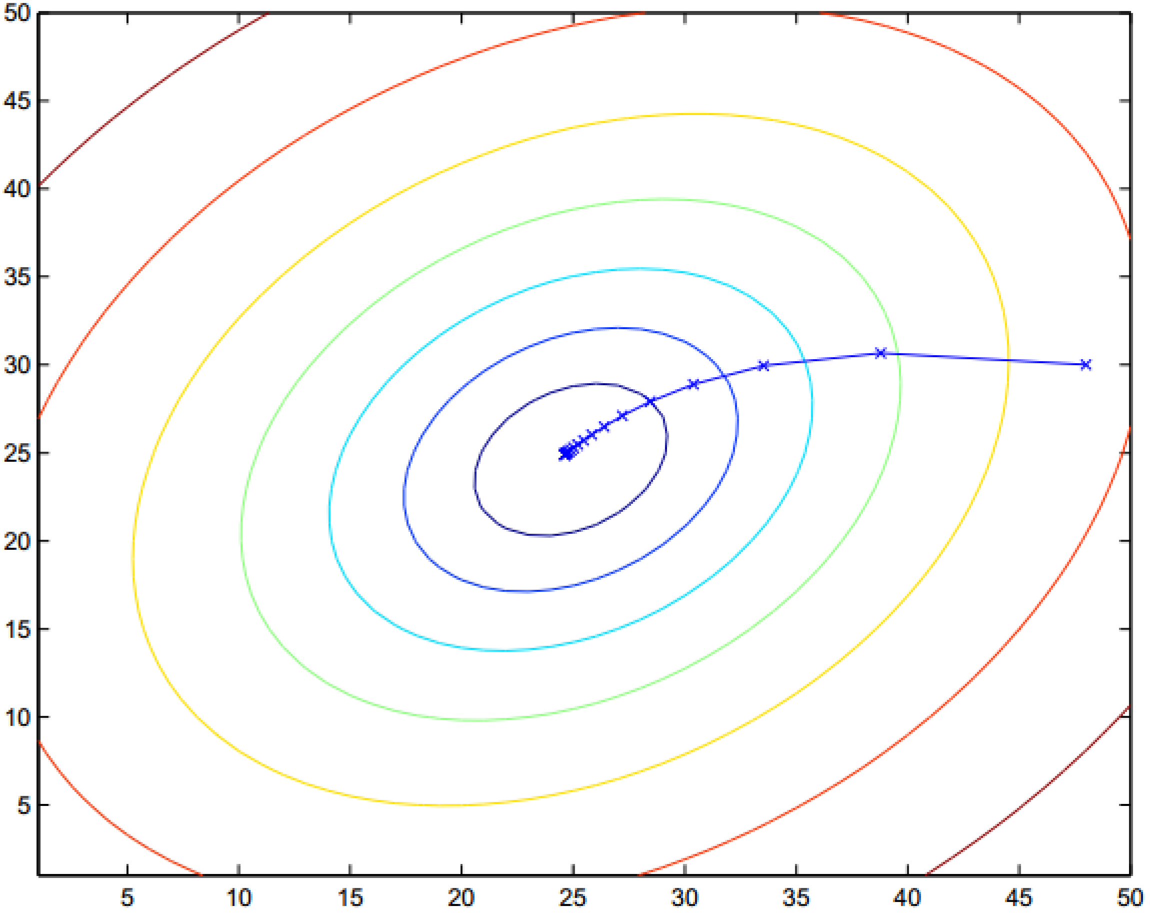
```
Repeat until convergence {
    $\theta_j := \theta_j + \alpha \sum_{i=1}^{m} \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}$
}
```

关于theta是一个凸函数



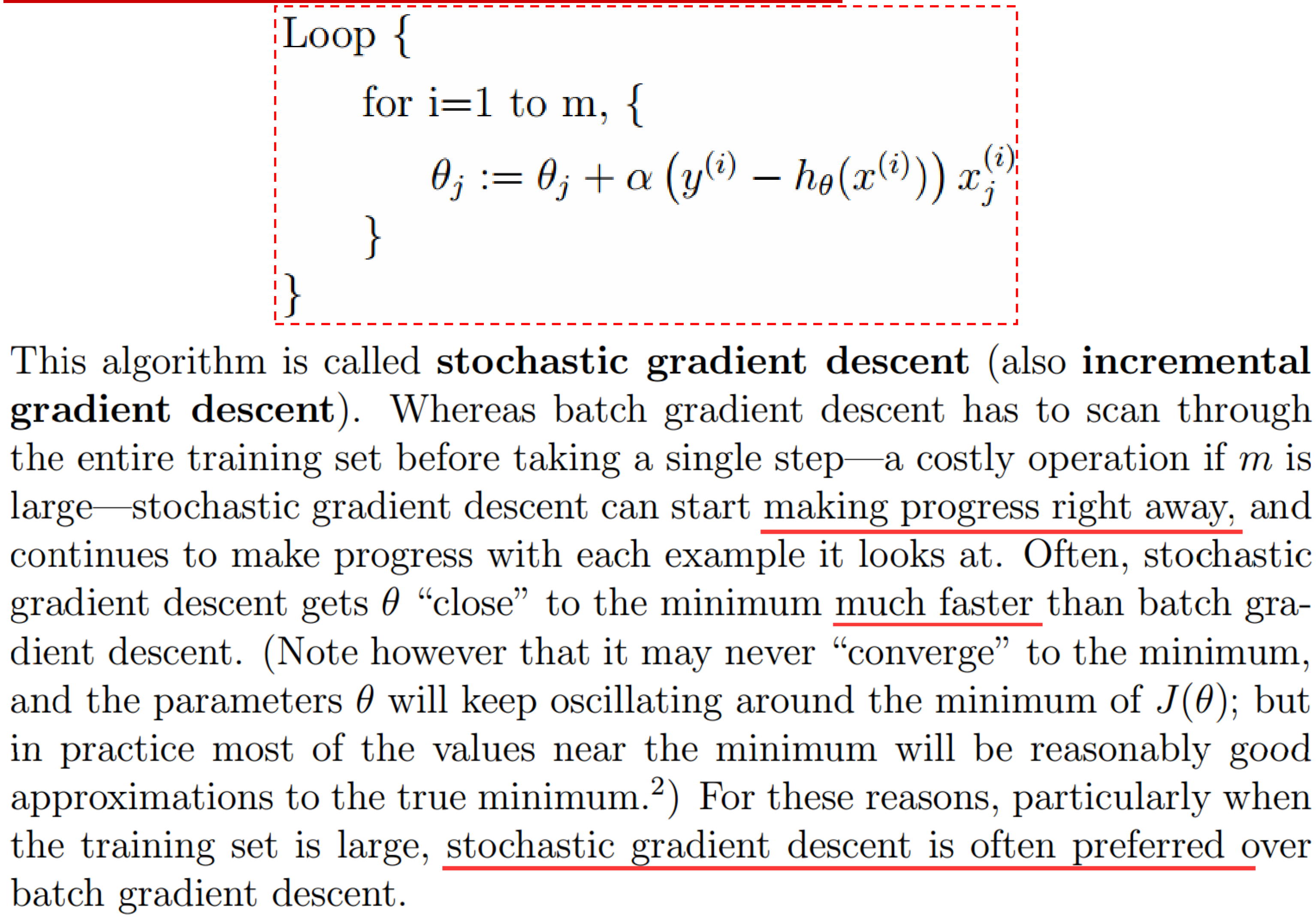
gradient descent. Note that, while gradient descent can be susceptible to local minima in general, the optimization problem we have posed here for linear regression has only one global, and no other local, optima; thus gradient descent always converges (assuming the learning rate α is not too large)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

批量梯度下降图示



拿到一个值就开始下降，可能某次下降并不是全局下降但是堪用，有噪声也可能是好事。用于在线学习

随机梯度下降算法

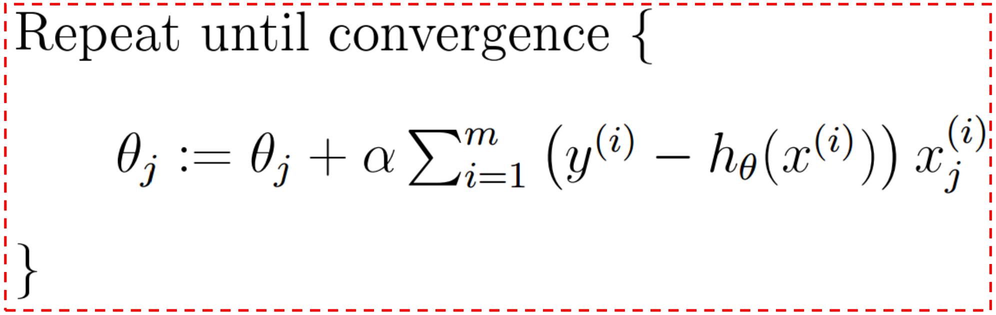


```
Loop {
    for i=1 to m, {
        $\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}$
    }
}
```

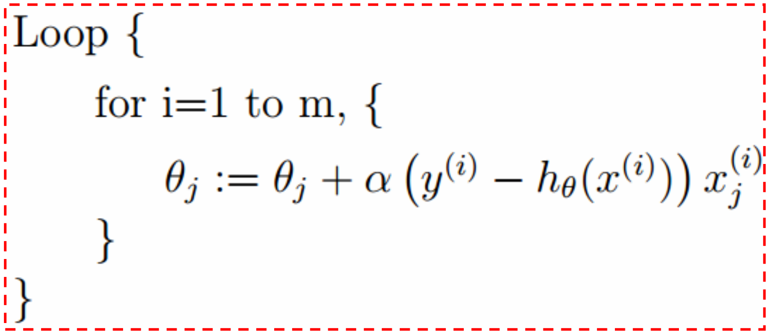
This algorithm is called **stochastic gradient descent** (also **incremental gradient descent**). Whereas batch gradient descent has to scan through the entire training set before taking a single step—a costly operation if m is large—stochastic gradient descent can start making progress right away, and 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.

折中：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。

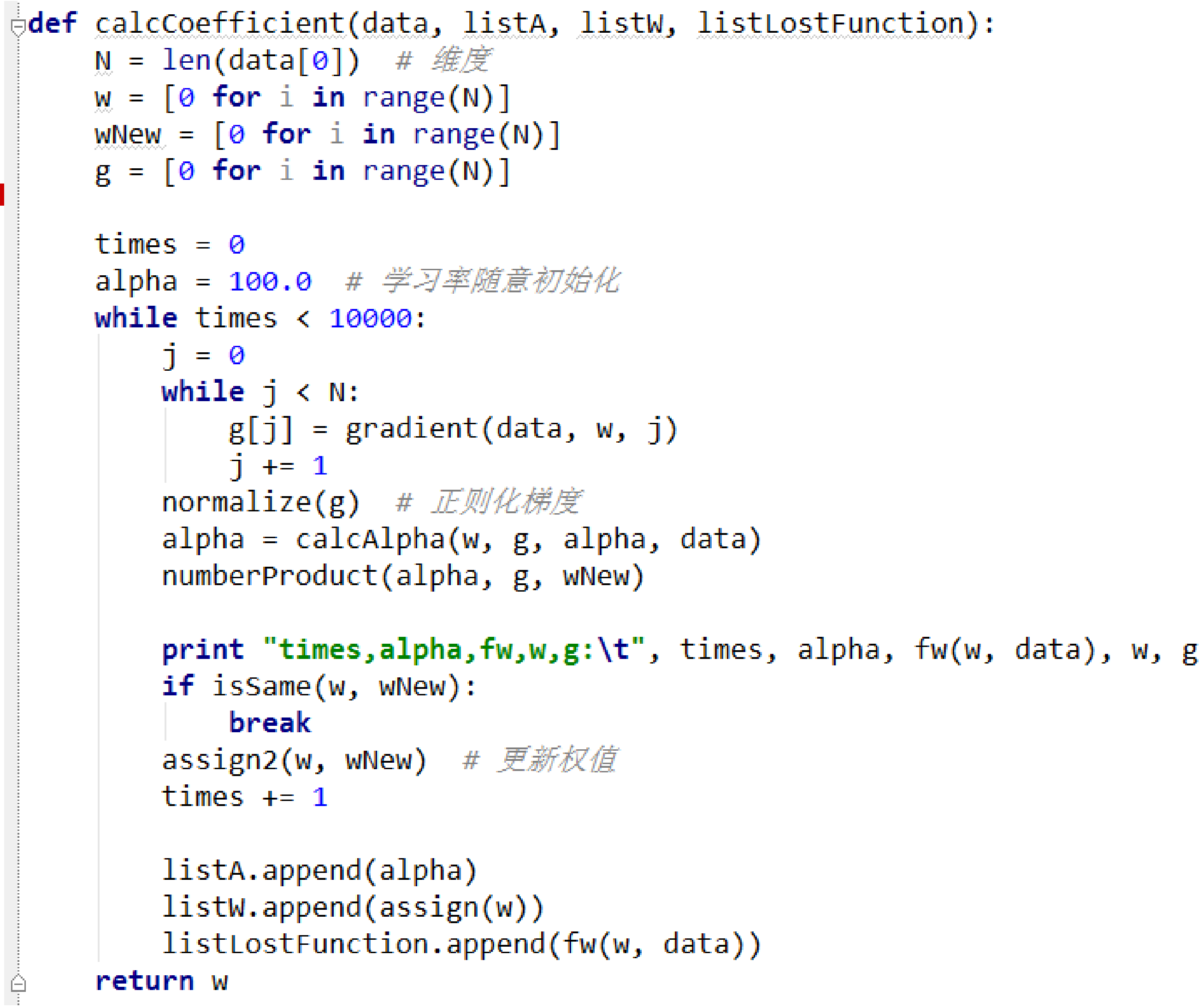


```
Repeat until convergence {
    $\theta_j := \theta_j + \alpha \sum_{i=1}^{m} \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}$
}
```



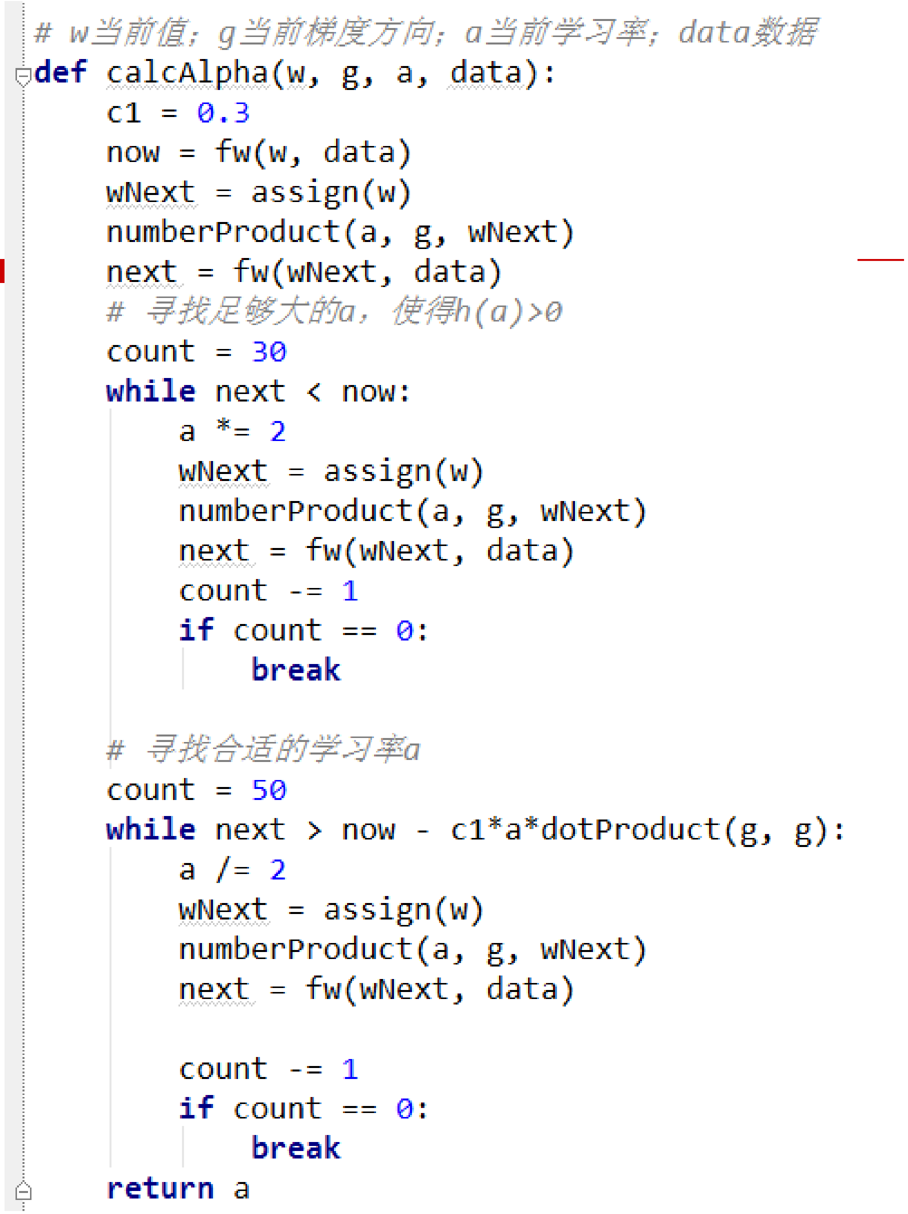
```
Loop {
    for i=1 to m, {
        $\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}$
    }
}
```

