梯度下降和拟牛顿

3月机器学习在线班 邹博

2015年3月21日

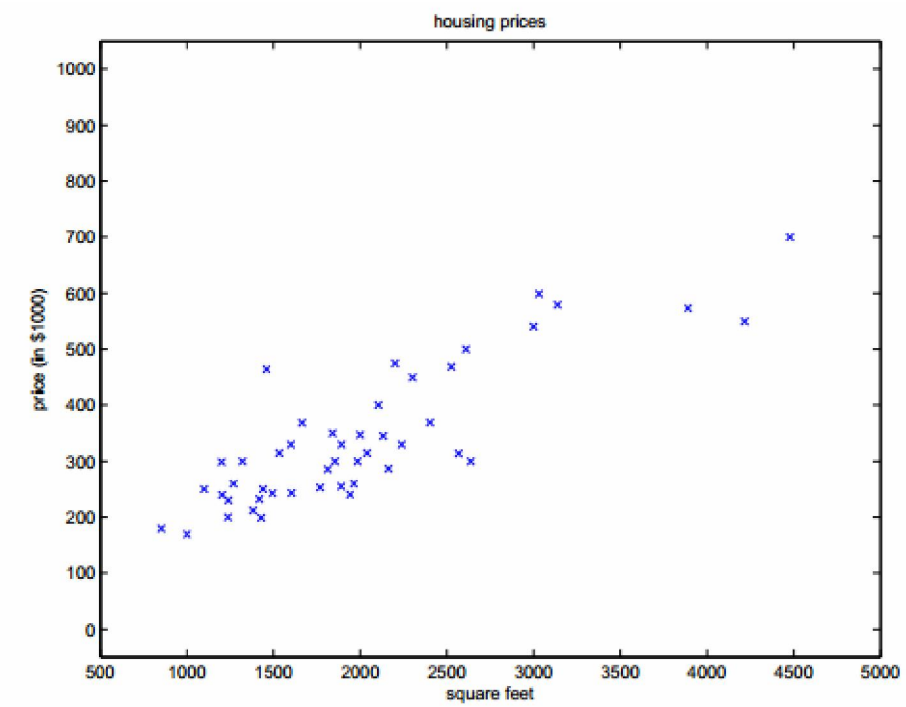
预备题目

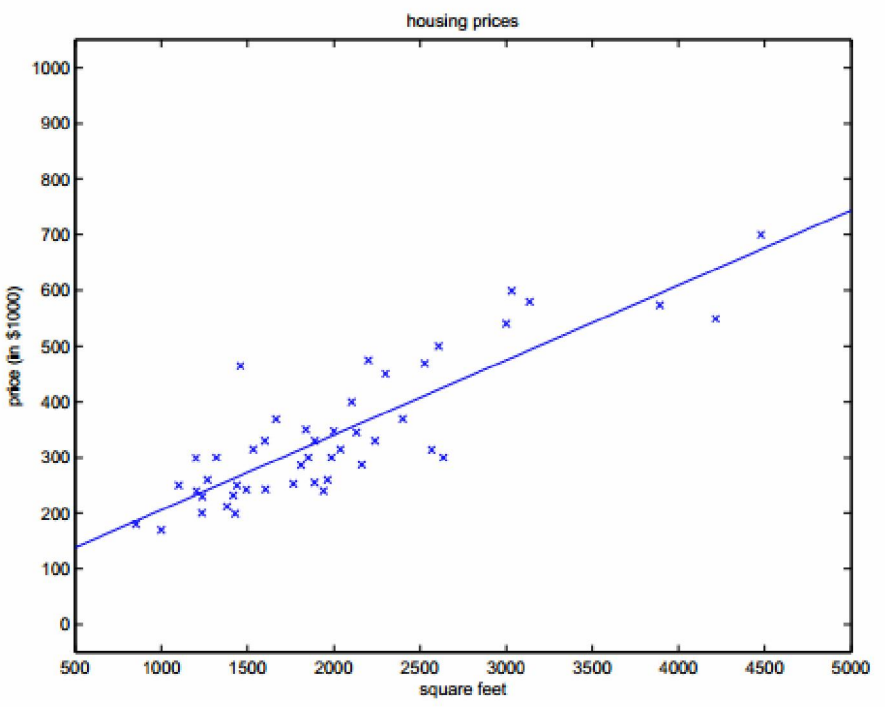
- 已知二次函数的一个点函数值和导数值，以及另外一个点的函数值，如果确定该函数的解析式？
  - 即：二次函数f(x)，已知f(a)，f'(a)，f(b)，求f(x)
  - 特殊的，若a=0，题目变成：
  - 对于二次函数f(x)，已知f(0)，f'(0)，f(a)，求f(x)

$$f(x)=\frac{f(a)-f'(0)a-f(0)}{a^2}x^2+f'(0)a+f(0)$$

从线性回归谈起

- $y=ax+b=ax_1+bx_0$，其中，$x_0 \equiv 1$





自变量扩展到多维

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

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

建立目标函数：最小二乘法

- 标记的含义：
  - 一共有m个样本，其中，第i个样本记做：
  - 一般的说，每个样本的自变量部分都是n维的，形成n维列向量；因此，每个 都是向量而不是数值。
    - 如：y=ax+b的例子中，样本是2维的，第一维是x1，第二维是齐次项x0恒为1
  - 样本的标记部分记做y，一般而言，是数值。
- 一个比较“符合常理”的误差函数为：

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

- 已经论证：最小二乘建立的目标函数，是在高斯噪声的假设下，利用极大似然估计的方法建立的。

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

- 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)无法继续减少或者达到循环上界次数，退出。

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

  - α：学习率、步长

线性回归目标函数的梯度方向计算

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

问题似乎完美解决

- 算法描述+凸函数极值

Repeat until convergence {

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

}

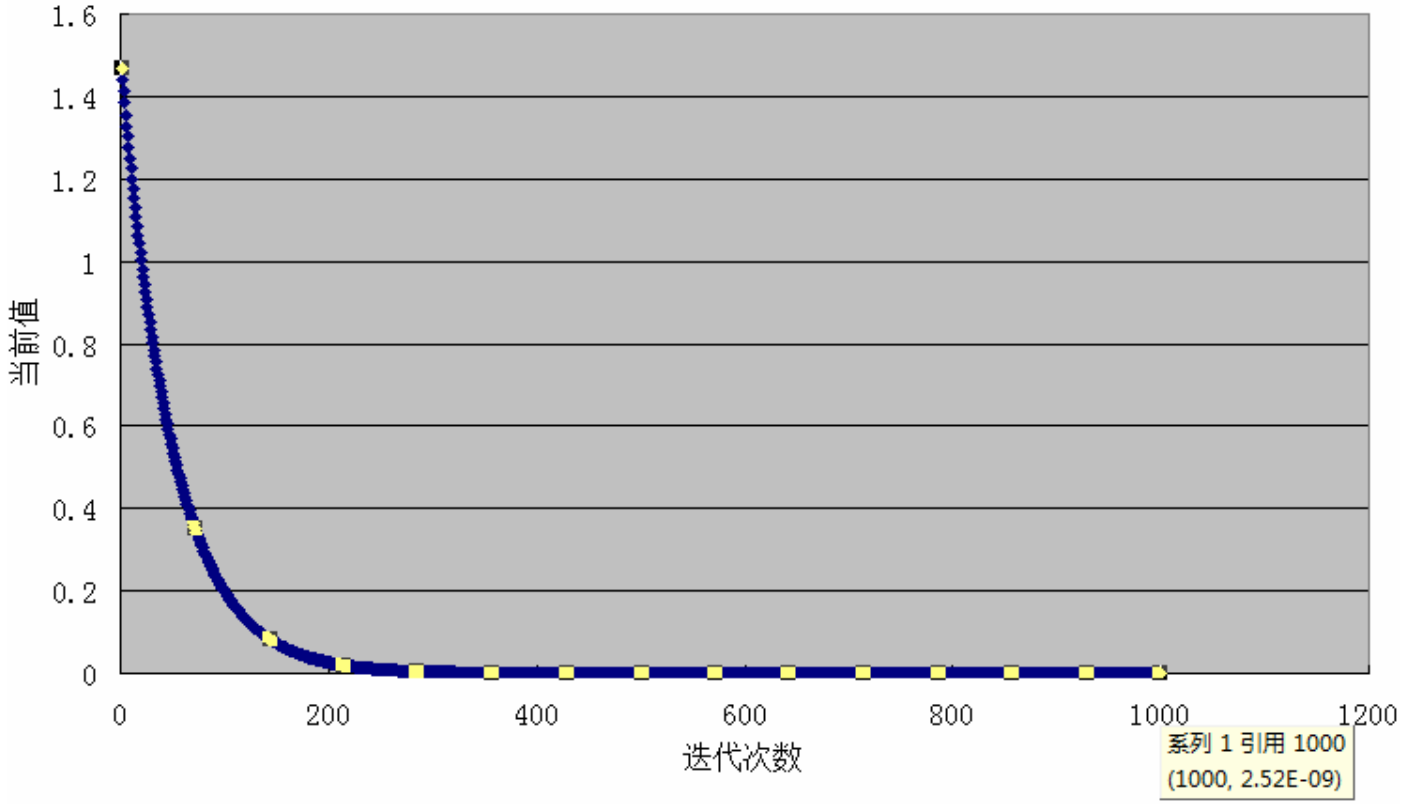
gradient descent. Note that, while gradient descent can be susceptible to local minima in general, the optimization problem we have posed here for linear regression has only one global, and no other local, optima; thus gradient descent always converges (assuming the learning rate α is not too large)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