回归Code



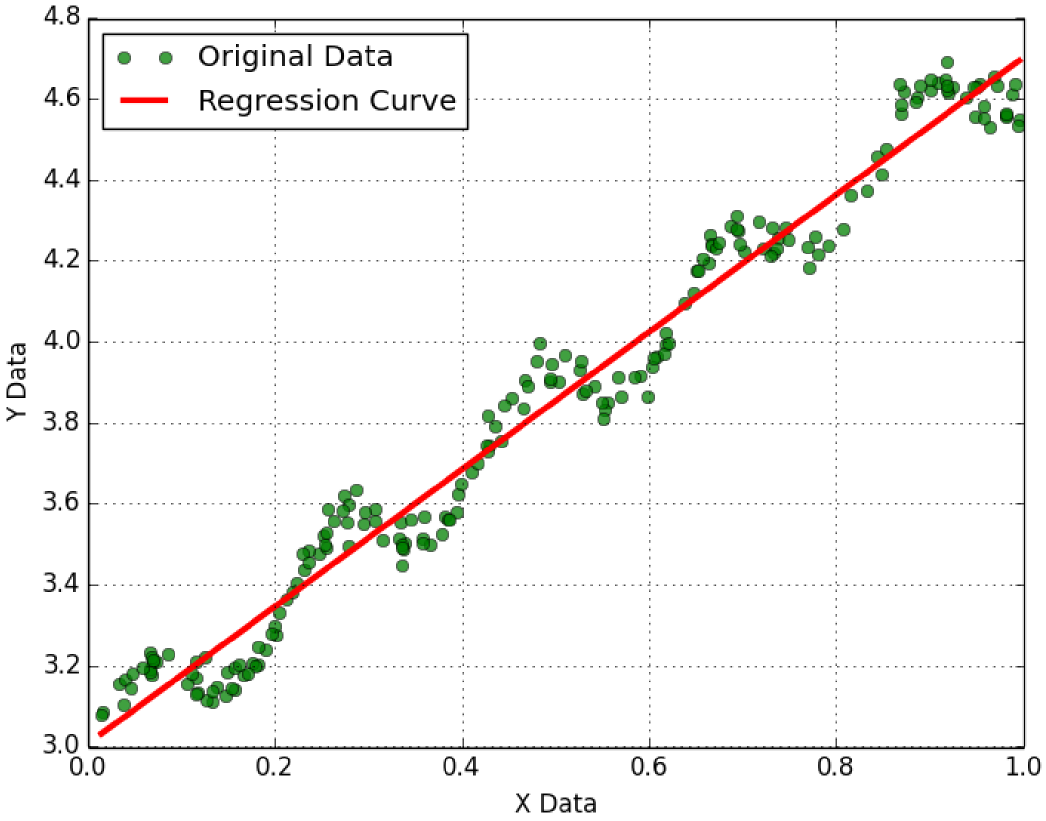
```
def calcCoefficient(data, listA, listW, listLostFunction):
    N = len(data[0])  # 维度
    w = [0 for i in range(N)]
    wNew = [0 for i in range(N)]
    g = [0 for i in range(N)]

    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
```

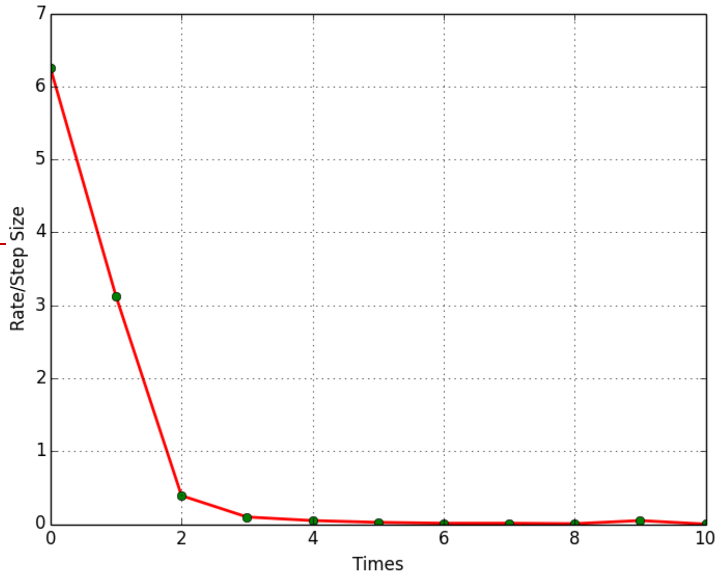
附：学习率Code

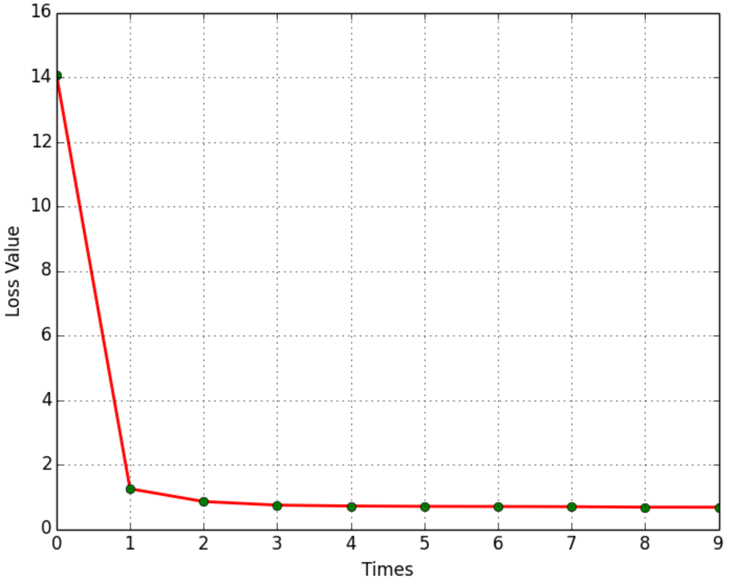


```
# w当前值；g当前梯度方向；a当前学习率；data数据
def calcAlpha(w, g, a, data):
    c1 = 0.3
    now = fw(w, data)
    wNext = assign(w)
    numberProduct(a, g, wNext)
    next = fw(wNext, data)
    # 寻找足够大的a，使得h(a)>0
    count = 30
    while next < now:
        a *= 2
        wNext = assign(w)
        numberProduct(a,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

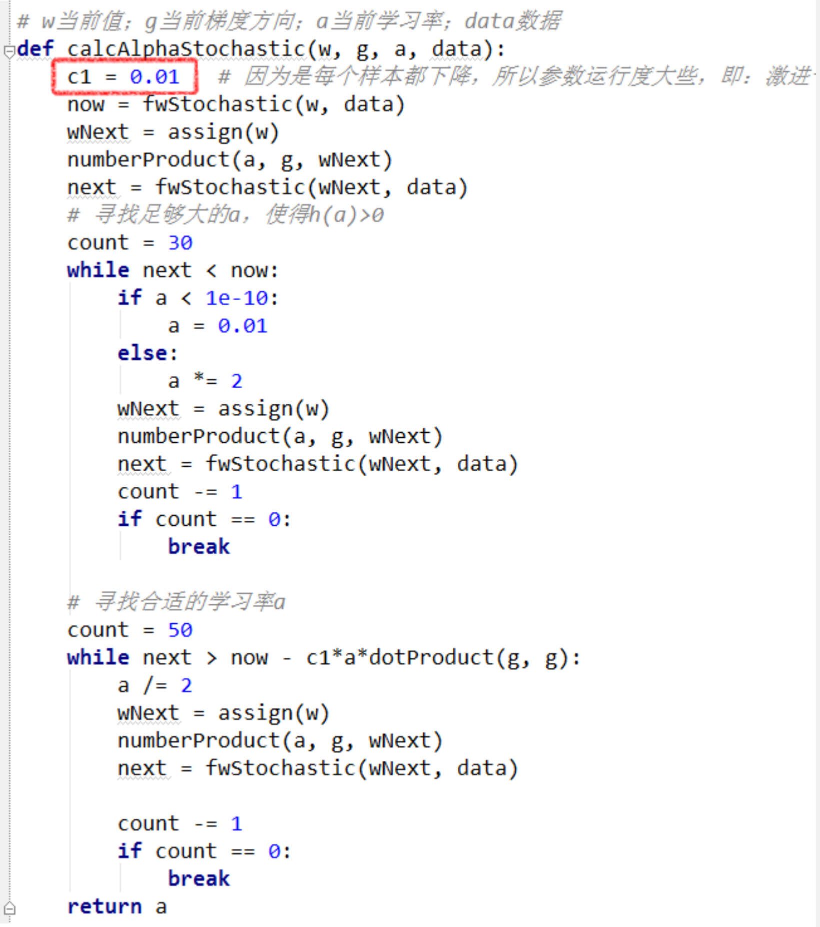
线性回归、rate、Loss



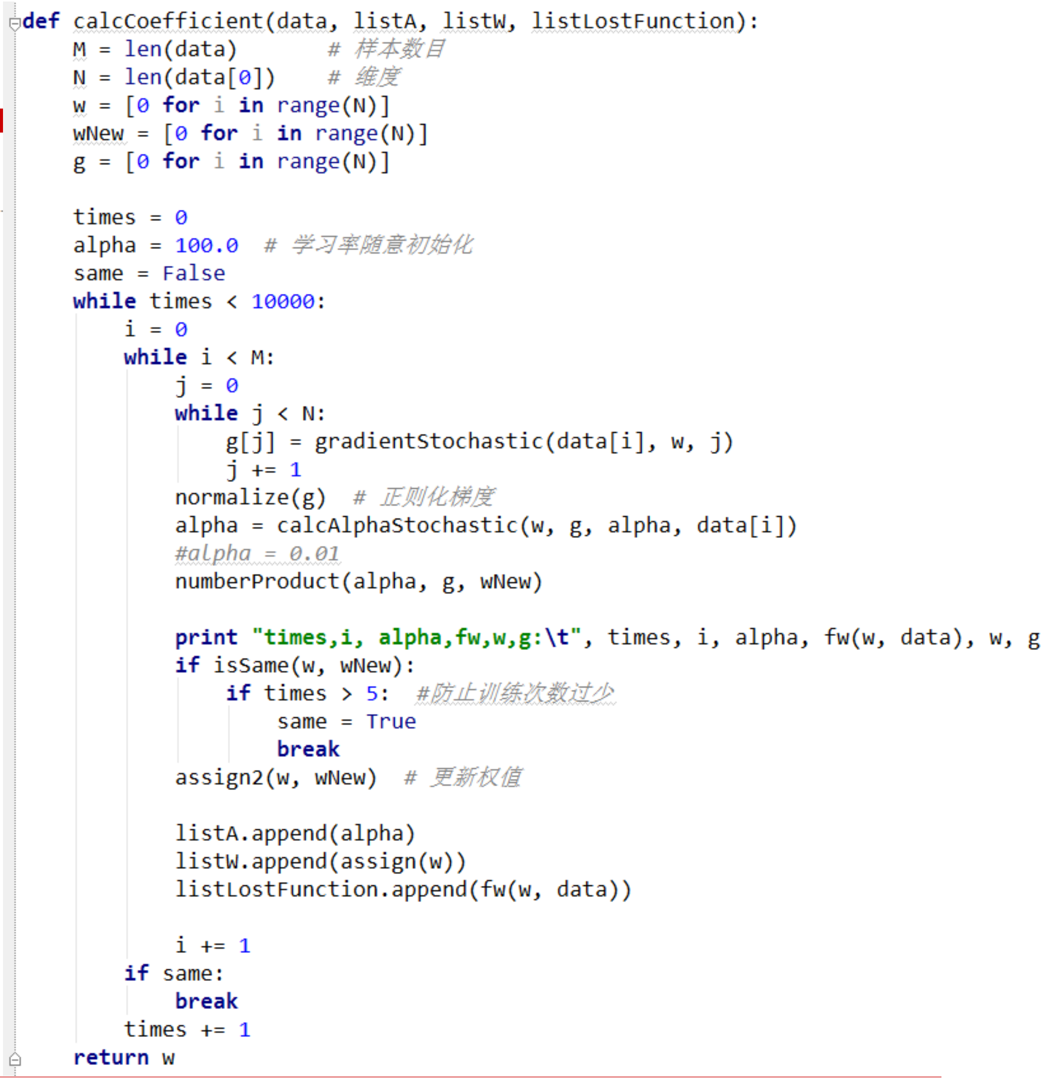




SGD与学习率

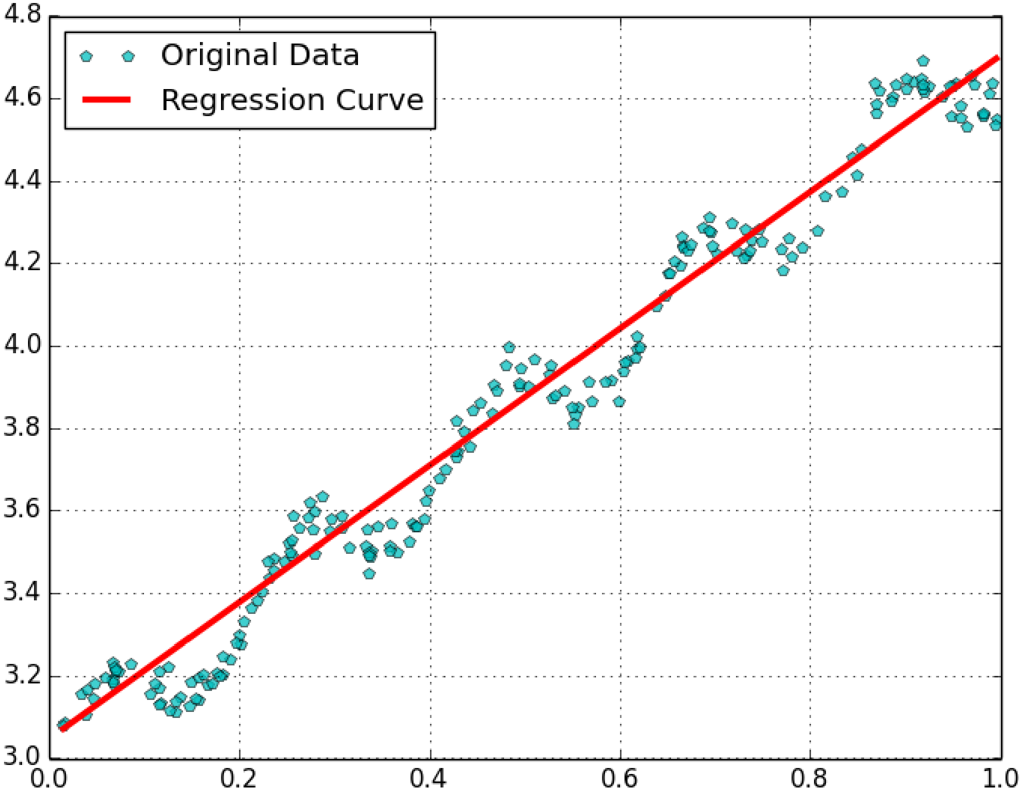


```python
# w当前值；g当前梯度方向；a当前学习率；data数据
def calcAlphaStochastic(w, g, a, data):
    c1 = 0.01   # 因为是每个样本都下降，所以参数运行度大些，即：激进
    now = fwStochastic(w, data)
    wNext = assign(w)
    numberProduct(a, g, wNext)
    next = fwStochastic(wNext, data)
    # 寻找足够大的a，使得h(a)>0
    count = 30
    while next < now:
        if a < 1e-10:
            a = 0.01
        else:
            a *= 2
        wNext = assign(w)
        numberProduct(a, g, wNext)
        next = fwStochastic(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 = fwStochastic(wNext, data)

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

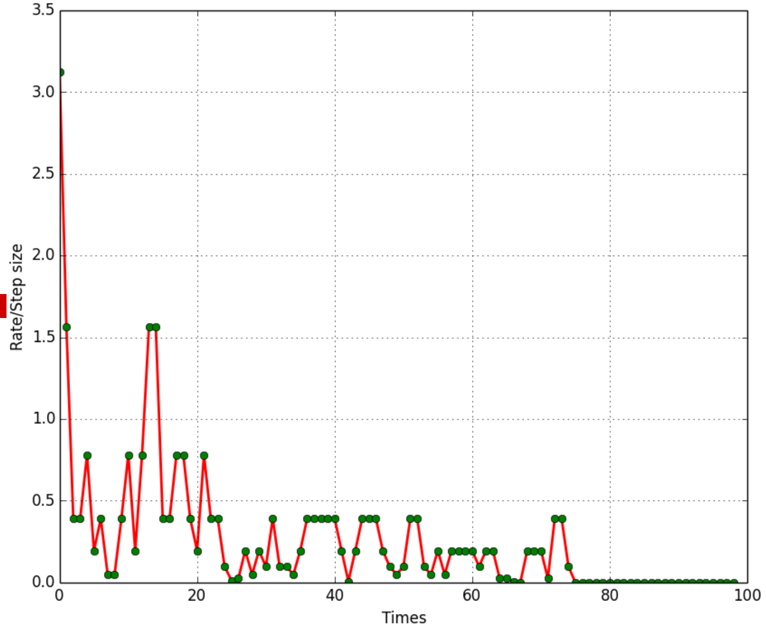


```python
def calcCoefficient(data, listA, listW, listLostFunction):
    M = len(data)       # 样本数目
    N = len(data[0])    # 维度
    w = [0 for i in range(N)]
    wNew = [0 for i in range(N)]
    g = [0 for i in range(N)]

    times = 0
    alpha = 100.0  # 学习率随意初始化
    same = False
    while times < 10000:
        i = 0
        while i < M:
            j = 0
            while j < N:
                g[j] = gradientStochastic(data[i], w, j)
                j += 1
            normalize(g)  # 正则化梯度
            alpha = calcAlphaStochastic(w, g, alpha, data[i])
            #alpha = 0.01
            numberProduct(alpha, g, wNew)

            print "times,i, alpha,fw,w,g:\t", times, i, alpha, fw(w, data), w, g
            if isSame(w, wNew):
                if times > 5:  #防止训练次数过少
                    same = True
                    break
            assign2(w, wNew)  # 更新权值

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

            i += 1
        if same:
            break
        times += 1
    return w
```