思考

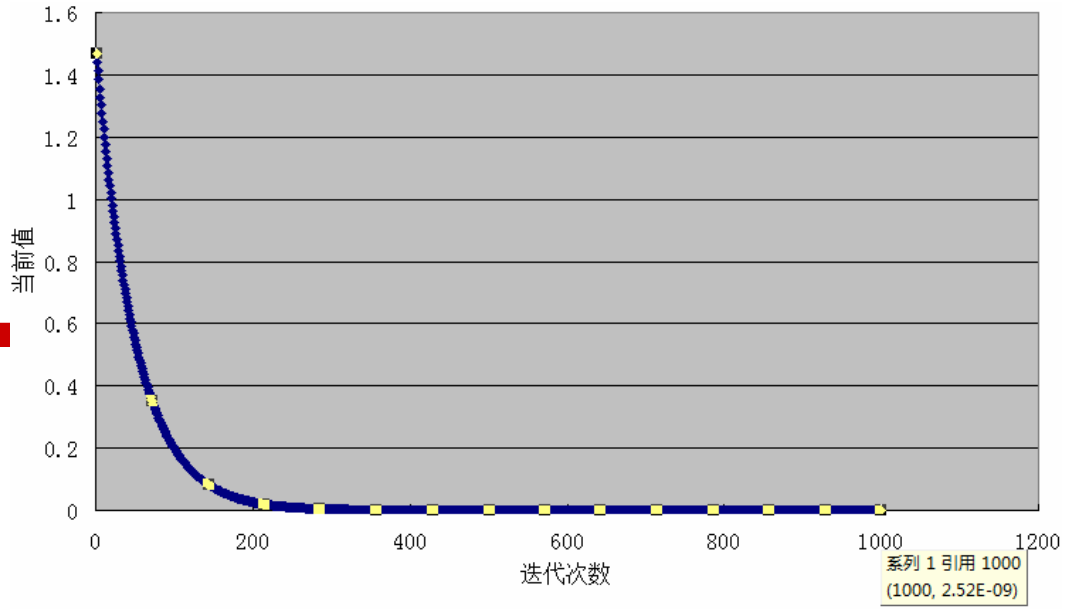
- 学习率α如何确定
  - 使用固定学习率还是变化学习率
  - 学习率设置多大比较好？
- 下降方向
  - 处理梯度方向，其他方向是否可以？
  - 可行方向和梯度方向有何关系？

实验：固定学习率的梯度下降

- $y=x^2$，初值取x=1.5，学习率使用0.01



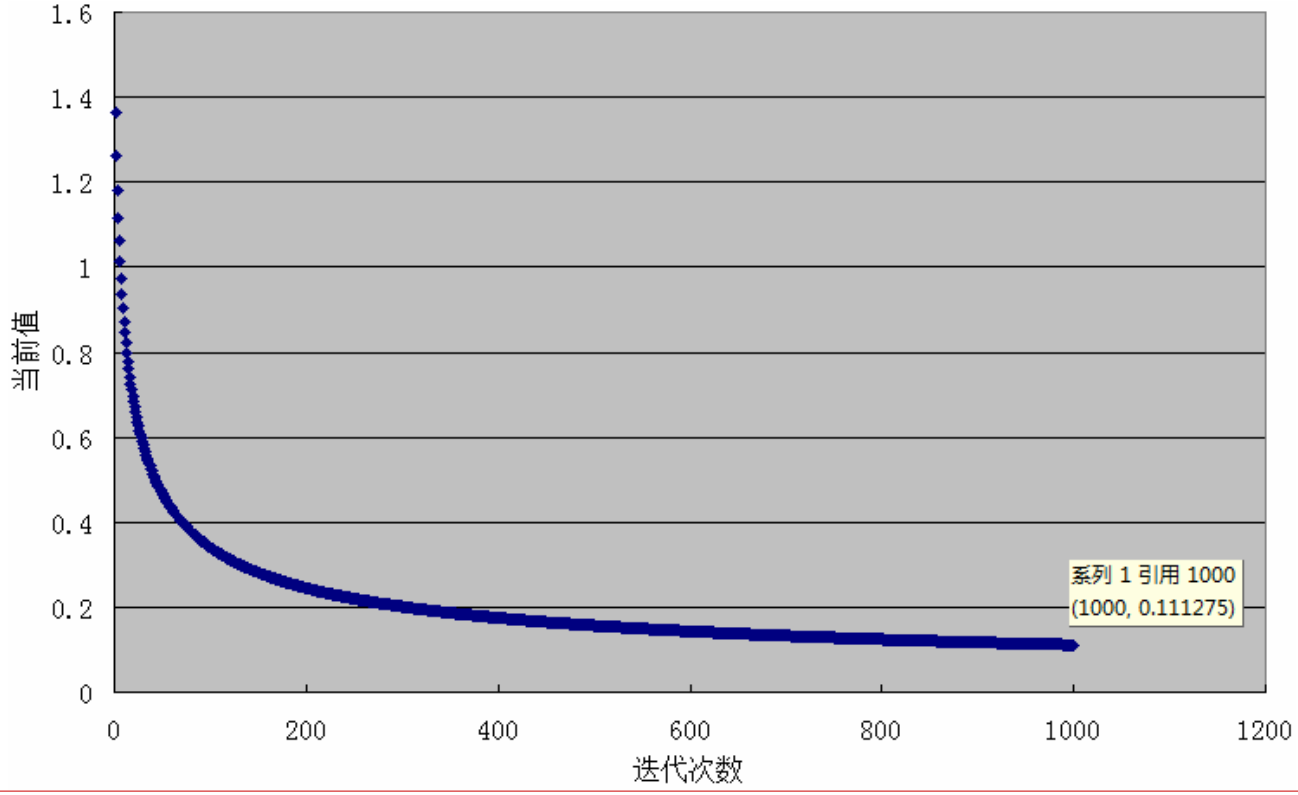
固定学习率



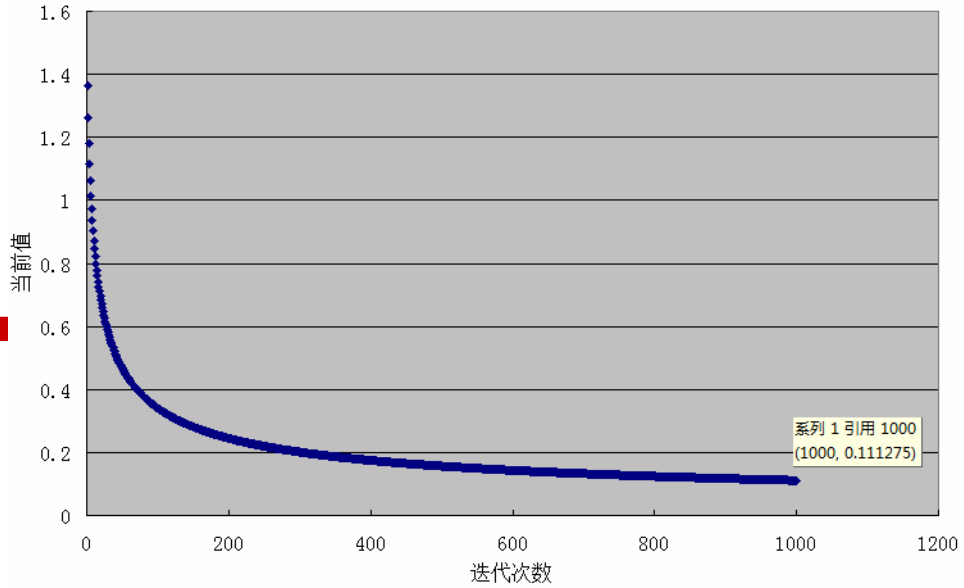
- 分析：
- 效果还不错
- 经过200次迭代，x=0.0258543；
- 经过1000次迭代，$x=2.52445\times10^{-9}$

实验2：固定学习率的梯度下降

- $y=x^4$，初值取x=1.5，学习率使用0.01



固定学习率



- 分析：
- 效果不理想
- 经过200次迭代，x=0.24436；
- 经过1000次迭代，x=0.111275

附：固定学习率实验的C代码

```
int _tmain(int argc, _TCHAR* argv[])
{
    double x = 1.5;
    double d;                //一阶导
    double a = 0.01;         //学习率
    for(int i = 0; i < 1000; i++)
    {
        d = g(x);
        x -= d * a;
        cout << i << '\t' << a << '\t' << x << '\n';
    }
    return 0;
}
```

优化学习率

- 分析“学习率α”在f(x)中的意义
- 调整学习率：
  - 在斜率(方向导数)大的地方，使用小学习率
  - 在斜率(方向导数)小的地方，使用大学习率
- 如何构造学习率α

梯度下降的运行过程分析

- x_k=a，沿着负梯度方向，移动到x_{k+1}=b，有：

$$b = a - \alpha \nabla F(a) \Rightarrow f(a) \geq f(b)$$

- 从x0为出发点，每次沿着当前函数梯度反方向移动一定距离αk，得到序列：

$$x_0, x_1, \cdots, x_n$$

- 对应的各点函数值序列之间的关系为：

$$f(x_0) \geq f(x_1) \geq f(x_2) \geq \cdots \geq f(x_n)$$

- 当n达到一定值时，函数f(x)收敛到局部最小值

视角转换

- 记当前点为$\mathbf{x_k}$，当前搜索方向为$\mathbf{d_k}$(如：负梯度方向)，因为学习率α是待考察的对象，因此，将下列函数$f(\mathbf{x_k}+\alpha\,\mathbf{d_k})$看做是关于$\alpha$的函数$h(\alpha)$。

$$h(\alpha)=f(x_k+\alpha d_k),\quad \alpha>0$$

- 当$\alpha=0$时，$h(0)=f(\mathbf{x_k})$
- 导数 $\nabla h(\alpha)=\nabla f(x_k+\alpha d_k)^T d_k$

学习率α的计算标准

- 因为梯度下降是寻找f(x)的最小值，那么，在$\mathbf{x_k}$和$\mathbf{d_k}$给定的前提下，即寻找函数$f(\mathbf{x_k}+\alpha\,\mathbf{d_k})$的最小值。即：

$$\alpha = \arg\min_{\alpha>0} h(\alpha) = \arg\min_{\alpha>0} f\left(x_k + \alpha d_k\right)$$

- 进一步，如果h(α)可导，局部最小值处的α满足：

$$h'(\alpha) = \nabla f\left(x_k + \alpha d_k\right)^T d_k = 0$$

学习率函数导数的分析 $h'(\alpha)=\nabla f(x_k+\alpha d_k)^T d_k=0$

- 将α=0带入：

$$h'(0)=\nabla f(x_k+0*d_k)^T d_k=\nabla f(x_k)^T d_k$$

- 下降方向$\mathbf{d_k}$可以选负梯度方向 $d_k=-\nabla f(x_k)$
  - 或者选择与负梯度夹角小于90°的某方向(后面会继续阐述搜索方向问题)
- 从而： $h'(0)<0$
- 如果能够找到足够大的α，使得 $h'(\hat{\alpha})>0$
- 则必存在某α，使得 $h'(\alpha^*)=0$
- α^*即为要寻找的学习率。

线性搜索(Line Search)

- 最简单的处理方式
  - 二分线性搜索(Bisection Line Search)
  - 不断将区间[α1, α2]分成两半，选择端点异号的一侧，知道区间足够小或者找到当前最优学习率。

回溯线性搜索(Backing Line Search)

- 基于Armijo准则计算搜素方向上的最大步长，其基本思想是沿着搜索方向移动一个较大的步长估计值，然后以迭代形式不断缩减步长，直到该步长使得函数值$f(\mathbf{x_k}+\alpha\,\mathbf{d_k})$相对与当前函数值$f(\mathbf{x_k})$的减小程度大于预设的期望值(即满足Armijo准则)为止。

$$f\left(x_k+\alpha d_k\right)\le f\left(x_k\right)+c_1\alpha\nabla f\left(x_k\right)^T d_k \qquad c_1\in\left(0,1\right)$$

回溯与二分线性搜索的异同

- 二分线性搜索的目标是求得满足$h'(\alpha)\approx 0$的最优步长近似值，而回溯线性搜索放松了对步长的约束，只要步长能使函数值有足够大的变化即可。
- 二分线性搜索可以减少下降次数，但在计算最优步长上花费了不少代价；回溯线性搜索找到一个差不多的步长即可。

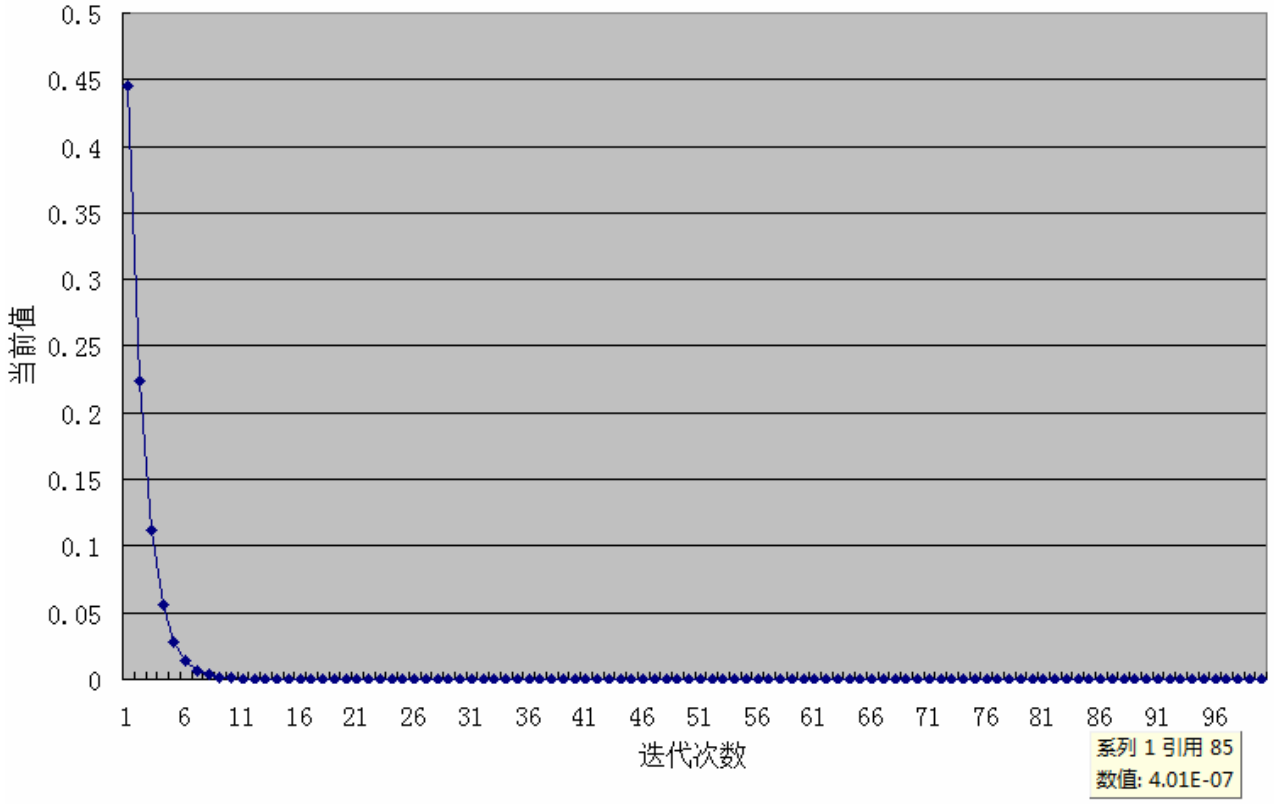
回溯线性搜索

- x为当前值
- d为x处的导数
- a为输入学习率
- 返回调整后的学习率

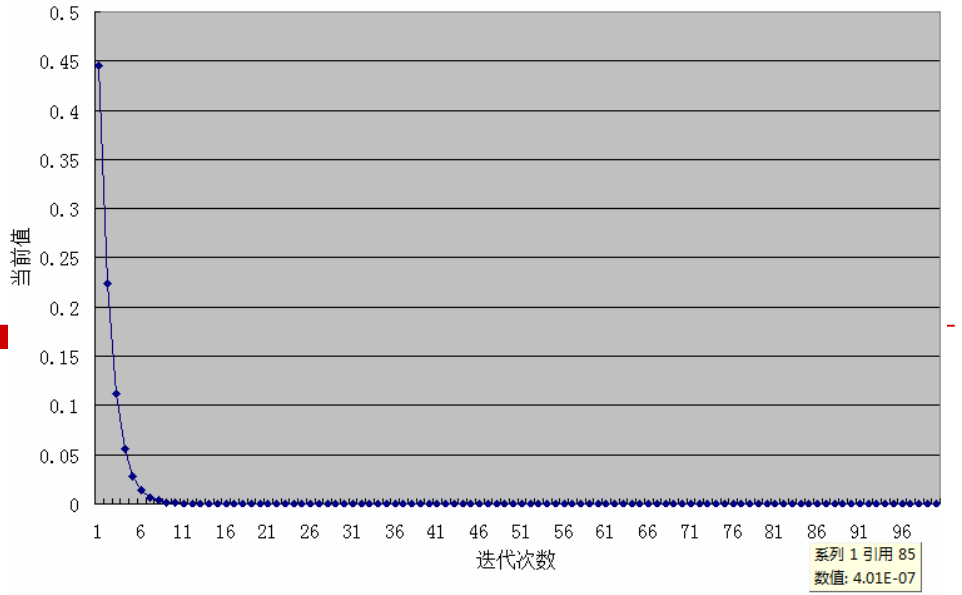
```
double GetA_Armijo(double x, double d, double a)
{
    double c1 = 0.3;
    double now = f(x);
    double next = f(x - a*d);

    int count = 30;
    while(next < now)
    {
        a *= 2;
        next = f(x - a*d);
        count--;
        if(count == 0)
            break;
    }

    count = 50;
    while(next > now - c1*a*d*d)
    {
        a /= 2;
        next = f(x - a*d);
        count--;
        if(count == 0)
            break;
    }
    return a;
}
```