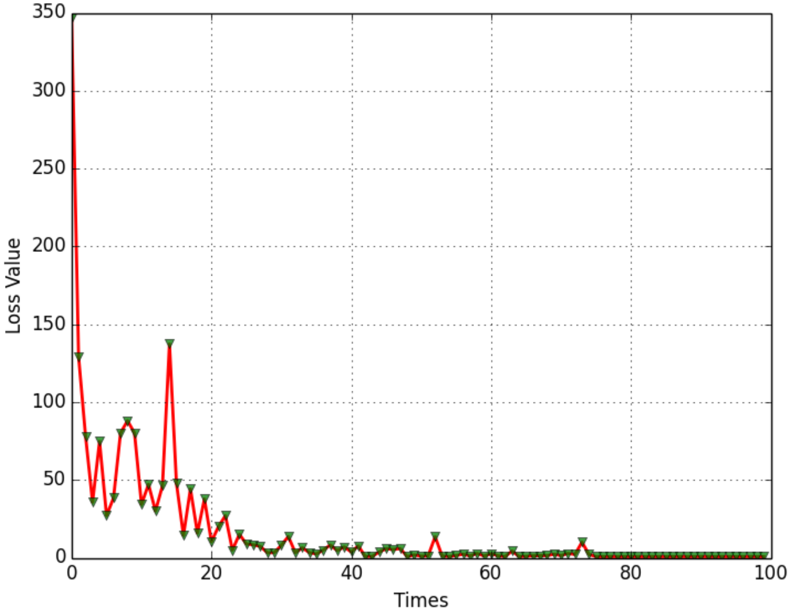
随机梯度下降SGD



Original Data
Regression Curve
4.8
4.6
4.4
4.2
4.0
3.8
3.6
3.4
3.2
3.0
0.0
0.2
0.4
0.6
0.8
1.0

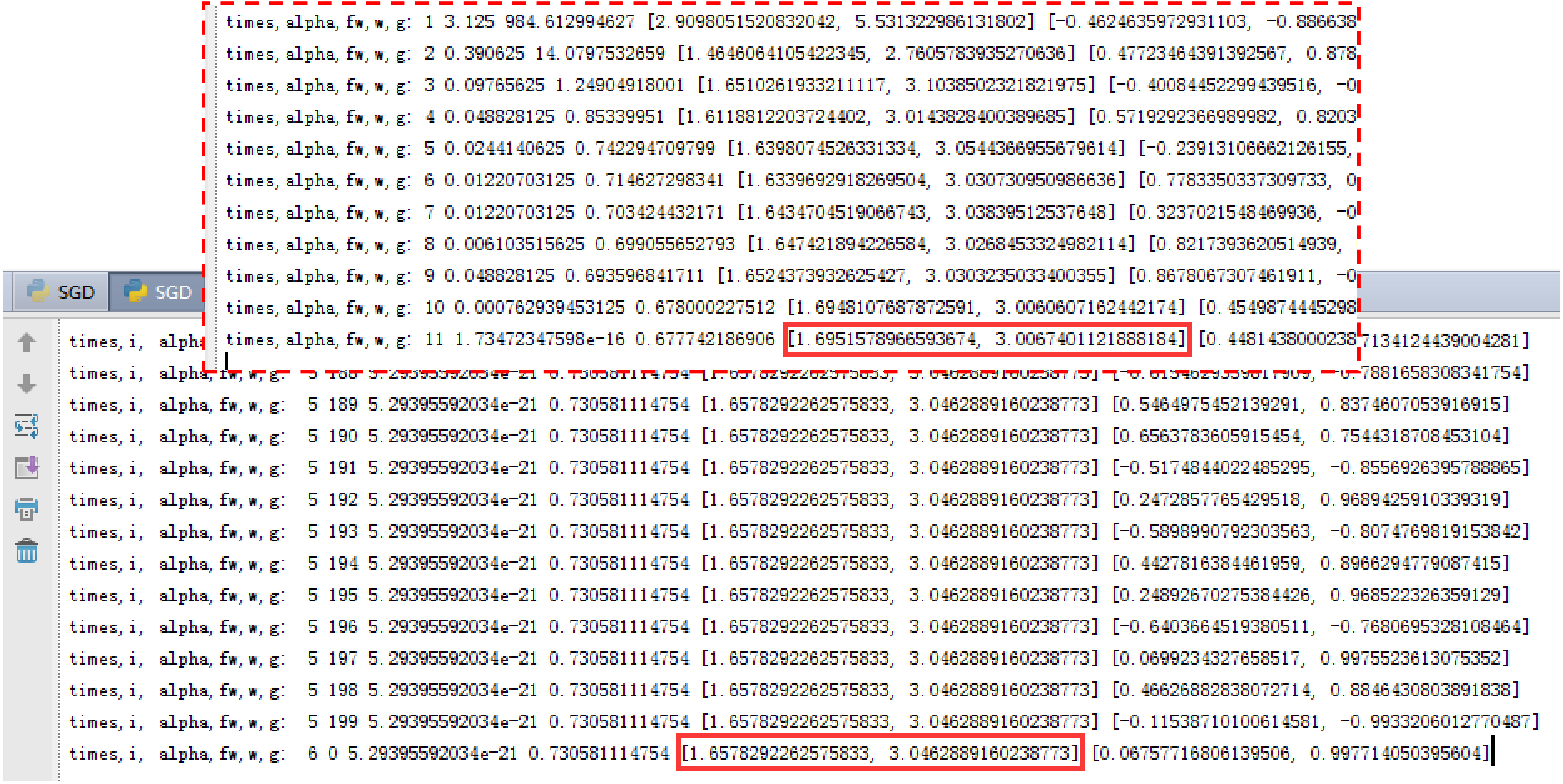


Rate/Step size
Times
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
20
40
60
80
100



Loss Value
Times
350
300
250
200
150
100
50
0
0
20
40
60
80
100

批量与随机梯度下降

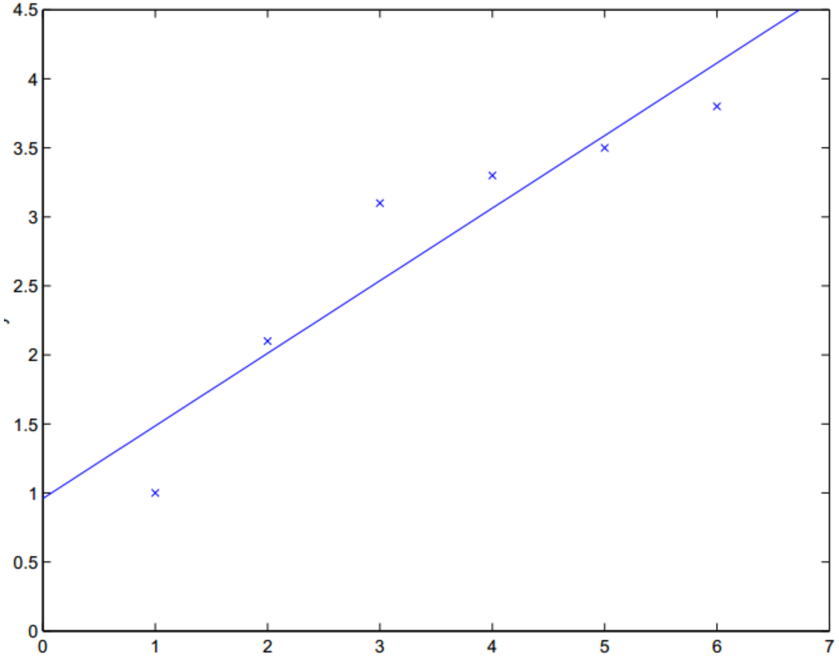


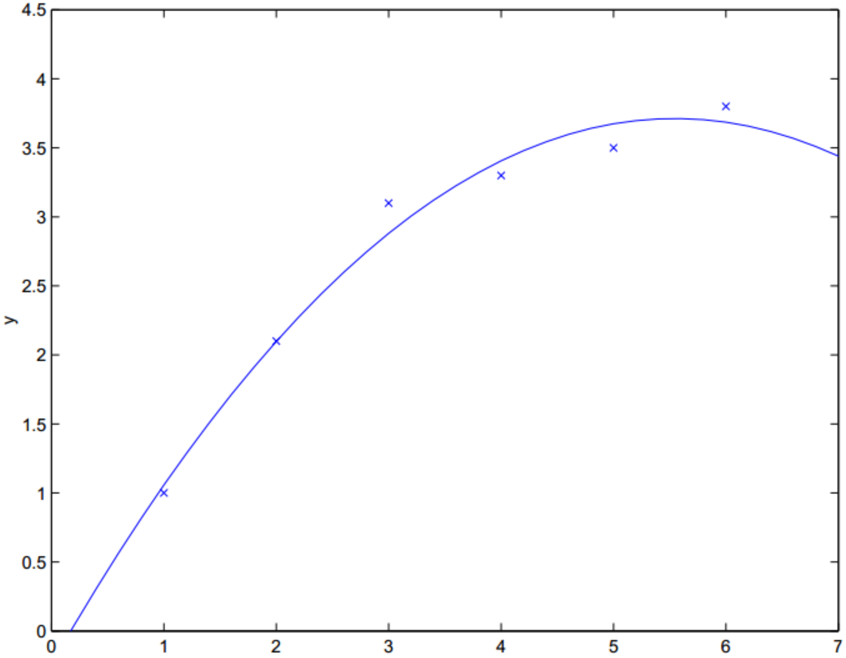
```
times, alpha, fw, w, g: 1 3.125 984.612994627 [2.9098051520832042, 5.531322986131802] [-0.4624635972931103, -0.886638
times, alpha, fw, w, g: 2 0.390625 14.0797532659 [1.4646064105422345, 2.7605783935270636] [0.47723464391392567, 0.878
times, alpha, fw, w, g: 3 0.09765625 1.24904918001 [1.6510261933211117, 3.1038502321821975] [-0.40084452299439516, -0
times, alpha, fw, w, g: 4 0.048828125 0.85339951 [1.6118812203724402, 3.0143828400389685] [0.5719292366989982, 0.8203
times, alpha, fw, w, g: 5 0.0244140625 0.742294709799 [1.6398074526331334, 3.0544366955679614] [-0.23913106662126155,
times, alpha, fw, w, g: 6 0.01220703125 0.714627298341 [1.6339692918269504, 3.030730950986636] [0.7783350337309733, 0
times, alpha, fw, w, g: 7 0.01220703125 0.703424432171 [1.6434704519066743, 3.03839512537648] [0.3237021548469936, -0
times, alpha, fw, w, g: 8 0.006103515625 0.699055652793 [1.647421894226584, 3.0268453324982114] [0.8217393620514939,
times, alpha, fw, w, g: 9 0.048828125 0.693596841711 [1.6524373932625427, 3.0303235033400355] [0.8678067307461911, -0
times, alpha, fw, w, g: 10 0.000762939453125 0.678000227512 [1.6948107687872591, 3.0060607162442174] [0.4549874445298
times, alpha, fw, w, g: 11 1.73472347598e-16 0.677742186906 [1.6951578966593674, 3.0067401121888184] [0.4481438000238
```

```
times, i,  alph[illegible]7134124439004281]
times, i,  alpha, fw, w, g:  5 188 5.29395592034e-21 0.730581114754 [1.6578292262575833, 3.0462889160238773] [-0.615462[illegible], -0.7881658308341754]
times, i,  alpha, fw, w, g:  5 189 5.29395592034e-21 0.730581114754 [1.6578292262575833, 3.0462889160238773] [0.5464975452139291, 0.8374607053916915]
times, i,  alpha, fw, w, g:  5 190 5.29395592034e-21 0.730581114754 [1.6578292262575833, 3.0462889160238773] [0.6563783605915454, 0.7544318708453104]
times, i,  alpha, fw, w, g:  5 191 5.29395592034e-21 0.730581114754 [1.6578292262575833, 3.0462889160238773] [-0.5174844022485295, -0.8556926395788865]
times, i,  alpha, fw, w, g:  5 192 5.29395592034e-21 0.730581114754 [1.6578292262575833, 3.0462889160238773] [0.2472857765429518, 0.9689425910339319]
times, i,  alpha, fw, w, g:  5 193 5.29395592034e-21 0.730581114754 [1.6578292262575833, 3.0462889160238773] [-0.5898990792303563, -0.8074769819153842]
times, i,  alpha, fw, w, g:  5 194 5.29395592034e-21 0.730581114754 [1.6578292262575833, 3.0462889160238773] [0.4427816384461959, 0.8966294779087415]
times, i,  alpha, fw, w, g:  5 195 5.29395592034e-21 0.730581114754 [1.6578292262575833, 3.0462889160238773] [0.24892670275384426, 0.968522326359129]
times, i,  alpha, fw, w, g:  5 196 5.29395592034e-21 0.730581114754 [1.6578292262575833, 3.0462889160238773] [-0.6403664519380511, -0.7680695328108464]
times, i,  alpha, fw, w, g:  5 197 5.29395592034e-21 0.730581114754 [1.6578292262575833, 3.0462889160238773] [0.0699234327658517, 0.9975523613075352]
times, i,  alpha, fw, w, g:  5 198 5.29395592034e-21 0.730581114754 [1.6578292262575833, 3.0462889160238773] [0.46626882838072714, 0.8846430803891838]
times, i,  alpha, fw, w, g:  5 199 5.29395592034e-21 0.730581114754 [1.6578292262575833, 3.0462889160238773] [-0.11538710100614581, -0.9933206012770487]
times, i,  alpha, fw, w, g:  6 0 5.29395592034e-21 0.730581114754 [1.6578292262575833, 3.0462889160238773] [0.06757716806139506, 0.997714050395604]
```

线性回归的进一步分析

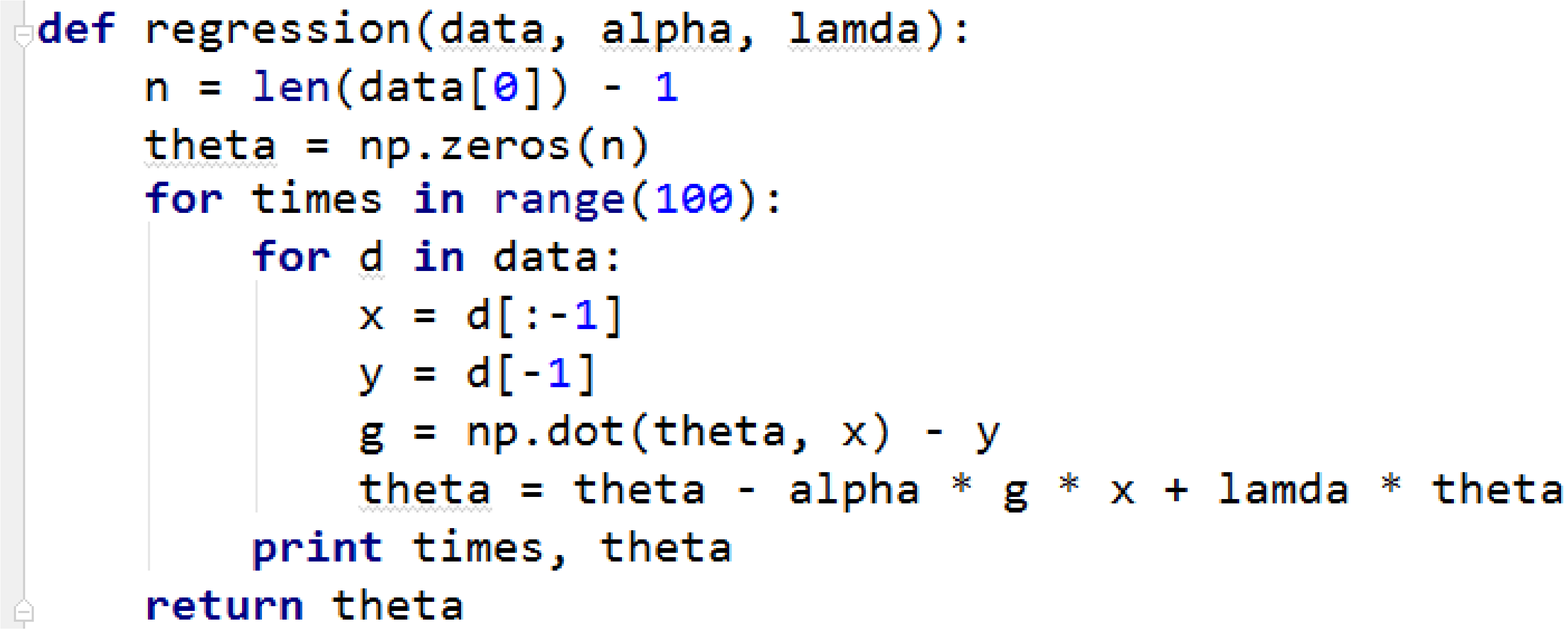
- 可以对样本是非线性的，只要对参数θ线性





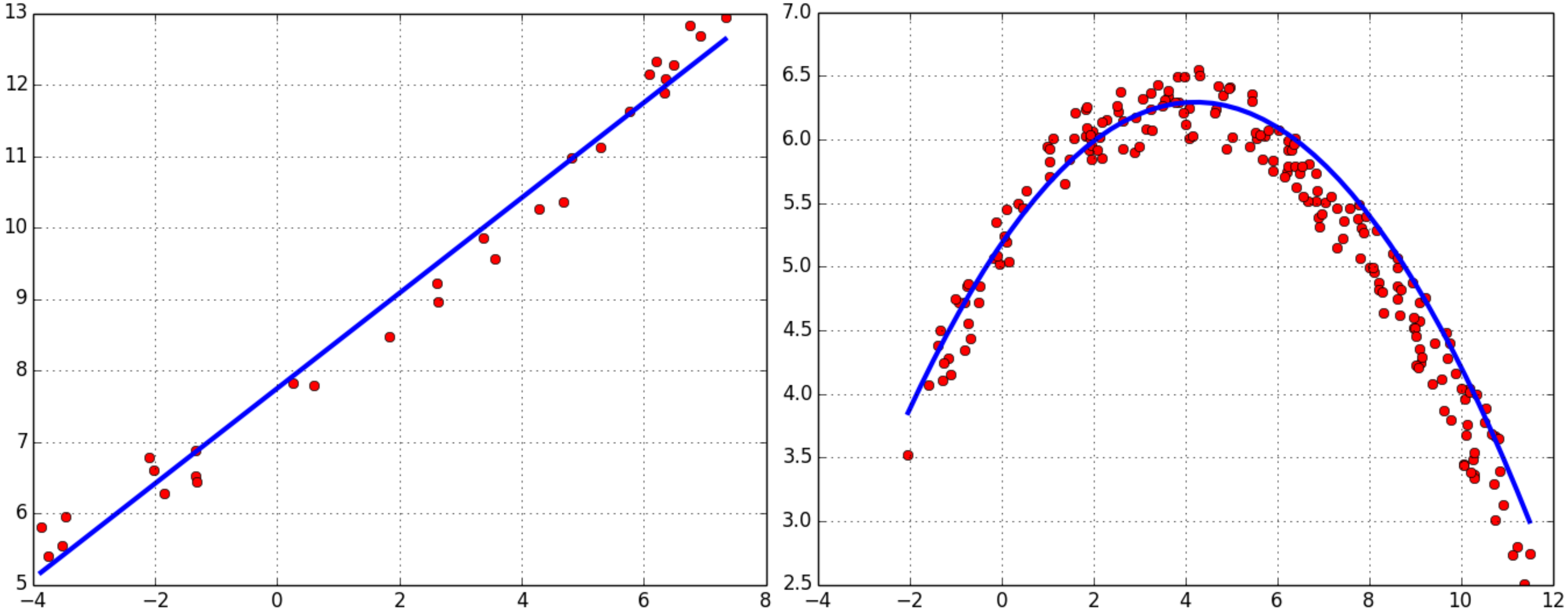
$$y = \theta_0 + \theta_1 x + \theta_2 x^2$$

Code

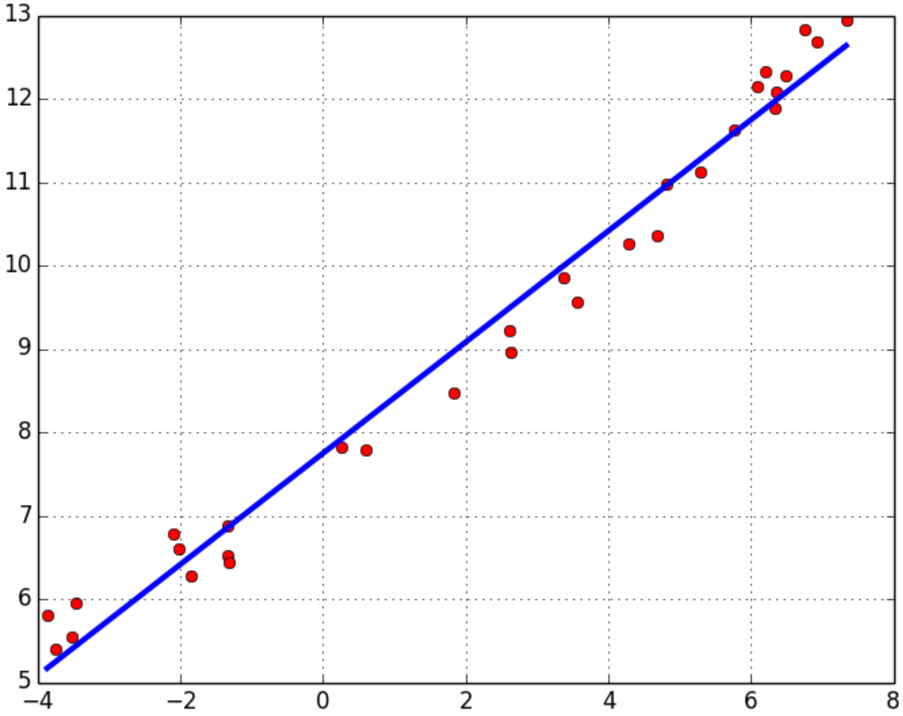


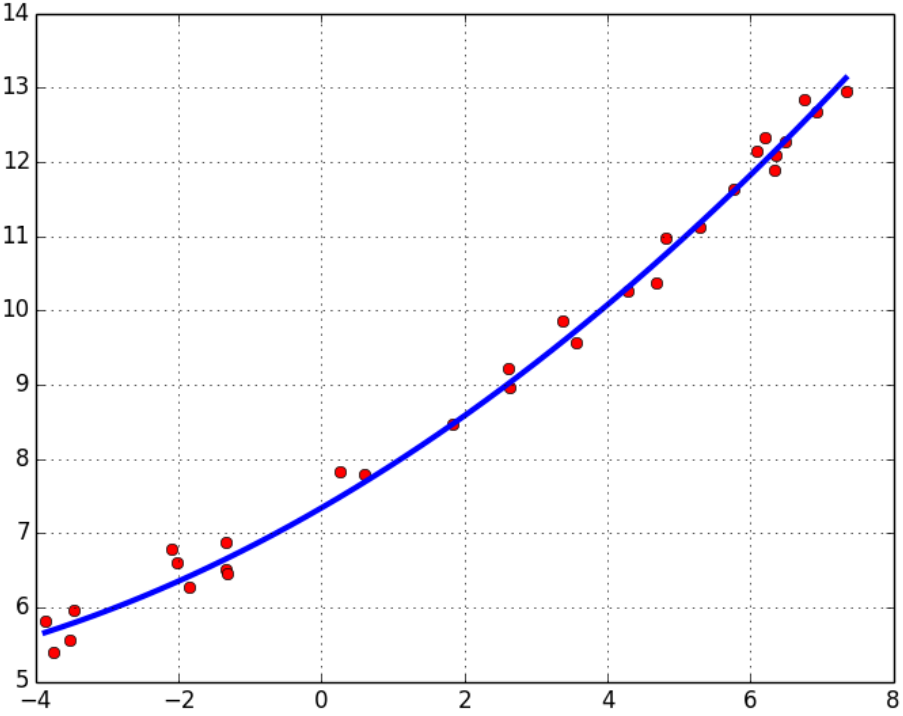
```
def regression(data, alpha, lamda):
    n = len(data[0]) - 1
    theta = np.zeros(n)
    for times in range(100):
        for d in data:
            x = d[:-1]
            y = d[-1]
            g = np.dot(theta, x) - y
            theta = theta - alpha * g * x + lamda * theta
        print times, theta
    return theta
```

线性回归

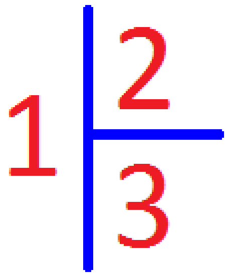

13
12
11
10
9
8
7
6
5
−4
−2
0
2
4
6
8
7.0
6.5
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
−4
−2
0
2
4
6
8
10
12

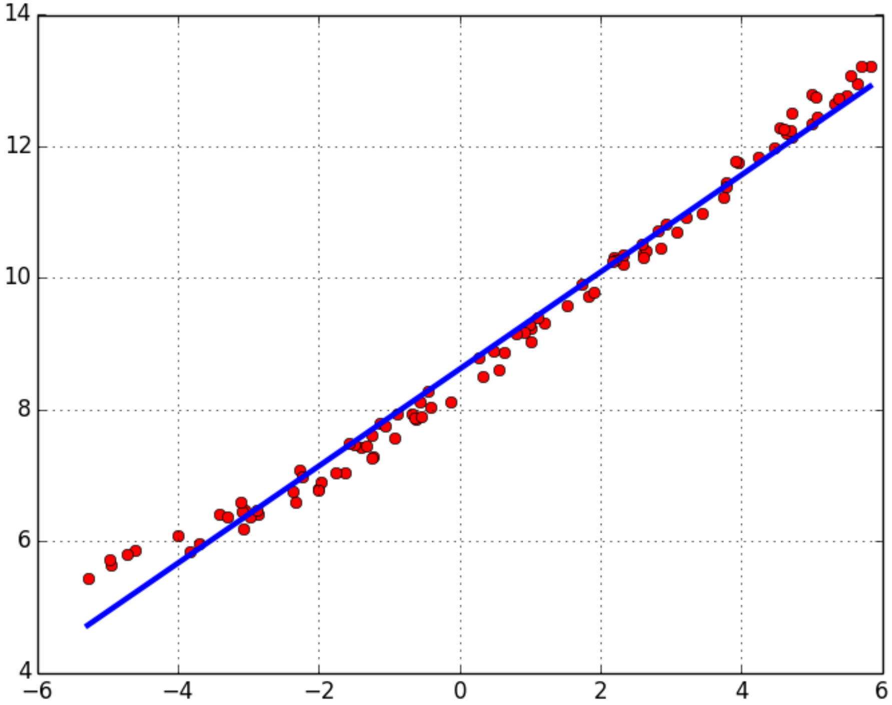
线性回归

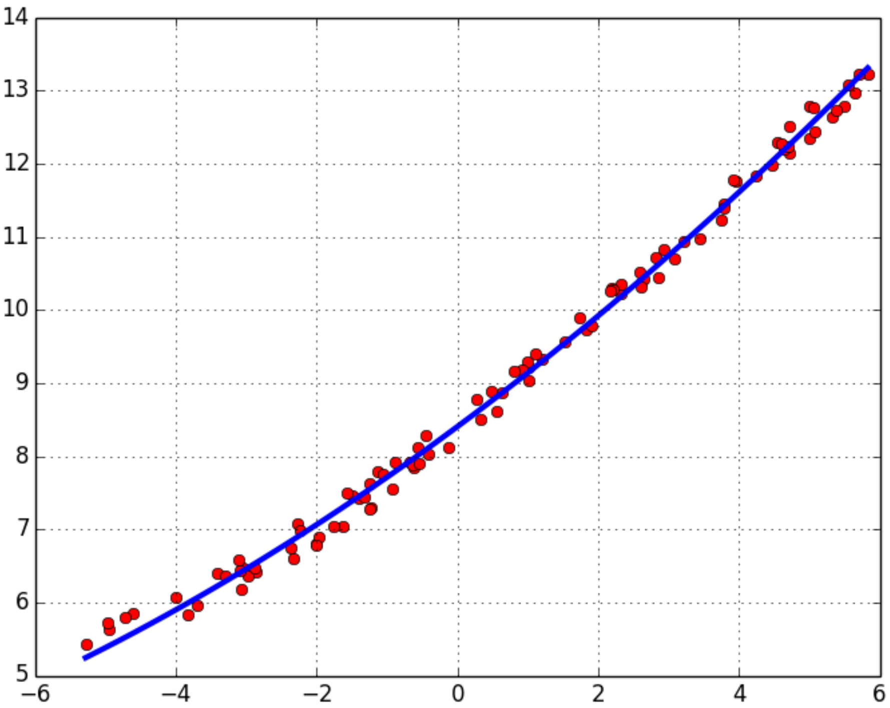


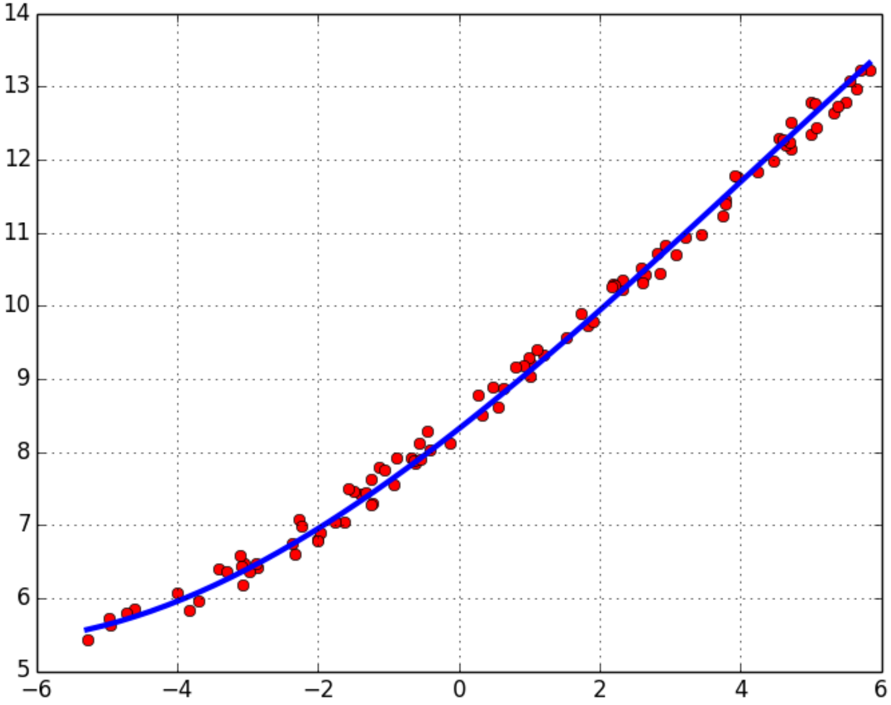


特征选择

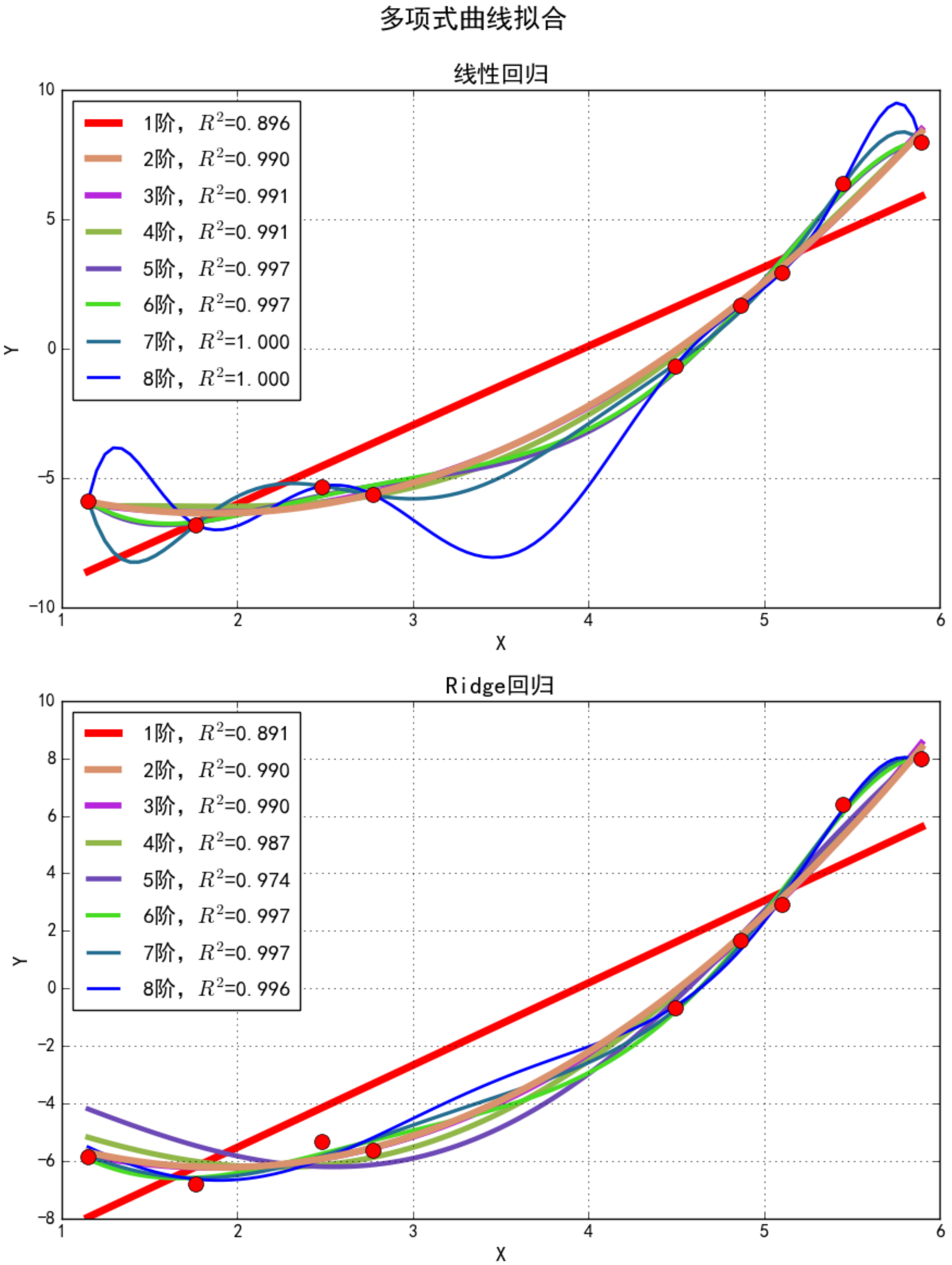

1
2
3


14
12
10
8
6
4
−6
−4
−2
0
2
4
6


14
13
12
11
10
9
8
7
6
5
−6
−4
−2
0
2
4
6


14
13
12
11
10
9
8
7
6
5
−6
−4
−2
0
2
4
6

超参与过拟合


多项式曲线拟合
线性回归
1阶，R^2=0.896
2阶，R^2=0.990
3阶，R^2=0.991
4阶，R^2=0.991
5阶，R^2=0.997
6阶，R^2=0.997
7阶，R^2=1.000
8阶，R^2=1.000
10
5
0
−5
−10
Y
1
2
3
4
5
6
X
Ridge回归
1阶，R^2=0.891
2阶，R^2=0.990
3阶，R^2=0.990
4阶，R^2=0.987
5阶，R^2=0.974
6阶，R^2=0.997
7阶，R^2=0.997
8阶，R^2=0.996
10
8
6
4
2
0
−2
−4
−6
−8
Y
1
2
3
4
5
6
X