实验：回溯线性搜索寻找学习率

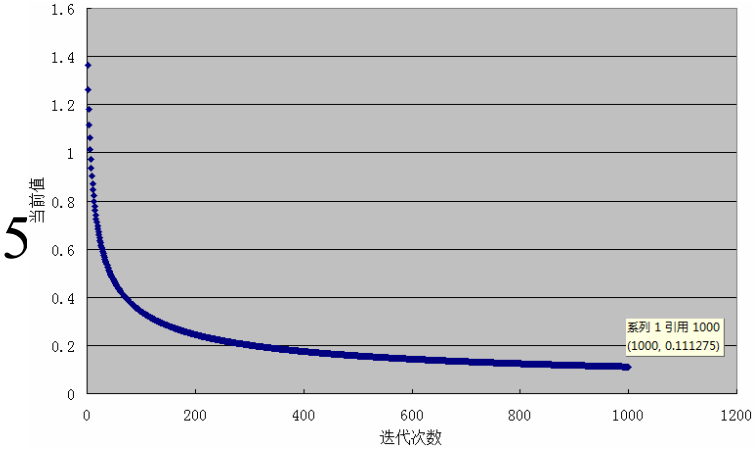
- $y=x^4$，初值取x=1.5，回溯线性方法



回溯线性搜索



- 分析：
- 效果还不错
- 经过12次迭代，x=0.00010872；
- 经过100次迭代，$x=3.64905\times10^{-7}$
  - 试比较固定学习率时：
    - 经过200次迭代，x=0.24436；
    - 经过1000次迭代，x=0.111275



回溯线性搜索的思考：插值法

- 采用多项式插值法(Interpolation)拟合简单函数，然后根据该简单函数估计函数的极值点，这样选择合适步长的效率会高很多。
- 现在拥有的数据为：x_k处的函数值$f(x_k)$及其导数$f'(x_k)$，再加上第一次尝试的步长α_0。如果α_0满足条件，显然算法退出；若α_0不满足条件，则根据上述信息可以构造一个二次近似函数：

$$h_q(\alpha)=\frac{h(\alpha_0)-h'(0)\alpha_0-h(0)}{\alpha_0^2}\alpha^2+h'(0)\alpha+h(0)$$

二次插值法求极值 $h_q(\alpha)=\frac{h(\alpha_0)-h'(0)\alpha_0-h(0)}{\alpha_0^2}\alpha^2+h'(0)\alpha+h(0)$

- 显然，导数为0的最优值为：

$$\alpha_1=\frac{h'(0)\alpha_0^2}{2[h'(0)\alpha_0+h(0)-h(\alpha_0)]}$$

- 若α_1满足Armijo准则，则输出该学习率；否则，继续迭代。

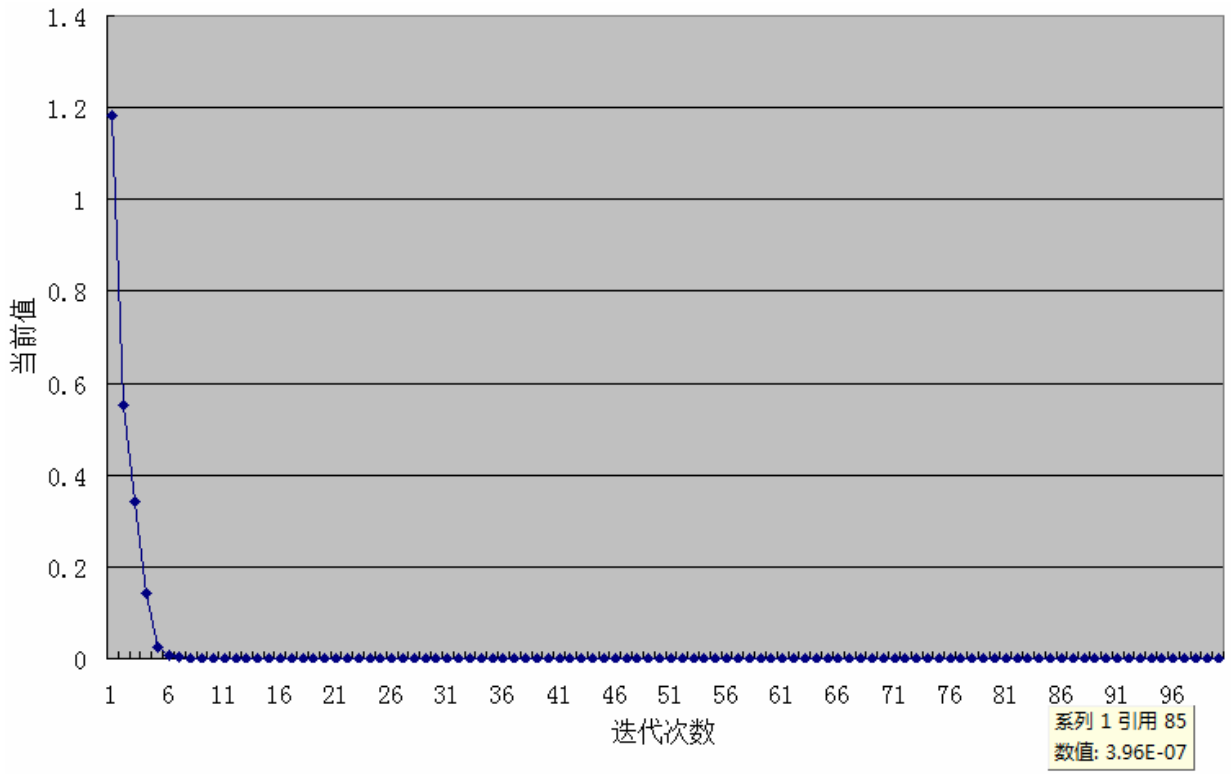
二次插值法

- x为当前值
- d为x处的导数
- a为输入学习率
- 返回调整后的学习率

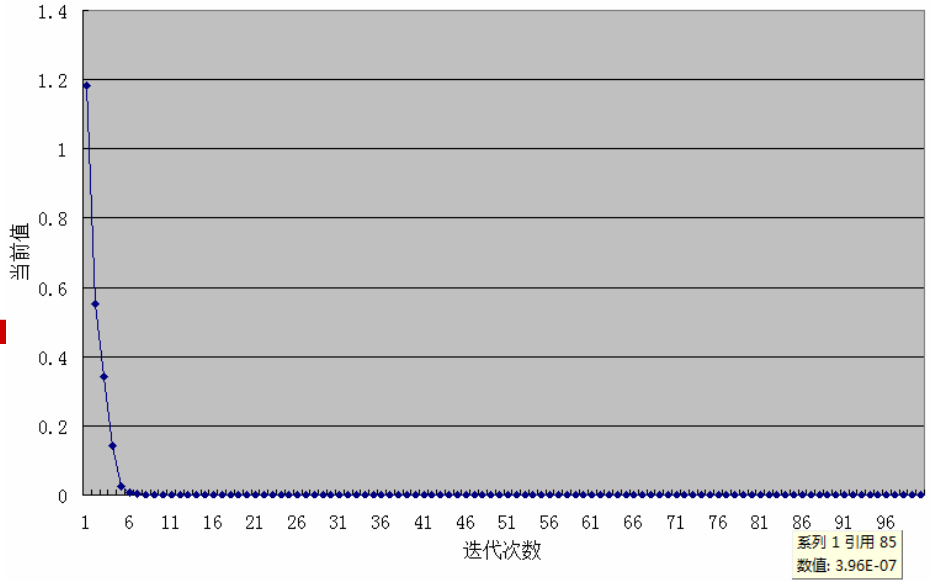
```
double GetA_Quad(double x, double d, double a)
{
    double c1 = 0.3;
    double now = f(x);
    double next = f(x - a*d);

    int count = 30;
    while(next < now)
    {
        a *= 2;
        next = f(x - a*d);
        count--;
        if(count == 0)
            break;
    }

    count = 50;
    double b;
    while(next > now - c1*a*d*d)
    {
        b = d * a * a / (now + d * a - next);
        b /= 2;
        if(b < 0)
            a /= 2;
        else
            a = b;
        next = f(x - a*d);
        count--;
        if(count == 0)
            break;
    }
    return a;
}
```