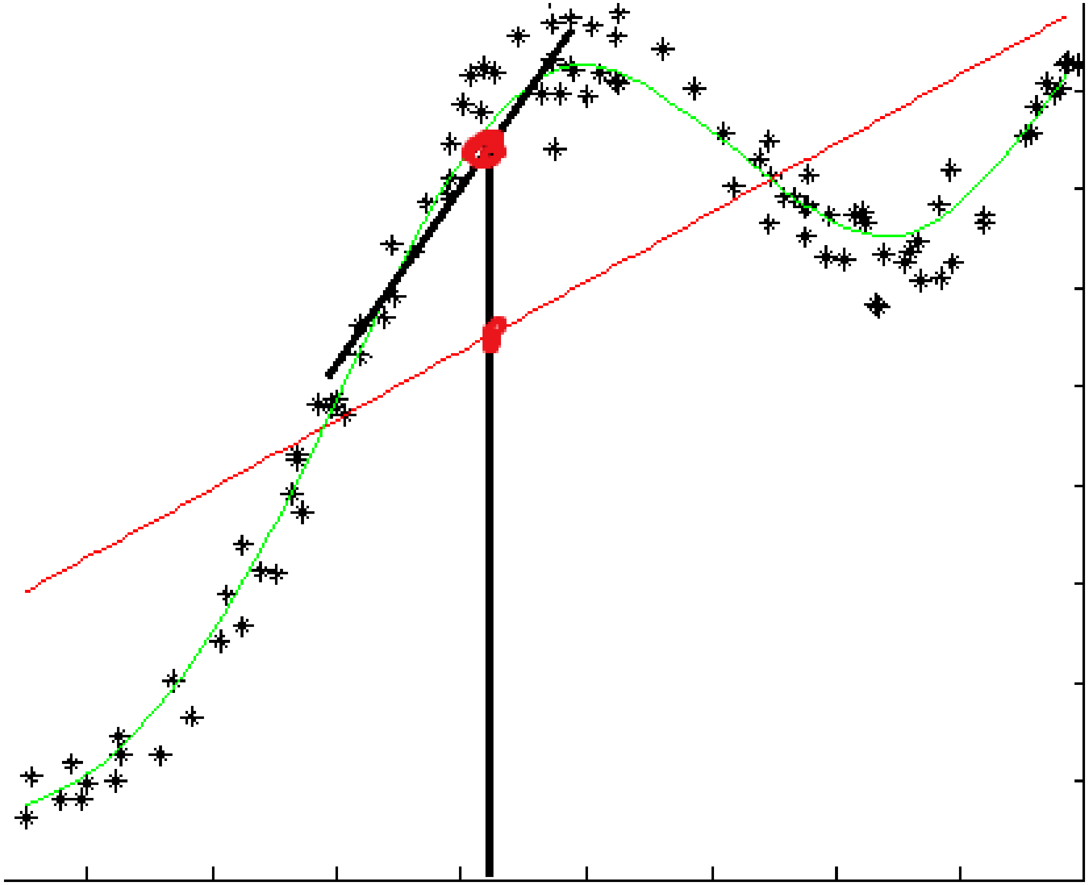
Coefficient of Determination

$$R^2 = 1 - \frac{RSS}{TSS} = 1 - \frac{\sum_{i=1}^{m}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{m}(y_i - \bar{y})^2}$$

- □ 对于m个样本$(\vec{x}_1, y_1), (\vec{x}_2, y_2), \cdots, (\vec{x}_m, y_m)$
- □ 某模型的估计值为$(\vec{x}_1, \hat{y}_1), (\vec{x}_2, \hat{y}_2), \cdots, (\vec{x}_m, \hat{y}_m)$
- □ 计算样本的总平方和TSS(Total Sum of Squares)：$TSS = \sum_{i=1}^{m}(y_i - \bar{y})^2$
  - ■ 即样本伪方差的m倍 $Var(Y) = TSS / m$
- □ 计算残差平方和RSS(Residual Sum of Squares)：$RSS = \sum_{i=1}^{m}(\hat{y}_i - y_i)^2$
  - ■ 注：RSS即误差平方和SSE(Sum of Squares for Error)
- □ 定义$R^2 = 1 - RSS / TSS$
  - ■ R^{2}越大，拟合效果越好
  - ■ R^{2}的最优值为1；若模型预测为随机值，R^{2}有可能为负
  - ■ 若预测值恒为样本期望，R^{2}为0
- □ 亦可定义ESS(Explained Sum of Squares)：$ESS = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - \bar{y})^2$
  - ■ TSS = ESS + RSS
  - ■ ESS又称回归平方和SSR(Sum of Squares for Regression)

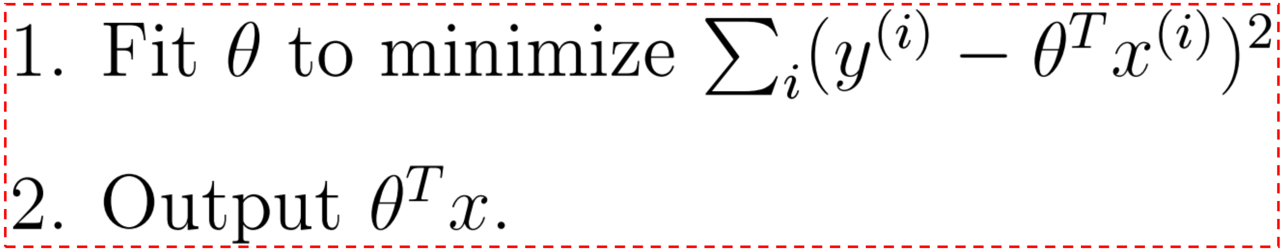
局部加权回归

- □ 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



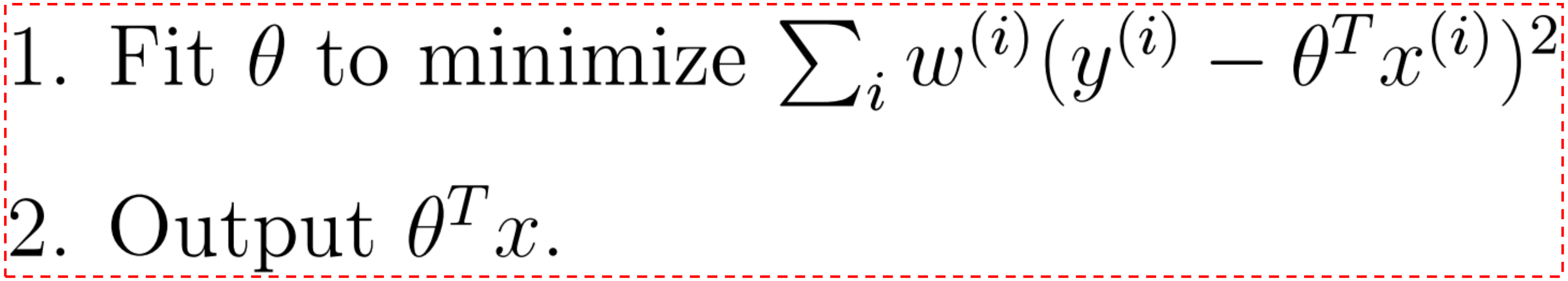
局部加权线性回归

□ 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):

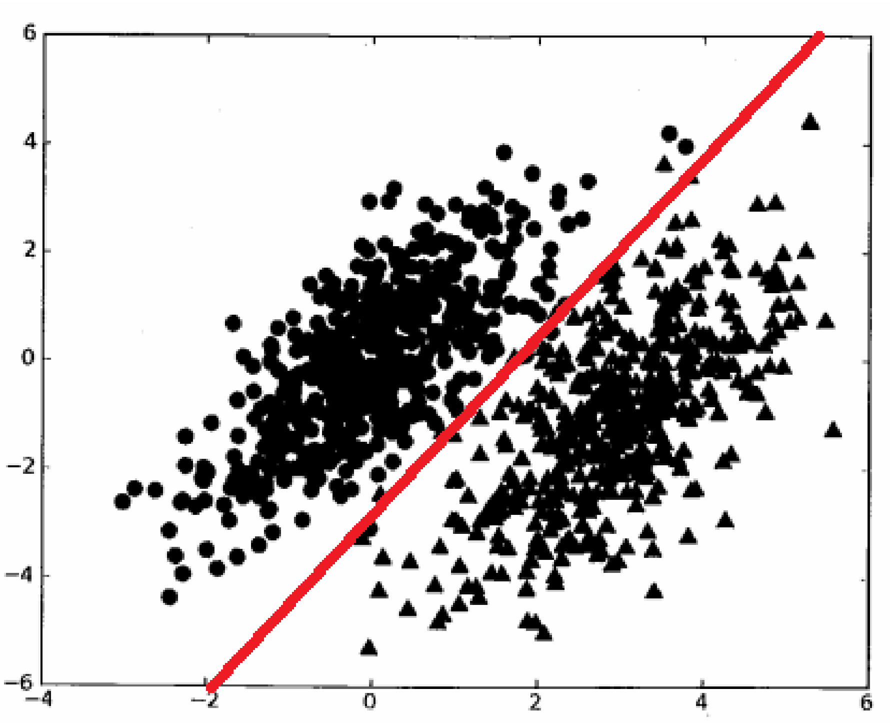
$$w^{(i)} = \exp\left(-\frac{(x^{(i)}-x)^2}{2\tau^2}\right)$$

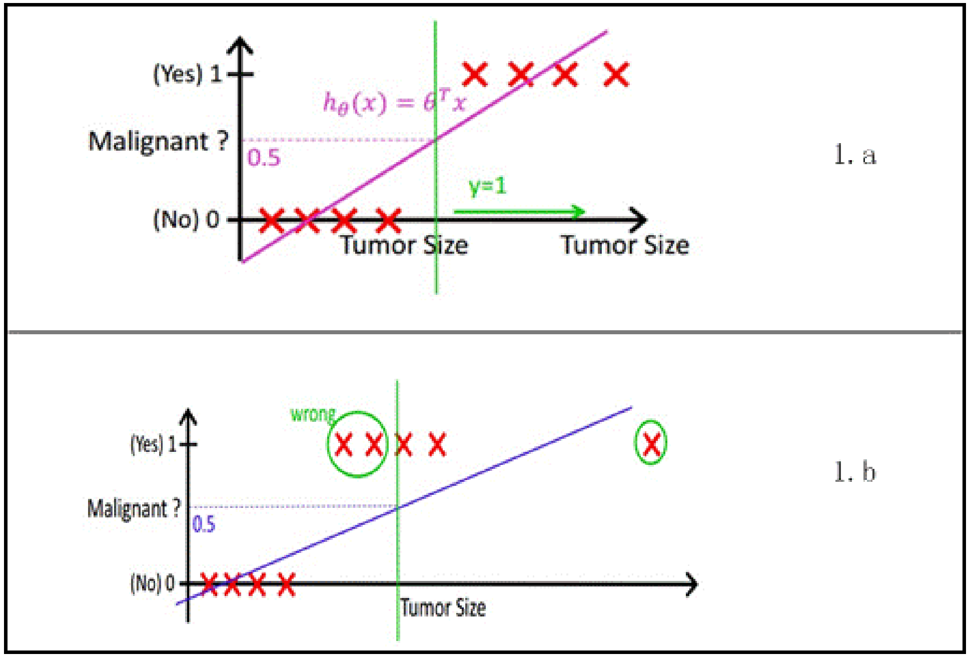
- τ称为带宽，它控制着训练样本随着与$x^{(i)}$距离的衰减速率。
- 多项式核函数

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

  - 在SVM章节继续核函数的讨论。

用回归解决分类问题，如何？

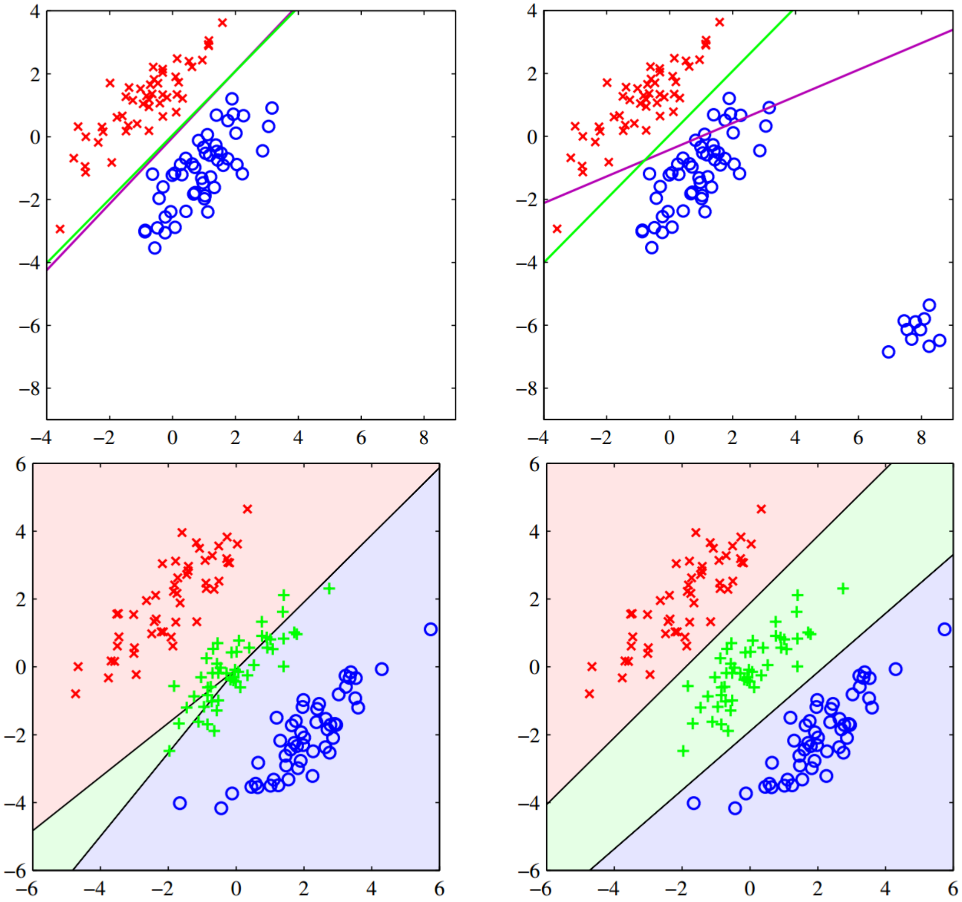




线性回归-Logistic回归

- 紫色：
  - 线性回归
- 绿色：
  - Logistic回归

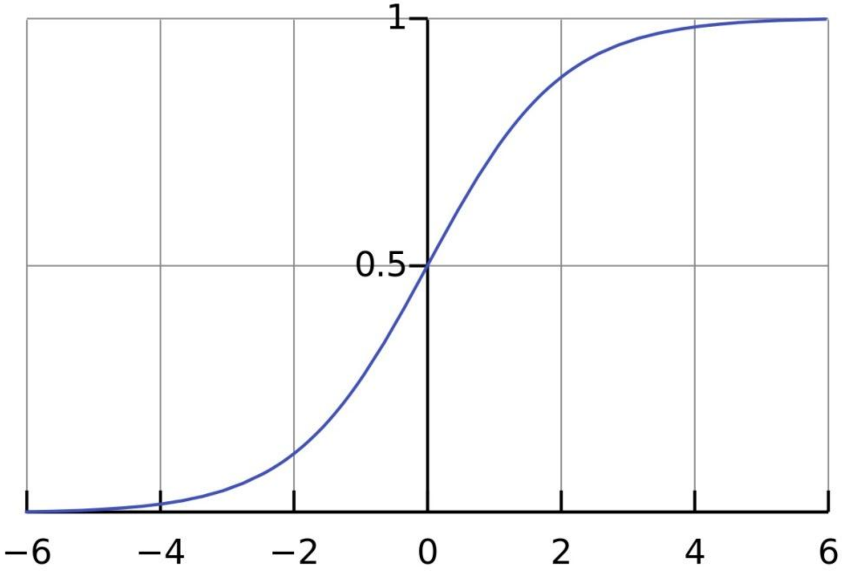
- 左侧：
  - 线性回归
- 右侧：
  - Softmax回归



Logistic回归

- □ Logistic/sigmoid函数

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



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

$$g'(x) = \left(\frac{1}{1+e^{-x}}\right)' = \frac{e^{-x}}{\left(1+e^{-x}\right)^2}$$

$$= \frac{1}{1+e^{-x}} \cdot \frac{e^{-x}}{1+e^{-x}} = \frac{1}{1+e^{-x}} \cdot \left(1 - \frac{1}{1+e^{-x}}\right)$$

$$= g(x) \cdot (1 - g(x))$$

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

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

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

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

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

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

Logistic回归的损失函数 $y_i \in \{0,1\}$

$$\hat{y}_i = \begin{cases} p_i & y_i = 1 \\ 1 - p_i & y_i = 0 \end{cases}$$

$$L(\theta) = \prod_{i=1}^{m} p_i^{y_i} (1 - p_i)^{1-y_i}$$

$$\Rightarrow l(\theta) = \sum_{i=1}^{m} \ln\left[p_i^{y_i} (1 - p_i)^{1-y_i}\right]$$

$$\xrightarrow{p_i = \frac{1}{1+e^{-f_i}}} l(\theta) = \sum_{i=1}^{m} \ln\left[\left(\frac{1}{1+e^{-f_i}}\right)^{y_i} \left(\frac{1}{1+e^{f_i}}\right)^{1-y_i}\right]$$

$$\therefore \ loss(y_i, \hat{y}_i) = -l(\theta)$$

$$= \sum_{i=1}^{m} \left[y_i \ln\left(1 + e^{-f_i}\right) + (1 - y_i) \ln\left(1 + e^{f_i}\right)\right]$$

Logistic回归的损失：$y_i \in \{-1,1\}$

$$L(\theta)=\prod_{i=1}^{m} p_i^{(y_i+1)/2}(1-p_i)^{-(y_i-1)/2} \Rightarrow l(\theta)=\sum_{i=1}^{m}\ln\left[p_i^{(y_i+1)/2}(1-p_i)^{-(y_i-1)/2}\right]$$

$$\xrightarrow{p_i=\frac{1}{1+e^{-f_i}}} l(\theta)=\sum_{i=1}^{m}\ln\left[\left(\frac{1}{1+e^{-f_i}}\right)^{(y_i+1)/2}\left(\frac{1}{1+e^{f_i}}\right)^{-(y_i-1)/2}\right]$$

$$\therefore\ loss(y_i,\hat{y}_i)=-l(\theta)$$

$$=\sum_{i=1}^{m}\left[\frac{1}{2}(y_i+1)\ln\left(1+e^{-f_i}\right)-\frac{1}{2}(y_i-1)\ln\left(1+e^{f_i}\right)\right]$$