实验：二次插值线性搜索寻找学习率

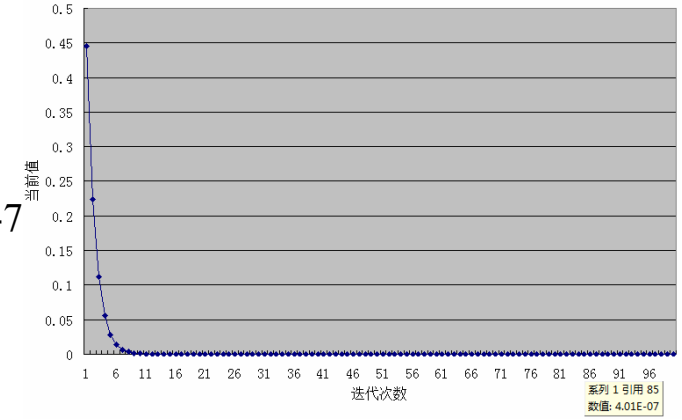
- $y=x^4$，初值取x=1.5，二次插值线性搜索方法



二次插值线性搜索



- 分析：
- 效果还不错
- 经过12次迭代，x=0.0000282229；
- 经过100次迭代，$x=3.61217\times10^{-7}$
  - 试比较回溯线性搜索时：
    - 经过12次迭代，x=0.00010872；
    - 经过1000次迭代，$x=3.649\times10^{-7}$



总结与思考

- 通过使用线性搜索的方式，能够比较好的解决学习率问题
- 一般的说，回溯线性搜索和二次插值线性搜索能够基本满足实践中的需要
- 问题：
  - 可否在搜索过程中，随着信息的增多，使用三次或者更高次的函数曲线，从而得到更快的学习率收敛速度？
  - 为避免高次产生的震荡，可否使用三次Hermite多项式，在端点保证函数值和一阶导都相等，从而构造更光顺的简单低次函数？

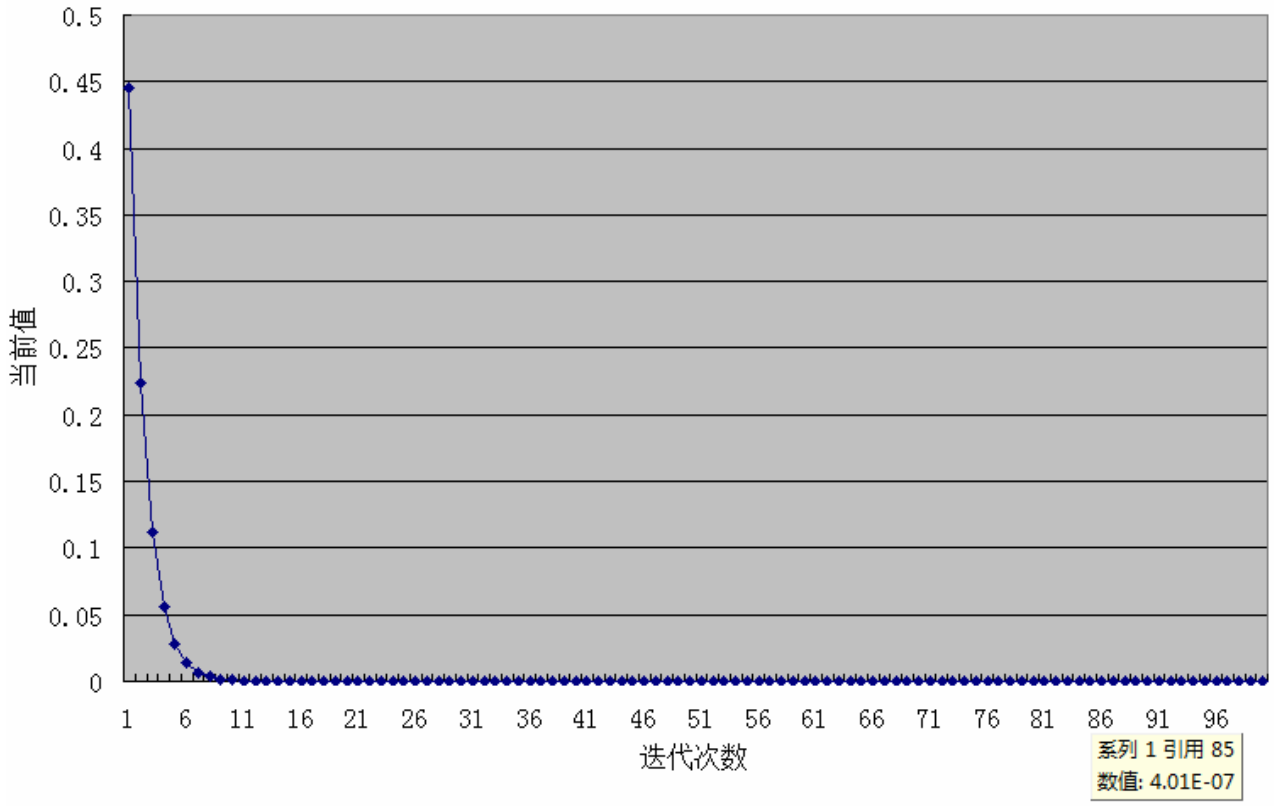
搜索方向

- 若搜索方向不是严格梯度方向，是否可以？
- 思考：
- 因为函数二阶导数反应了函数的凸凹性；二阶导越大，一阶导的变化越大。在搜索中，可否用二阶导做些“修正”？如：二者相除？

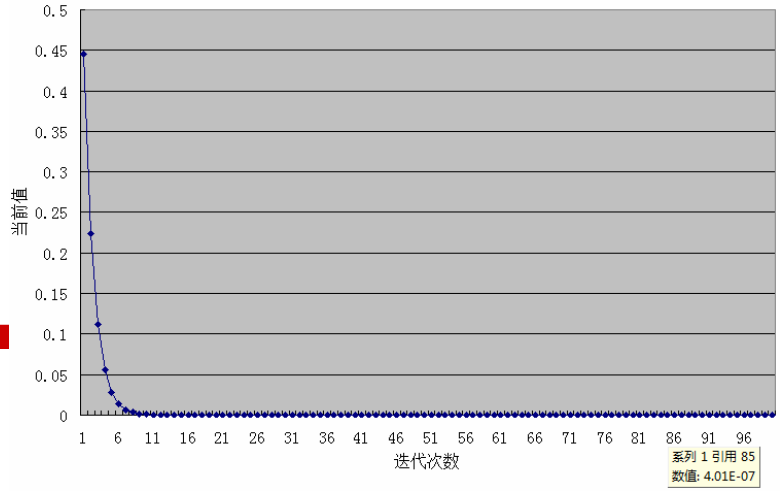
$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

实验：搜索方向的探索

- $y=x^4$，初值取x=1.5，负梯度除以二阶导



搜索方向的探索



- 效果出奇的好！
- 经过12次迭代，x=0.00770735；
- 经过100次迭代，$x=3.68948\times10^{-18}$
  - 试比较二次插值线性搜索时：
    - 经过12次迭代，x=0.0000282229；
    - 经过1000次迭代，$x=3.61217\times10^{-7}$

分析上述结果的原因

- 若f(x)二阶导连续，将f(x)在x_k处Taylor展开：

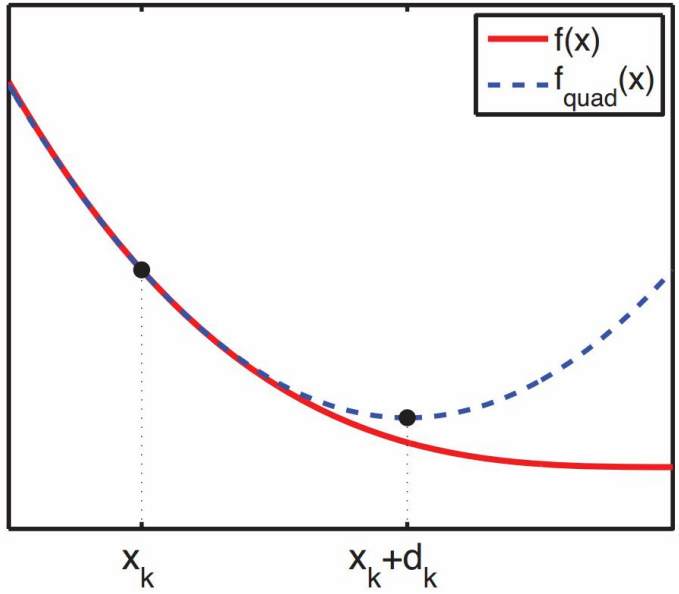
$$\varphi(x)=f(x_k)+f'(x_k)(x-x_k)+\frac{1}{2}f''(x_k)(x-x_k)^2+R_2(x)$$

$$\varphi'(x)\approx f'(x_k)+f''(x_k)(x-x_k)$$

$$x_{k+1}=x_k-\frac{f'(x_k)}{f''(x_k)}$$

牛顿法

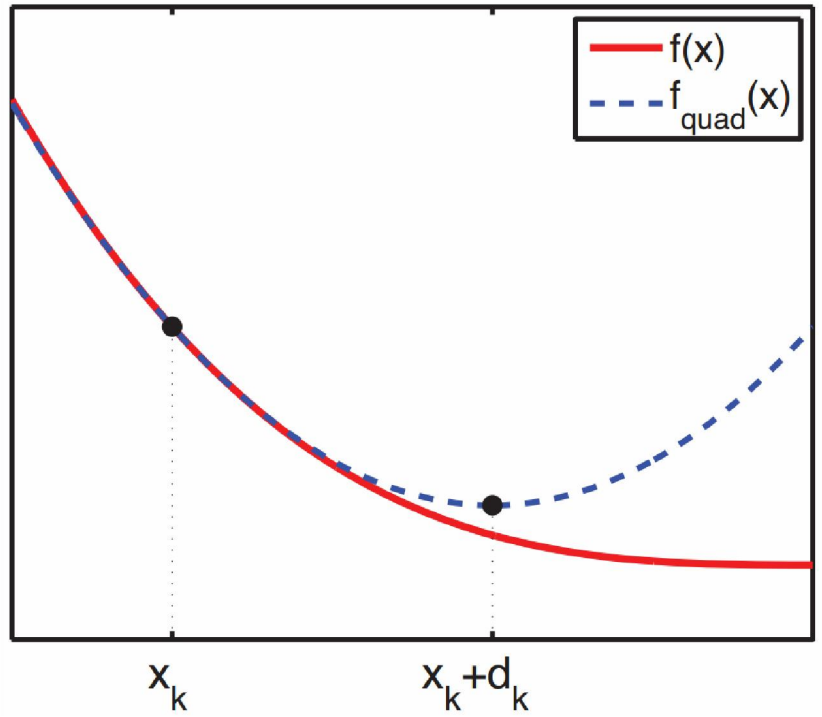
- 上述迭代公式，即牛顿法
- 该方法可以直接推广到多维：用方向导数代替一阶导，用Hessian矩阵代替二阶导

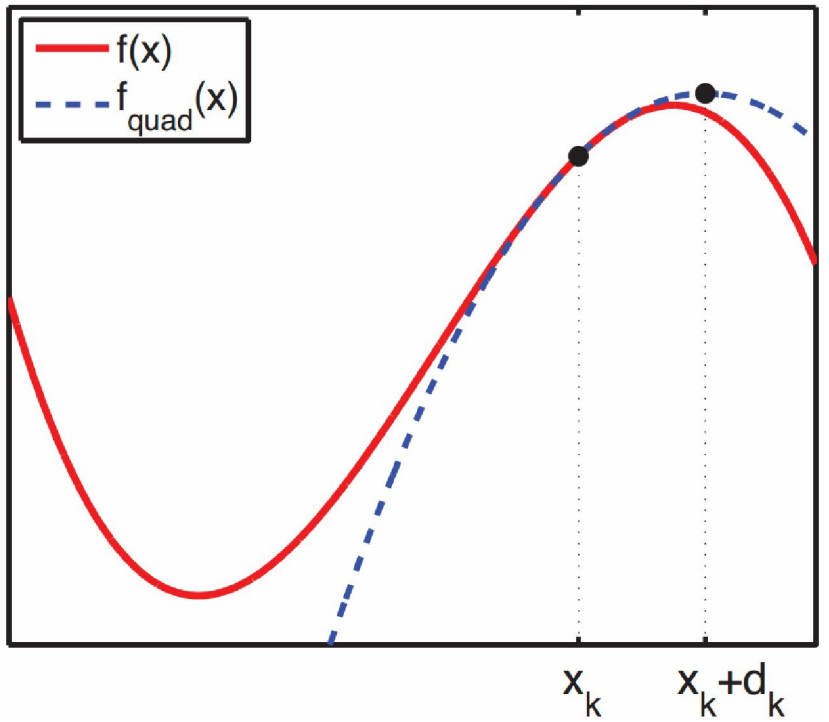


牛顿法的特点

- 经典牛顿法虽然具有二次收敛性，但是要求初始点需要尽量靠近极小点，否则有可能不收敛。
- 计算过程中需要计算目标函数的二阶偏导数，难度较大。
- 目标函数的Hessian矩阵无法保持正定，会导致算法产生的方向不能保证是f在x_k处的下降方向，从而令牛顿法失效；
- 如果Hessian矩阵奇异，牛顿方向可能根本是不存在的。

二阶导非正定的情况(一阶则为负数)





修正牛顿方向

(1) Goldstein 和 Price 于 1967 年提出, 当 $\boldsymbol{G}_k$ 非正定时, 将搜索方向取为最速下降方向 $-\boldsymbol{g}^k$. 考虑到牛顿方向 $\boldsymbol{d}_k^N$ 与负梯度方向的夹角 $\langle \boldsymbol{d}_k^N, -\boldsymbol{g}^k \rangle$, 令搜索方向

$$\boldsymbol{d}^k = \begin{cases} \boldsymbol{d}_k^N, & \text{若 } \cos\langle \boldsymbol{d}_k^N, -\boldsymbol{g}^k \rangle \geqslant \eta > 0, \\ -\boldsymbol{g}^k, & \text{其他情形}, \end{cases}$$

其中 η 为某个事先设定的正数. 这样确定的搜索方向 $\boldsymbol{d}^k$ 满足 $\cos\langle \boldsymbol{d}^k, -\boldsymbol{g}^k \rangle \geqslant \eta > 0$, 从而可以保证算法的收敛性.

(2) Goldfeld 等人 (1966) 提出, 用正定矩阵 $\boldsymbol{G}_k + v_k \boldsymbol{I}$ 替代 Hesse 矩阵 $\boldsymbol{G}_k$, 计算修正牛顿方向. 比较理想的参数 $v_k > 0$ 满足: 适当大于使 $\boldsymbol{G}_k + v\boldsymbol{I}$ 正定的 "最小" 的 v. 此时, 可以借助于修正 Cholesky 分解算法确定参数 v_k.

拟牛顿的思路

- 求Hessian矩阵的逆影响算法效率，同时，搜索方向只要和负梯度的夹角小于90°即可，因此，可以用近似矩阵代替Hessian矩阵，只要满足该矩阵正定、容易求逆，或者可以通过若干步递推公式计算得到。
  - BFGS / LBFGS
  - Broyden – Fletcher – Goldfarb - Shanno

BFGS

- 矩阵迭代公式

$$\begin{aligned}
\mathbf{B}_{k+1} &= \mathbf{B}_k + \frac{\mathbf{y}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} - \frac{(\mathbf{B}_k \mathbf{s}_k)(\mathbf{B}_k \mathbf{s}_k)^T}{\mathbf{s}_k^T \mathbf{B}_k \mathbf{s}_k} \\
\mathbf{s}_k &= \boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1} \\
\mathbf{y}_k &= \mathbf{g}_k - \mathbf{g}_{k-1}
\end{aligned}$$

- 初值选单位阵 $\mathbf{B}_0 = \mathbf{I}$

L-BFGS

- BFGS需要存储n×n的矩阵H_k用于近似Hessian矩阵的逆矩阵；而L-BFGS仅需要存储最近m(m一般小于10，m=20足够)对n维的更新数据(x,f‘(xi))即可。L-BFGS的空间复杂度O(mn)，若将m看做常数，则为线性，特别适用于变量非常多的优化问题。

熵

- 后面是下一次课“最大熵模型”的初始部分，仅介绍熵的概念本身和定义，以及基本性质。为了完整性，列出关于联合熵H(X,Y)、相对熵D(X||Y)、条件熵H(X|Y)、互信息I(X,Y)的内容，讲在下次课与最大熵模型一起阐述。