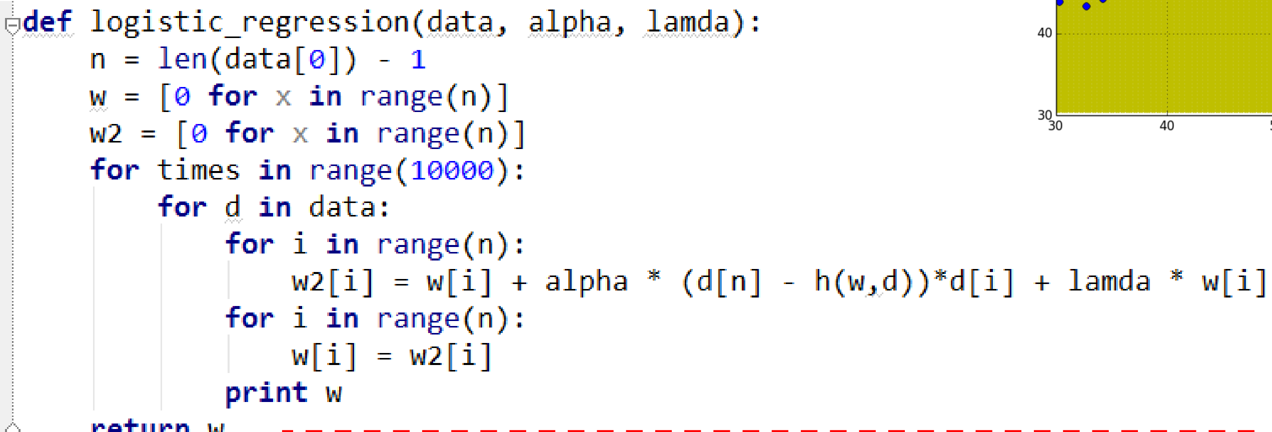
$$=\begin{cases}\sum_{i=1}^{m}\left[\ln\left(1+e^{-f_i}\right)\right] & y_i=1\\ \sum_{i=1}^{m}\left[\ln\left(1+e^{f_i}\right)\right] & y_i=-1\end{cases} \Rightarrow loss(y_i,\hat{y}_i)=\sum_{i=1}^{m}\left[\ln\left(1+e^{-y_i\cdot f_i}\right)\right]$$

$$y_i=\{-1,1\}$$

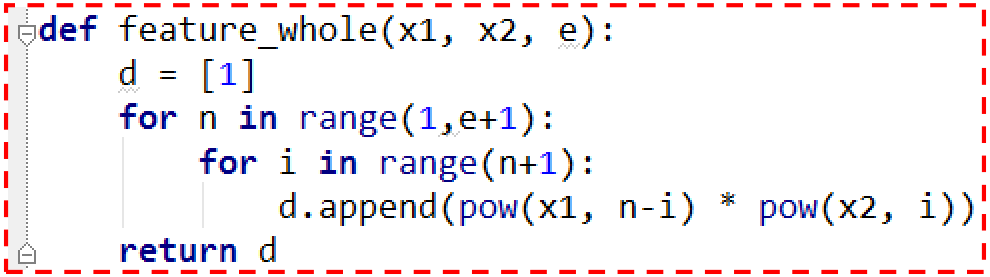
$$\hat{y}_i=\begin{cases}p_i & y_i=1\\ 1-p_i & y_i=-1\end{cases}$$

分类：Logistic回归

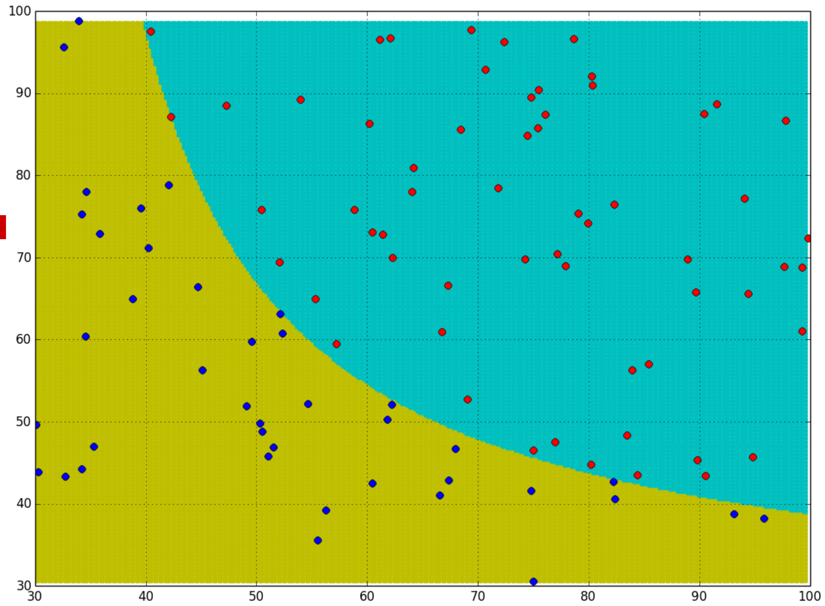
- □ 沿似然函数正梯度上升
- □ 维度提升

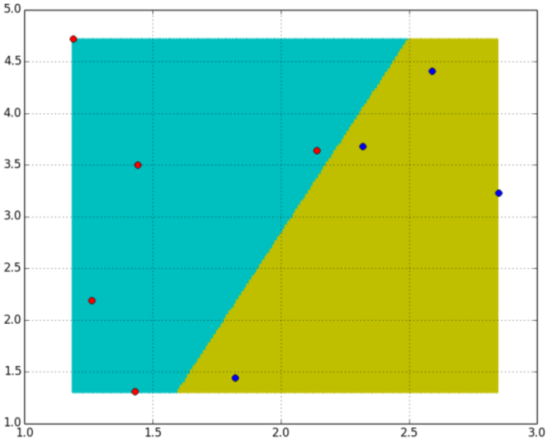


```
def logistic_regression(data, alpha, lamda):
    n = len(data[0]) - 1
    w = [0 for x in range(n)]
    w2 = [0 for x in range(n)]
    for times in range(10000):
        for d in data:
            for i in range(n):
                w2[i] = w[i] + alpha * (d[n] - h(w,d))*d[i] + lamda * w[i]
            for i in range(n):
                w[i] = w2[i]
            print w
    return w
```



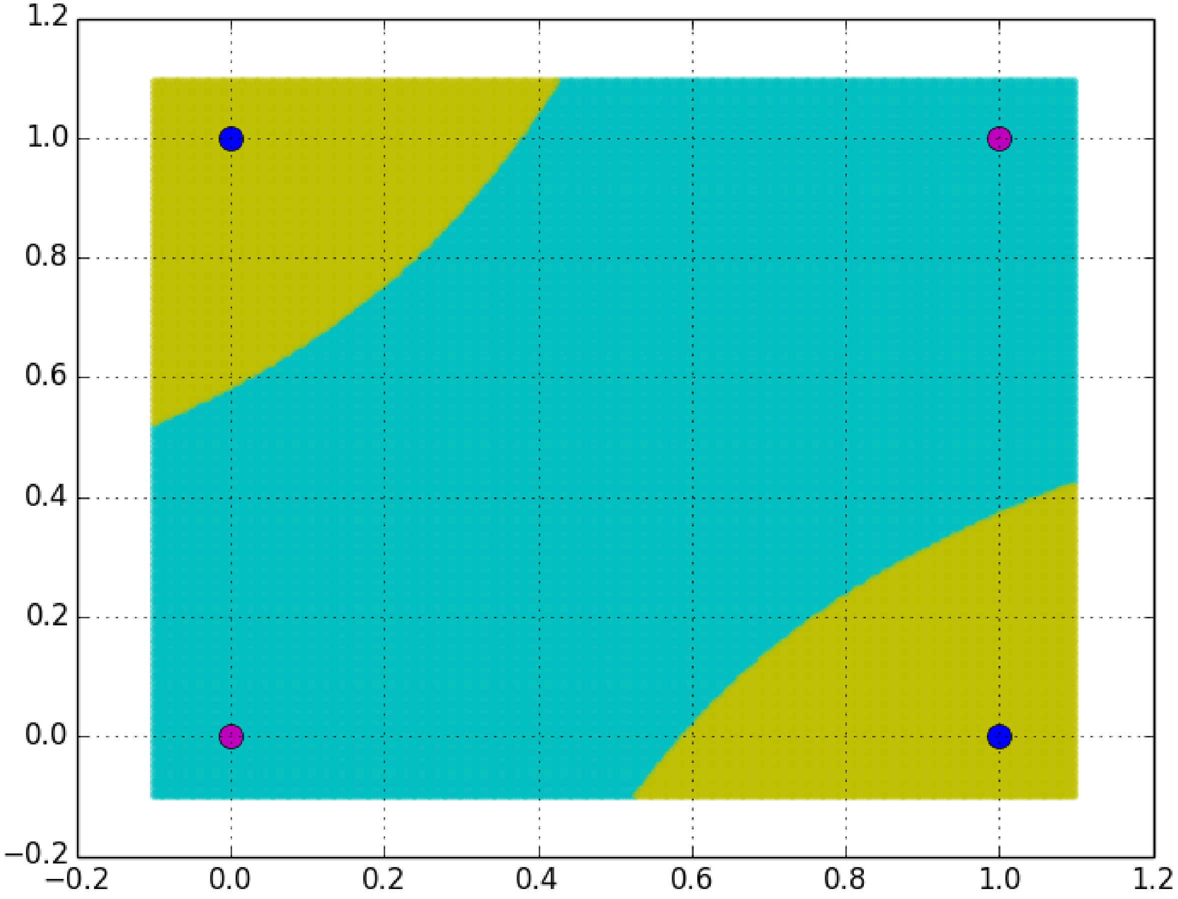
```
def feature_whole(x1, x2, e):
    d = [1]
    for n in range(1,e+1):
        for i in range(n+1):
            d.append(pow(x1, n-i) * pow(x2, i))
    return d
```





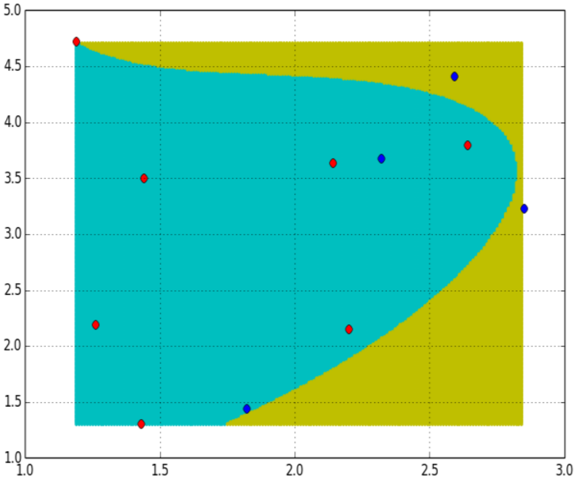
异或

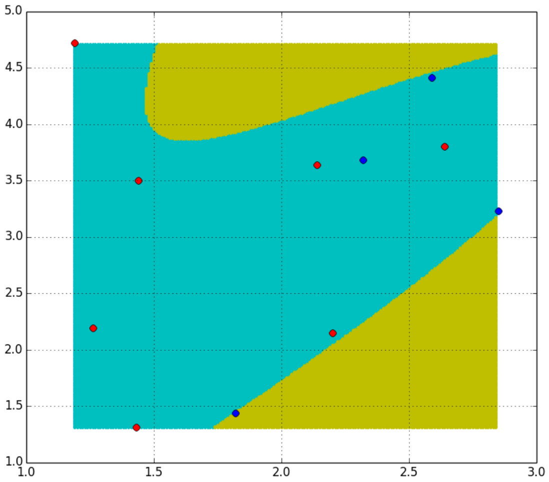
x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

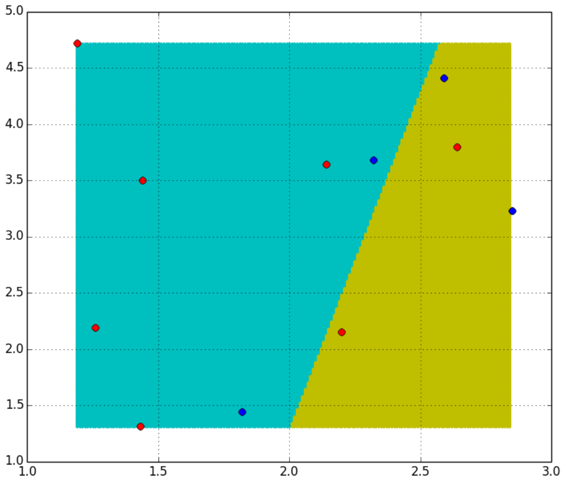


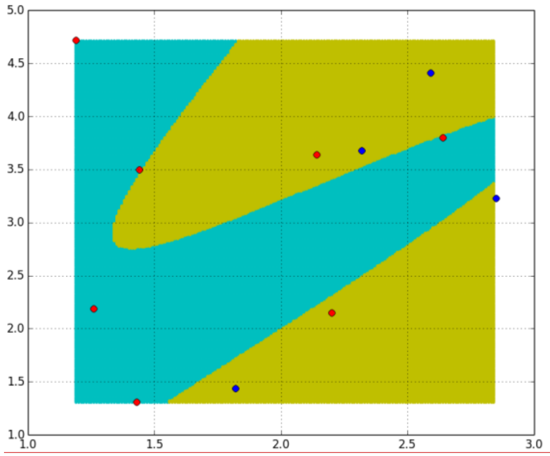
数据升维：“选取特征”

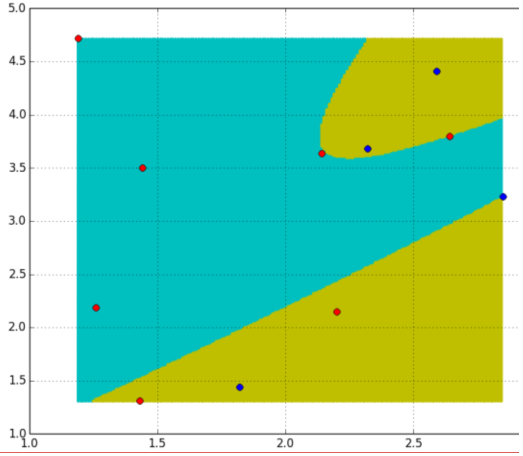
3	5	1
9	20	

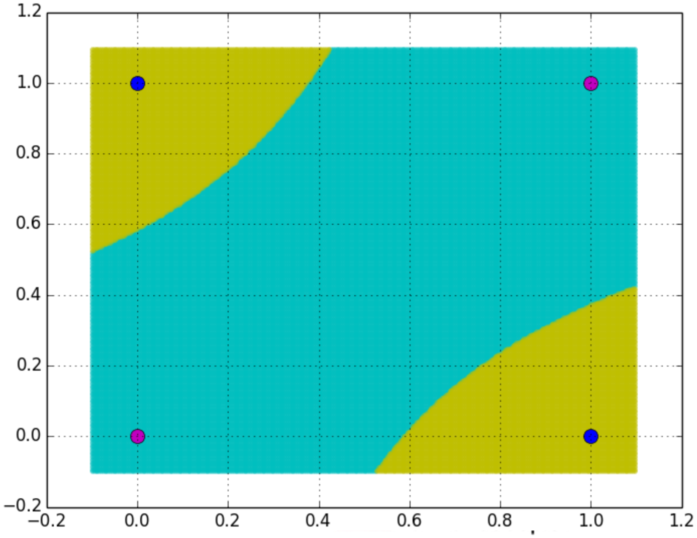












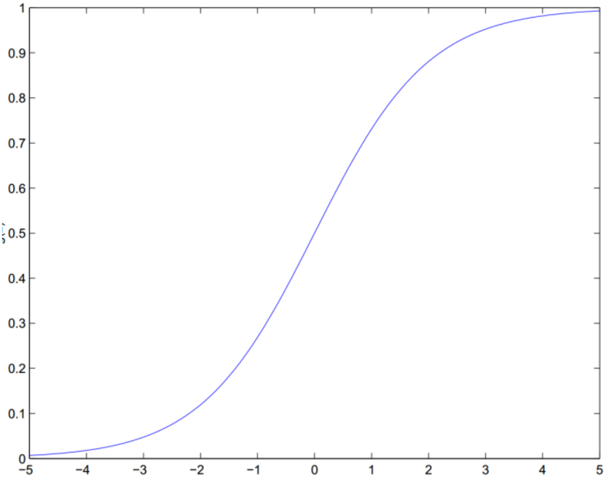
复习：指数族

- □ 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

- y不再只是正态分布，而是扩大为指数族中的任一分布；
- 变量x → g(x) → y
  - 连接函数g
- 连接函数g单调可导
  - 如Logistic回归中的 $g(z)=\dfrac{1}{1+e^{-z}}$
  - 拉伸变换：

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



Softmax回归

□ K分类，第k类的参数为 $\vec{\theta}_k$，组成二维矩阵 $\theta_{k\times n}$

□ 概率：$p(c=k\,|\,x;\theta)=\dfrac{\exp\left(\theta_k^T x\right)}{\sum_{l=1}^{K}\exp\left(\theta_l^T x\right)},\quad k=1,2,\cdots,K$

□ 似然函数：$$L(\theta)=\prod_{i=1}^{m}\prod_{k=1}^{K}p\left(c=k\,|\,x^{(i)};\theta\right)^{y_k^{(i)}}=\prod_{i=1}^{m}\prod_{k=1}^{K}\frac{\exp\left(\theta_k^T x^{(i)}\right)^{y_k^{(i)}}}{\sum_{l=1}^{K}\exp\left(\theta_l^T x^{(i)}\right)}$$

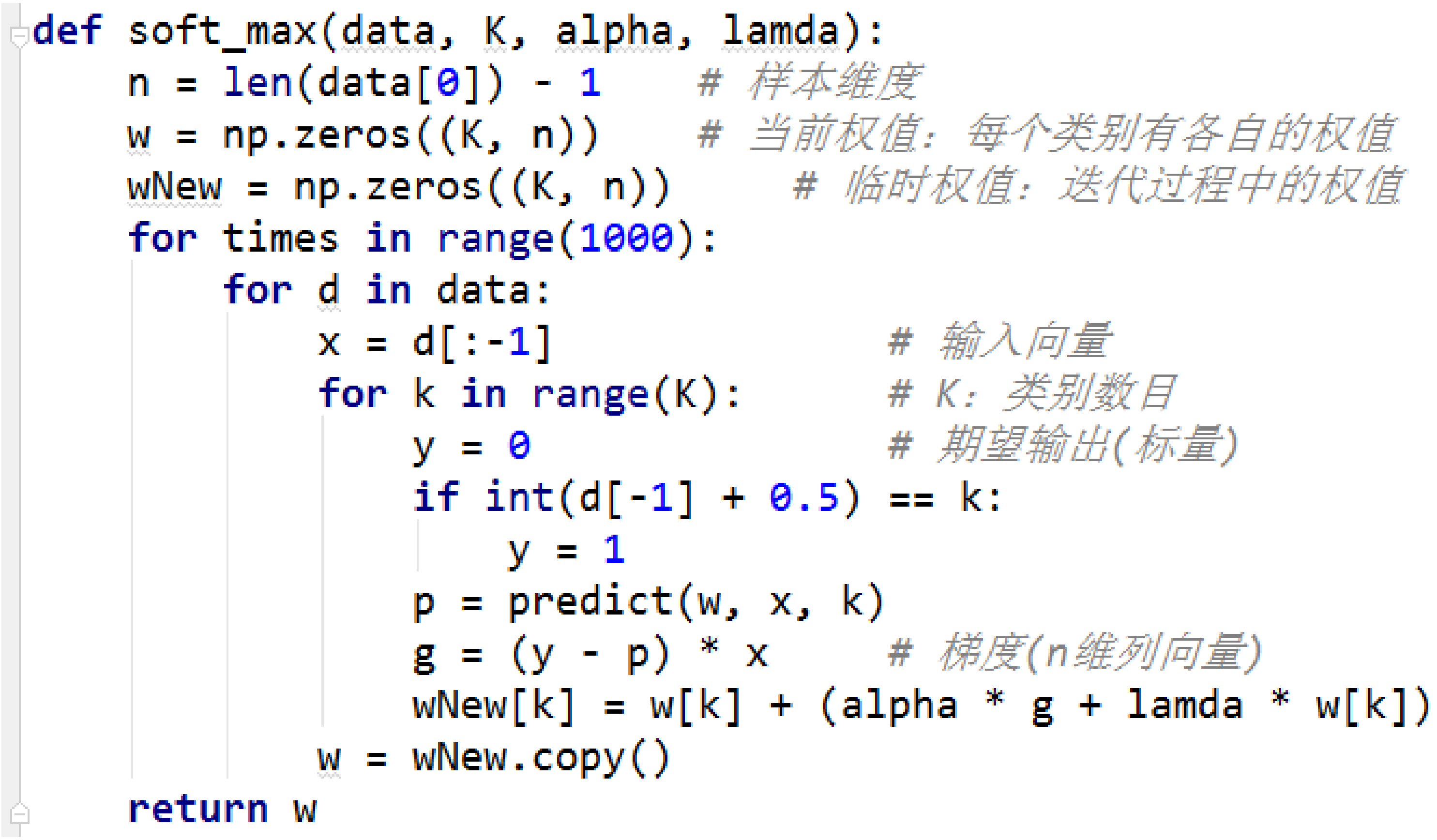
□ 对数似然：$$J_m(\theta)=\ln L(\theta)=\sum_{i=1}^{m}\sum_{k=1}^{K}\left(y_k^{(i)}\cdot\theta_k^T x^{(i)}-\ln\sum_{l=1}^{K}\exp\left(\theta_l^T x^{(i)}\right)\right)$$

$$J(\theta)=\sum_{k=1}^{K}y_k\cdot\left(\theta_k^T x-\ln\sum_{l=1}^{K}\exp\left(\theta_l^T x\right)\right)$$

□ 随机梯度：

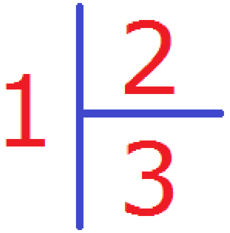
$$\frac{\partial J(\theta)}{\partial\theta_k}=\left(y_k-p\left(y_k\,|\,x;\theta\right)\right)\cdot x$$

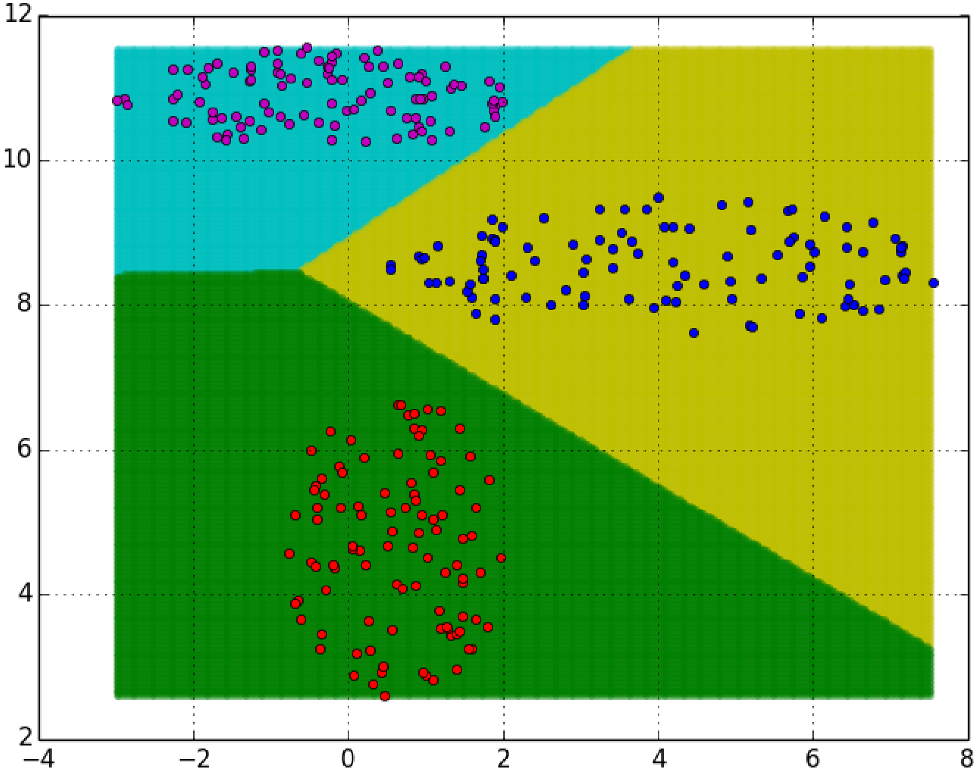
Code

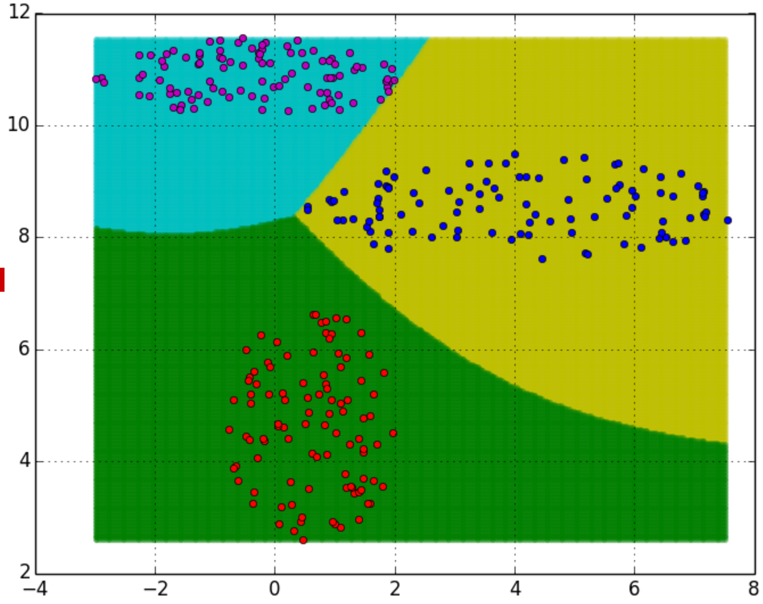


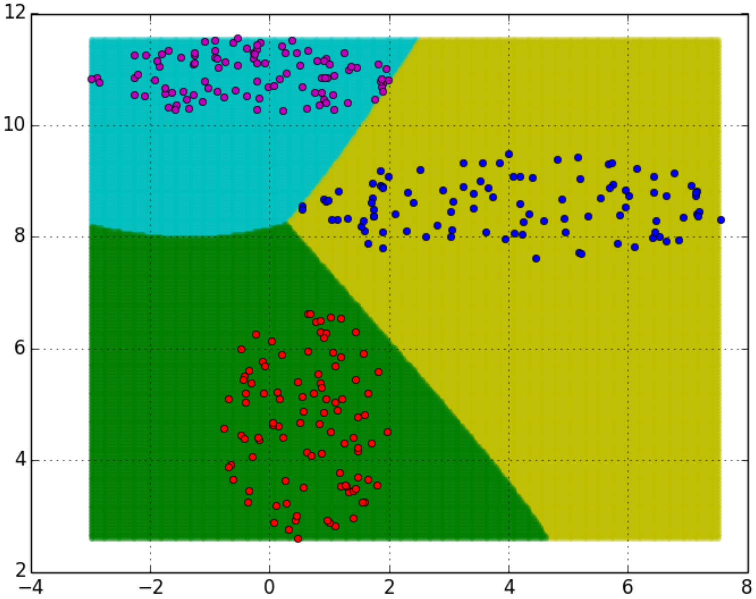
```
def soft_max(data, K, alpha, lamda):
    n = len(data[0]) - 1    # 样本维度
    w = np.zeros((K, n))    # 当前权值：每个类别有各自的权值
    wNew = np.zeros((K, n))     # 临时权值：迭代过程中的权值
    for times in range(1000):
        for d in data:
            x = d[:-1]              # 输入向量
            for k in range(K):      # K：类别数目
                y = 0               # 期望输出(标量)
                if int(d[-1] + 0.5) == k:
                    y = 1
                p = predict(w, x, k)
                g = (y - p) * x     # 梯度(n维列向量)
                wNew[k] = w[k] + (alpha * g + lamda * w[k])
            w = wNew.copy()
    return w
```

Softmax分类

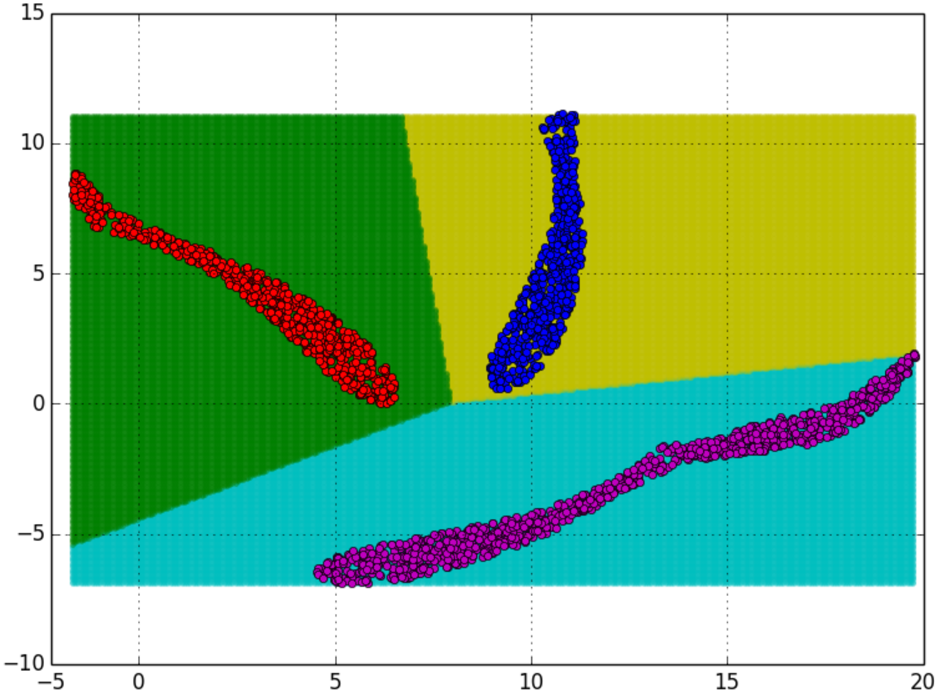


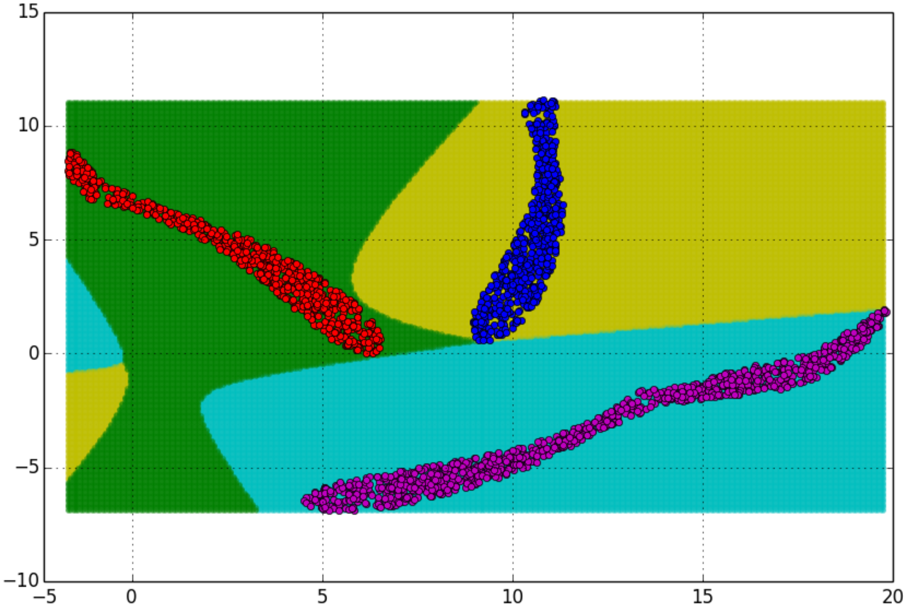


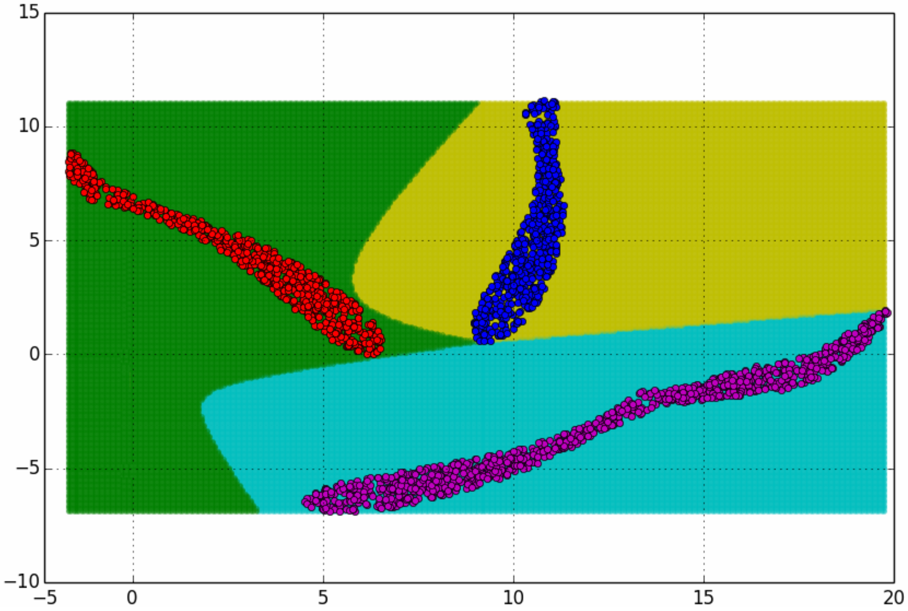




特征选择


15
10
5
0
−5
−10
−5
0
5
10
15
20


15
10
5
0
−5
−10
−5
0
5
10
15
20


15
10
5
0
−5
−10
−5
0
5
10
15
20

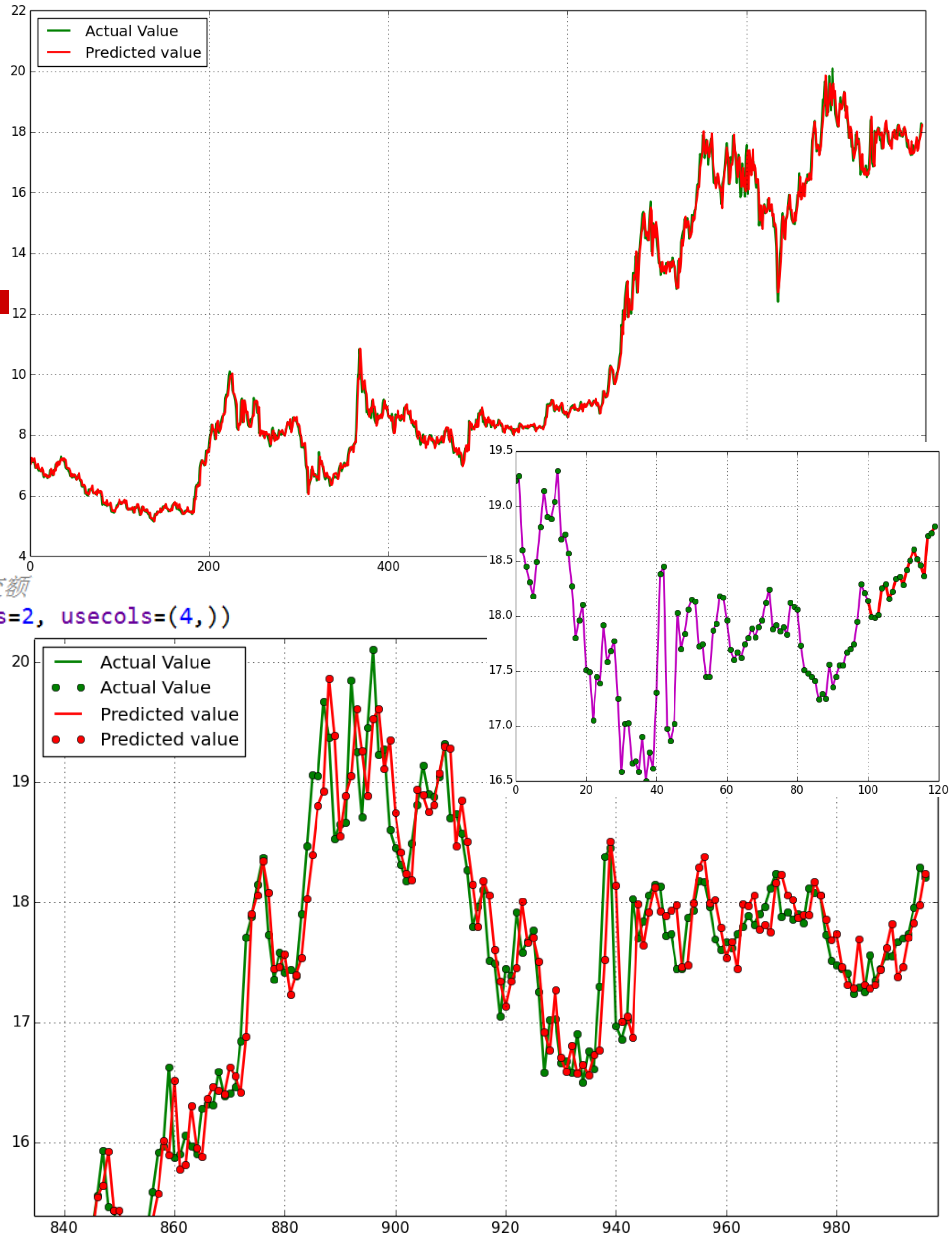
总结和思考

- □ Logistic/Softmax回归是实践中解决分类问题的最重要方法。
  - ■ 方法简单、容易实现、效果良好、易于解释
  - ■ 不止是分类：推荐系统
- □ 特征选择很重要，除了人工选择，还可以用其他机器学习方法，如随机森林、PCA、LDA等。
- □ 梯度下降算法是参数优化的重要手段，尤其SGD。
  - ■ 适用于在线学习
  - ■ 跳出局部极小值
  - ■ 思考：计算可逆方阵的逆，可否使用梯度下降算法？