骰子

每个面朝上的概率分别是多少

- 所有人都说是等概率，即各点的概率均为1/6

为什么？

- “一无所知”的骰子
- 假定它每一个朝上概率均等是最安全的做法

新问题

- N次投掷结果的平均值为$\mu = 5.5$
- 六个面出现的次数各是多少？

优化问题

- $S(\boldsymbol{p}) = -\sum_i p_i \ln p_i$

  $-\sum_i p_i = 1$

  $-\sum_i i \cdot p_i = \mu$

  拉格朗日乘子法

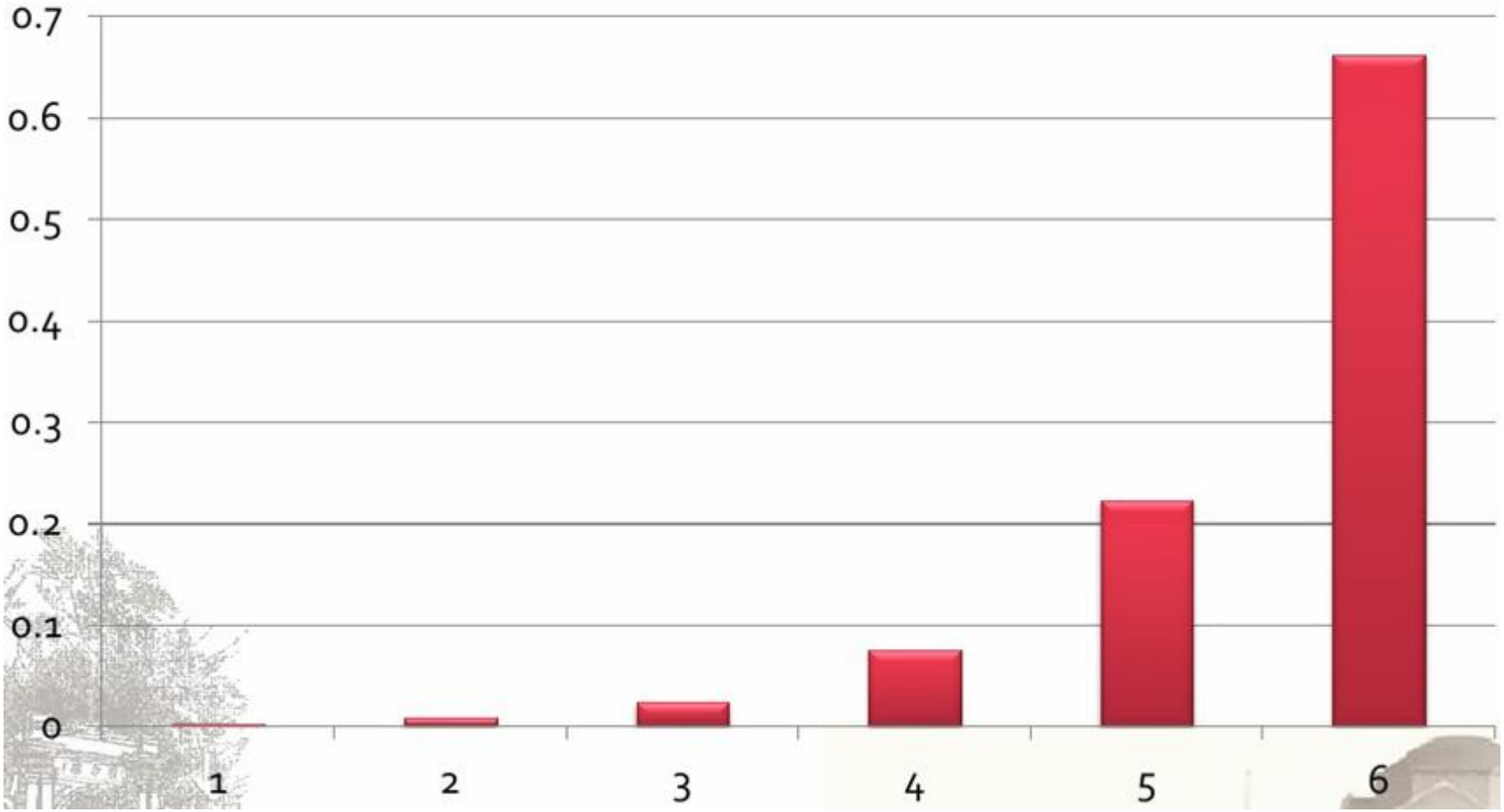
  $-\zeta = -\sum_i p_i \ln p_i + \lambda_0(1 - \sum_i p_i) + \lambda_1(\mu - \sum_i i \cdot p_i)$

  - 通过$\frac{\partial \zeta}{\partial p_i} = 0$，可得

$$p_i = e^{-1-\lambda_0 - i\lambda_1}$$

$$\lambda_0 = 5.932, \quad \lambda_1 = -1.087$$

预测结果

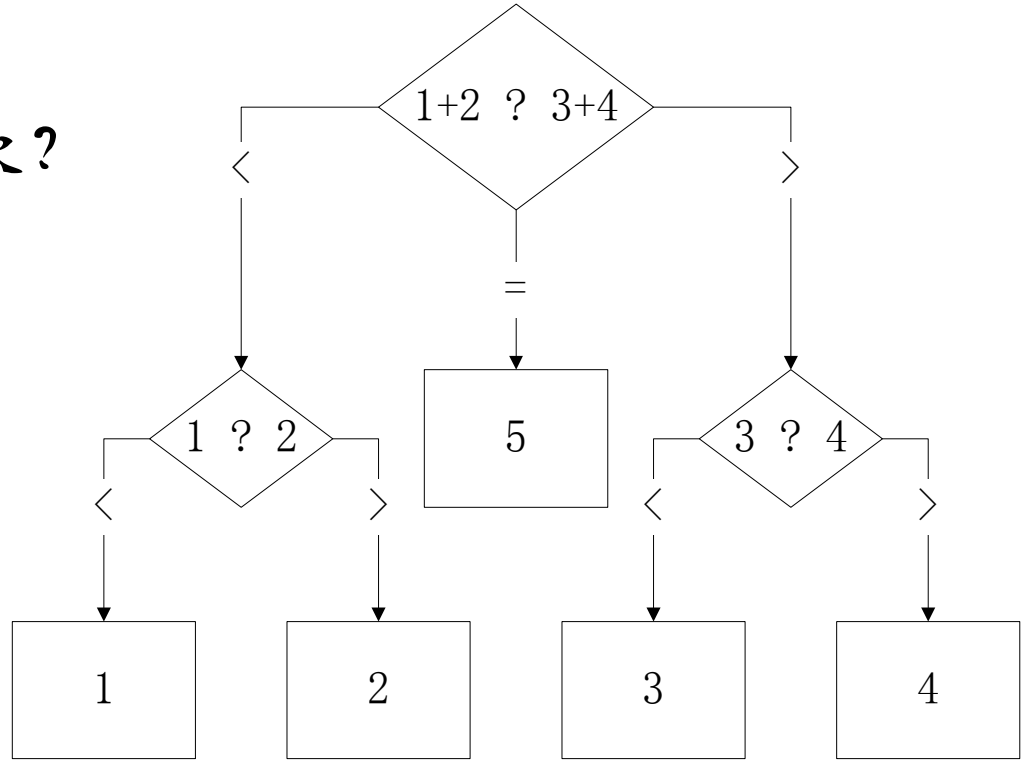


从小学数学开始

- 假设有5个硬币：1,2,3,4,5，其中一个是假的，比其他的硬币轻。有一个天平，天平每次能比较两堆硬币，得出的结果可能是以下三种之一：
  - 左边比右边轻
  - 右边比左边轻
  - 两边同样重
- 问：至少要使用天平多少次才能确保找到假硬币？

答案

- 一种可能的称量方法如右图所示
- 答案：2次
- 追问：为什么2次？



分析

- 令x表示假硬币的序号： $x \in X=\{1,2,3,4,5\}$；
- 令yi是第i次使用天平所得到的结果：
  $y \in Y=\{1,2,3\}$；
  - 1表示“左轻”，2表示“平衡”，3表示“右轻”
- 用天平称n次，获得的结果是：y1 y2… yn；
- y1 y2… yn的所有可能组合数目是3^n；
- 根据题意，要求通过y1 y2… yn确定x。即建立影射map(y1y2…yn)=x；
- 从而：y1y2…yn的变化数目大于等于x的变化数目
  - 即$3^n \geqslant 5$
  - 一般意义下：$|Y|^n \geq |X|$

进一步分析

- 用y1 y2… yn表达x。即设计编码：x-> y1 y2… yn
- X的“总不确定度”是：$H(X)=\log|X|=\log 5$
- Y的“表达能力”是：$H(Y)=\log|Y|=\log 3$
- 至少要多少个Y才能准确表示X？

$$\frac{H(X)}{H(Y)}=\frac{\log 5}{\log 3}=1.46$$

题目的变种

- 假设有5个硬币：1,2,3,4,5，其中一个是假的，比其他的硬币轻。已知第一个硬币是假硬币的概率是三分之一；第二个硬币是假硬币的概率也是三分之一，其他硬币是假硬币的概率都是九分之一。
- 有一个天平，天平每次能比较两堆硬币，得出的结果可能是以下三种之一：
  - 左边比右边轻
  - 右边比左边轻
  - 两边同样重
- 假设使用天平n次找到假硬币。问n的期望值至少是多少？