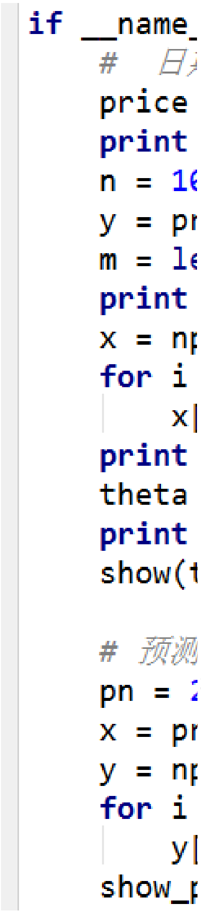
股价拟合

- □ 方法：自回归
- □ 参数：100阶

```
if __name__ == "__main__":
    #   日期      开盘      最高      最低      收盘      成交量         成交额
    price = np.loadtxt('sh_600000.txt', delimiter='\t', skiprows=2, usecols=(4,))
    print "原始价格：\n", price
    n = 100             # 阶数
    y = price[n:]
    m = len(y)          # 样本个数
    print "预测价格：\n", y
    x = np.zeros((m, n+1))
    for i in range(m):
        x[i] = np.hstack((price[i:i+n], 1))
    print "自变量：\n", x
    theta = np.linalg.lstsq(x, y)[0]      # theta为回归系数
    print theta
    show(theta, x, y)

    # 预测
    pn = 20      # 预测未来多少天
    x = price[-n:]
    y = np.hstack((price[-n:], np.zeros(pn)))
    for i in range(pn):
        y[n+i] = np.dot(theta, np.hstack((y[i:n+i], 1)))
    show_predict(y, pn)
```

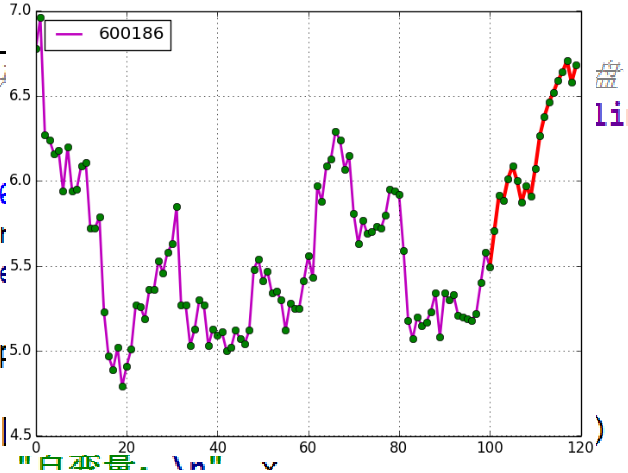


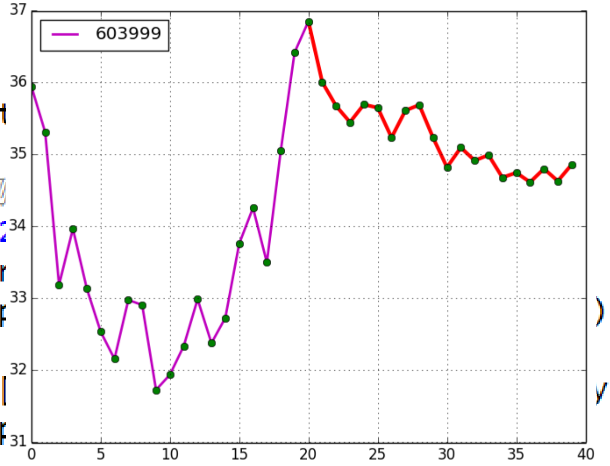
股价预测

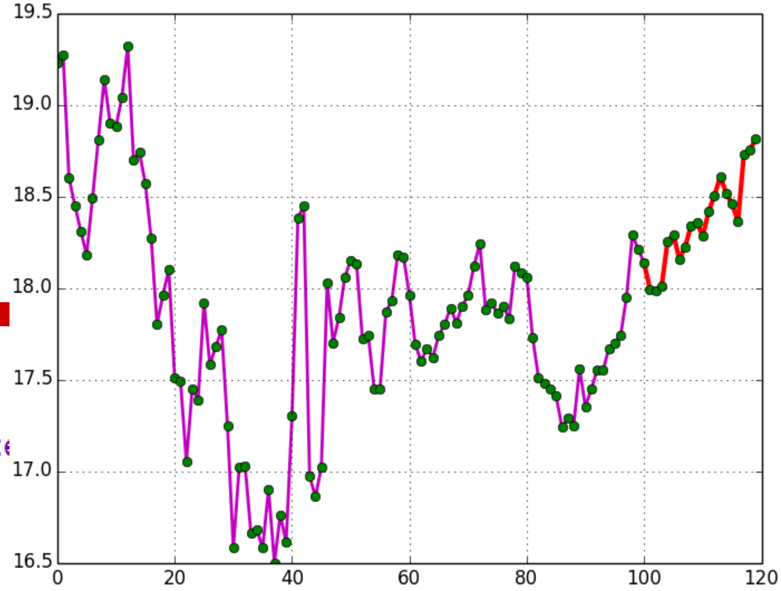


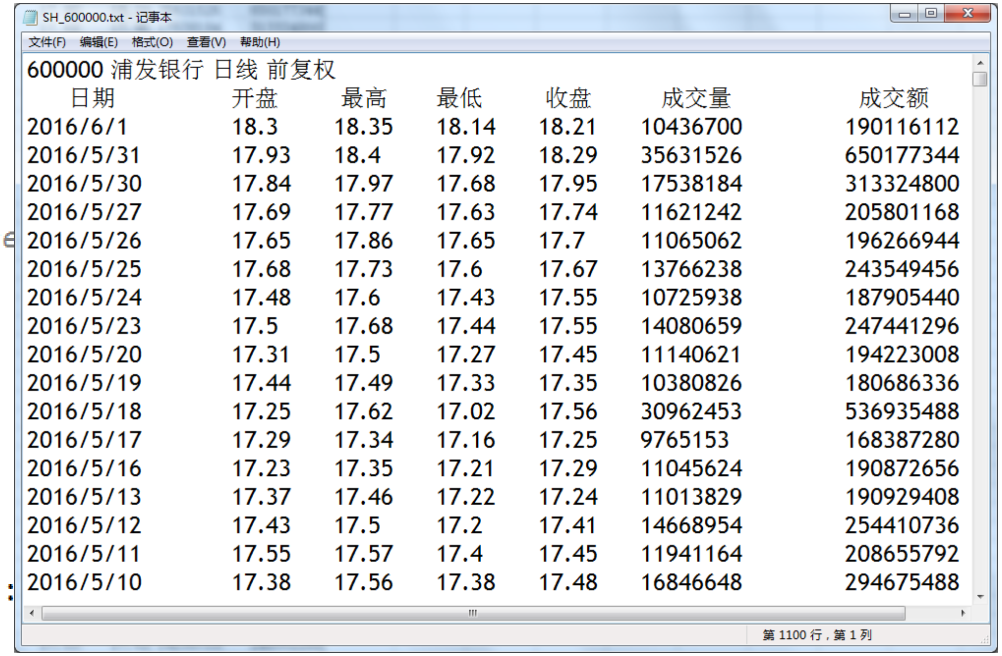
```
if __name_
    # 日
    price
    print
    n = 1(
    y = pr
    m = le
    print
    x = np
    for i
        x[
    print "自变量: \n", x
    theta
    print
    show(t

    # 预测
    pn = 2
    x = pr
    y = np
    for i
        y[
    show_p
```

```
盘
limit
)
the
)))
y[i:
```









600000 浦发银行 日线 前复权

日期	开盘	最高	最低	收盘	成交量	成交额
2016/6/1	18.3	18.35	18.14	18.21	10436700	190116112
2016/5/31	17.93	18.4	17.92	18.29	35631526	650177344
2016/5/30	17.84	17.97	17.68	17.95	17538184	313324800
2016/5/27	17.69	17.77	17.63	17.74	11621242	205801168
2016/5/26	17.65	17.86	17.65	17.7	11065062	196266944
2016/5/25	17.68	17.73	17.6	17.67	13766238	243549456
2016/5/24	17.48	17.6	17.43	17.55	10725938	187905440
2016/5/23	17.5	17.68	17.44	17.55	14080659	247441296
2016/5/20	17.31	17.5	17.27	17.45	11140621	194223008
2016/5/19	17.44	17.49	17.33	17.35	10380826	180686336
2016/5/18	17.25	17.62	17.02	17.56	30962453	536935488
2016/5/17	17.29	17.34	17.16	17.25	9765153	168387280
2016/5/16	17.23	17.35	17.21	17.29	11045624	190872656
2016/5/13	17.37	17.46	17.22	17.24	11013829	190929408
2016/5/12	17.43	17.5	17.2	17.41	14668954	254410736
2016/5/11	17.55	17.57	17.4	17.45	11941164	208655792
2016/5/10	17.38	17.56	17.38	17.48	16846648	294675488

作业

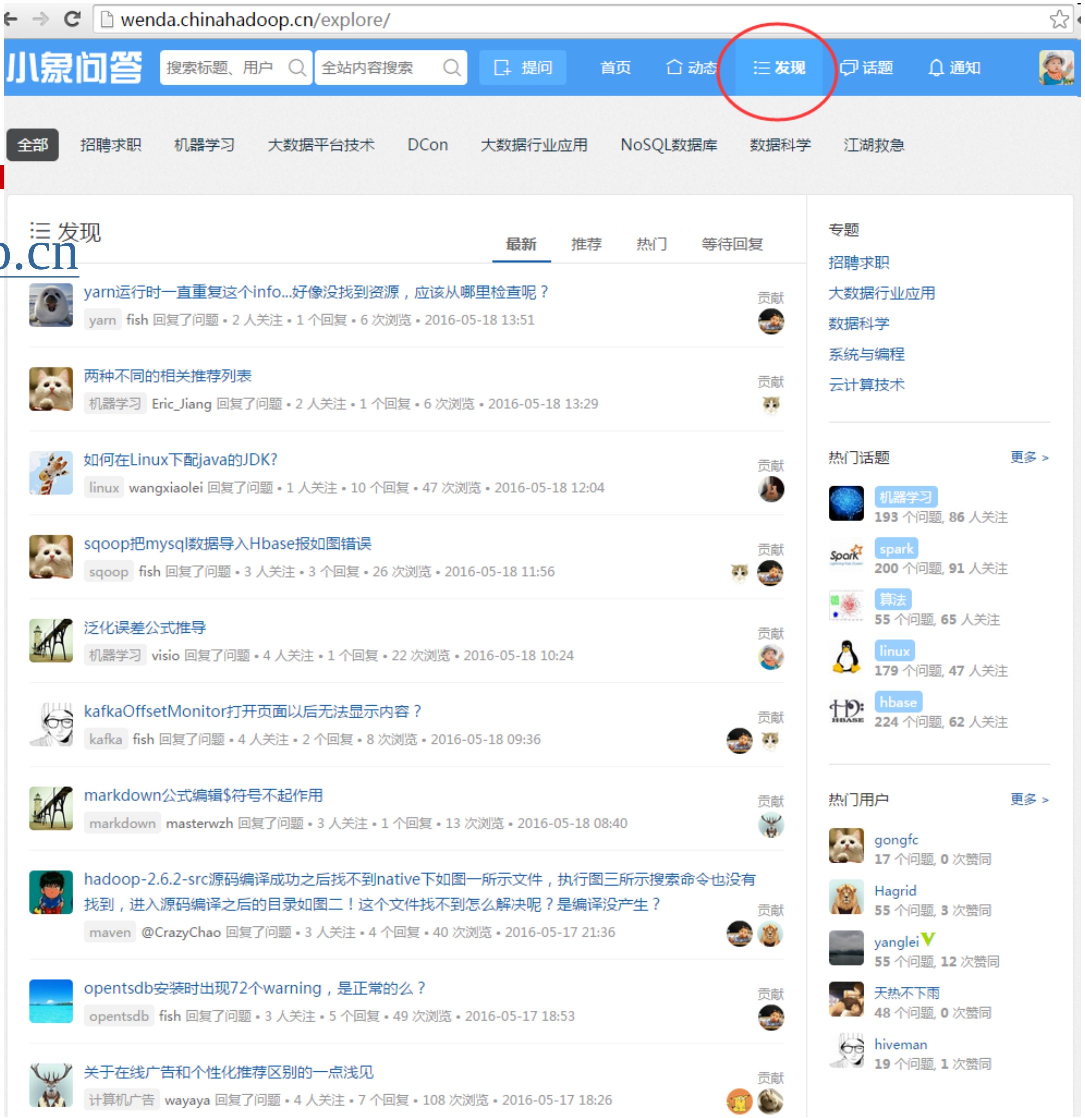
- □ 解释线性回归中使用误差平方和作为目标函数的原因。
- □ 请描述BGD和SGD的区别，并指出SGD的优势有哪些？
- □ 推导Logistic回归的梯度。

参考文献

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

我们在这里

- □ http://wenda.ChinaHadoop.cn
  - 视频/课程/社区
- □ 微博
  - @ChinaHadoop
  - @邹博_机器学习
- □ 微信公众号
  - 小象
  - 大数据分析挖掘



wenda.chinahadoop.cn/explore/

☰ 发现

yarn运行时一直重复这个info...好像没找到资源，应该从哪里检查呢？
yarn fish 回复了问题 • 2 人关注 • 1 个回复 • 6 次浏览 • 2016-05-18 13:51 贡献

两种不同的相关推荐列表
机器学习 Eric_Jiang 回复了问题 • 2 人关注 • 1 个回复 • 6 次浏览 • 2016-05-18 13:29 贡献

如何在Linux下配java的JDK?
linux wangxiaolei 回复了问题 • 1 人关注 • 10 个回复 • 47 次浏览 • 2016-05-18 12:04 贡献

sqoop把mysql数据导入Hbase报如图错误
sqoop fish 回复了问题 • 3 人关注 • 3 个回复 • 26 次浏览 • 2016-05-18 11:56 贡献

泛化误差公式推导
机器学习 visio 回复了问题 • 4 人关注 • 1 个回复 • 22 次浏览 • 2016-05-18 10:24 贡献

kafkaOffsetMonitor打开页面以后无法显示内容？
kafka fish 回复了问题 • 4 人关注 • 2 个回复 • 8 次浏览 • 2016-05-18 09:36 贡献

markdown公式编辑$符号不起作用
markdown masterwzh 回复了问题 • 3 人关注 • 1 个回复 • 13 次浏览 • 2016-05-18 08:40 贡献

hadoop-2.6.2-src源码编译成功之后找不到native下如图一所示文件，执行图三所示搜索命令也没有找到，进入源码编译之后的目录如图二！这个文件找不到怎么解决呢？是编译没产生？
maven @CrazyChao 回复了问题 • 3 人关注 • 4 个回复 • 40 次浏览 • 2016-05-17 21:36 贡献

opentsdb安装时出现72个warning，是正常的么？
opentsdb fish 回复了问题 • 3 人关注 • 5 个回复 • 49 次浏览 • 2016-05-17 18:53 贡献

关于在线广告和个性化推荐区别的一点浅见
计算机广告 wayaya 回复了问题 • 4 人关注 • 7 个回复 • 108 次浏览 • 2016-05-17 18:26 贡献

专题

- 招聘求职
- 大数据行业应用
- 数据科学
- 系统与编程
- 云计算技术

热门话题 更多 >

- 机器学习 193 个问题, 86 人关注
- spark 200 个问题, 91 人关注
- 算法 55 个问题, 65 人关注
- linux 179 个问题, 47 人关注
- hbase 224 个问题, 62 人关注

热门用户 更多 >

- gongfc 17 个问题, 0 次赞同
- Hagrid 55 个问题, 3 次赞同
- yanglei 55 个问题, 12 次赞同
- 天热不下雨 48 个问题, 0 次赞同
- hiveman 19 个问题, 1 次赞同

感谢大家！

恳请大家批评指正！