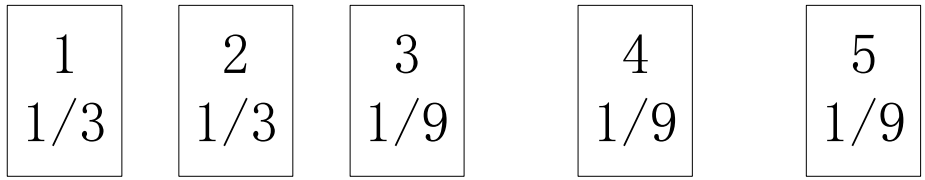
解

- 1/3概率的硬币有2个，1/9概率的硬币有3个：

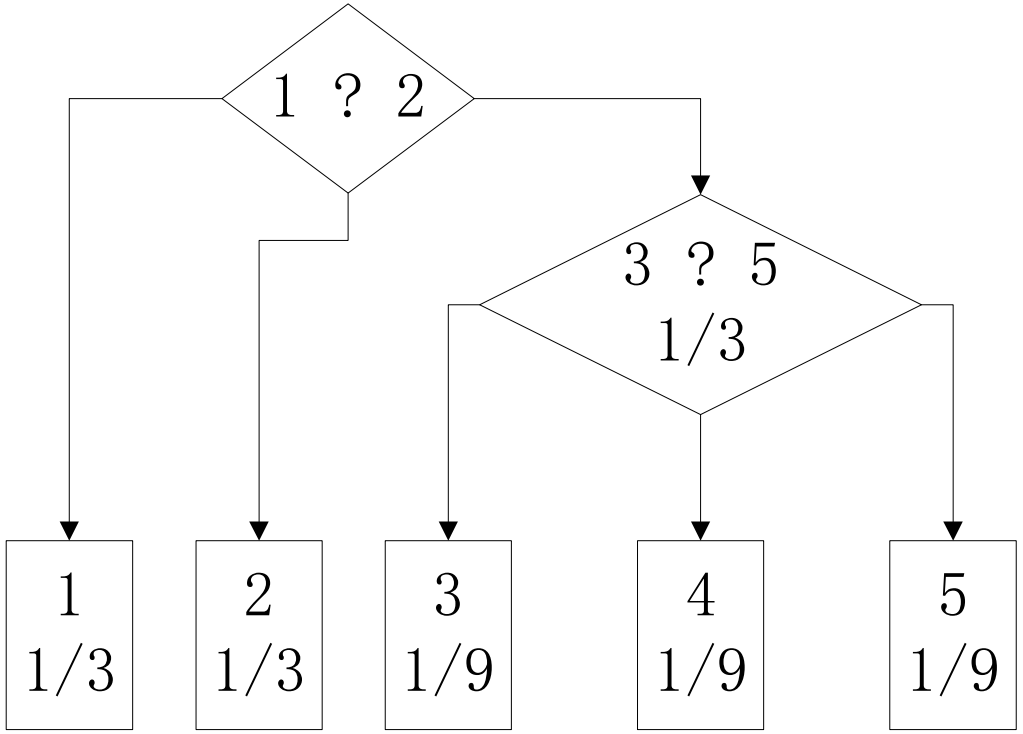
$$\left(\frac{1}{3}+\frac{1}{3}\right)\times\frac{\log 3}{\log 3}+3\frac{1}{9}\times\frac{\log 9}{\log 3}=\frac{4}{3}$$

- 定义：$-\Sigma p\log_a p$为熵

用熵解释Huffman编码



用熵解释Huffman编码



广泛的结论

- 如果一个随机变量x的可能取值为X={x1, x2,..., xk}。要用n位y: y1y2...yn表示(每位y有c种取值)X，那么n的期望值至少为：

$$\sum_{i=1}^{k} p(x=x_i)\frac{\log\frac{1}{p(x=x_i)}}{\log c} = \frac{\sum_{i=1}^{k} p(x=x_i)\log\frac{1}{p(x=x_i)}}{\log c}$$

- 一般地，我们令c为2(二进制表示)，于是，X的信息量为：

$$H(X)=\sum_{i=1}^{k} p(x=x_i)\log\frac{1}{p(x=x_i)}$$

熵

- 将P(x=xi)写成普适公式，就得到熵的定义：

$$H(X)=\sum_{x\in X}p(x)\log\frac{1}{p(x)}$$

研究函数f(x)=xlnx

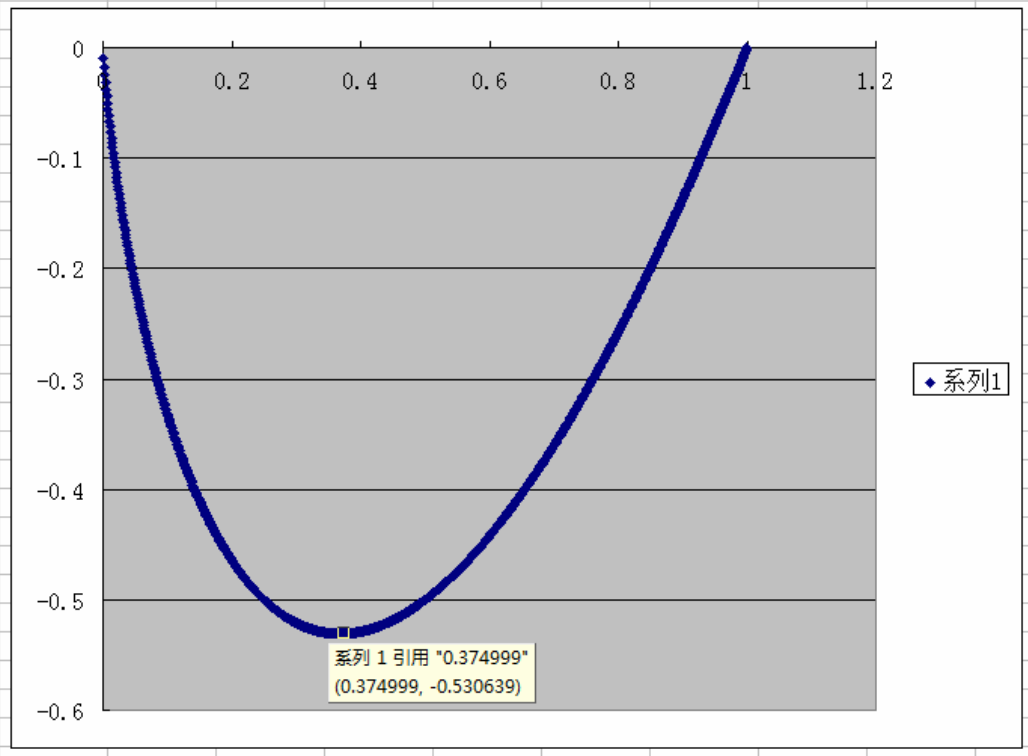
- f(x)=xlnx，x ∈ [0,1]
- f’(x) = lnx + 1
- f’’(x) = 1/x>0(凸函数)
- 当f’(x)=0时，x=1/e，取极小值；
- 由于 $\lim_{x\to 0} f(x)=0$ $\lim_{x\to 1} f(x)=0$
- 定义f(0)=f(1)=0

离散采样

```
int _tmain(int argc, _TCHAR* argv[])
{
    float x = 0.001f;
    float y;
    float log2 = log(2.0f);
    ofstream oFile;
    oFile.open(_T("D:\\entropy.txt"));
    while(x < 1)
    {
        y = x * log(x) / log2;
        oFile << x << '\t' << y << '\n';

        x += 0.001f;
    }
    oFile.close();
    return 0;
}
```

绘图

0.001	-0.00997
0.002	-0.01793
0.003	-0.02514
0.004	-0.03186
0.005	-0.03822
0.006	-0.04428
0.007	-0.05011
0.008	-0.05573
0.009	-0.06116
0.01	-0.06644
0.011	-0.07157
0.012	-0.07657
0.013	-0.08145
0.014	-0.08622
0.015	-0.09088
0.016	-0.09545
0.017	-0.09993
0.018	-0.10433
0.019	-0.10864
0.02	-0.11288
0.021	-0.11704
0.022	-0.12114
0.023	-0.12517
0.024	-0.12914
0.025	-0.13305
0.026	-0.1369
0.027	-0.14069
0.028	-0.14444
0.029	-0.14813
0.03	-0.15177
0.031	-0.15536



熵和不确定性

- 熵是随机变量不确定性的度量，不确定性越大，熵值越大；若随机变量退化成定值，熵为0
- 均匀分布是“最不确定”的分布
  - 如何证明？

两点分布的最大熵

- $H(X) = -p\ln p - (1-p)\ln(1-p)$
  - 注：经典熵的定义，底数是2，单位是bit
  - 本例中，为分析方便使用底数e
  - 若底数是e，单位是nat(奈特)
- 如何求最值？

X满足均匀分布时，熵最大

- 当p=0.5时，取H(X)取最大值；
  - 思考：若“多点”分布呢？
- X是随机变量，可以取从1到K的K个数。
  问：X满足什么分布时，X的熵最大？
  - p(X)=1/K：均匀分布

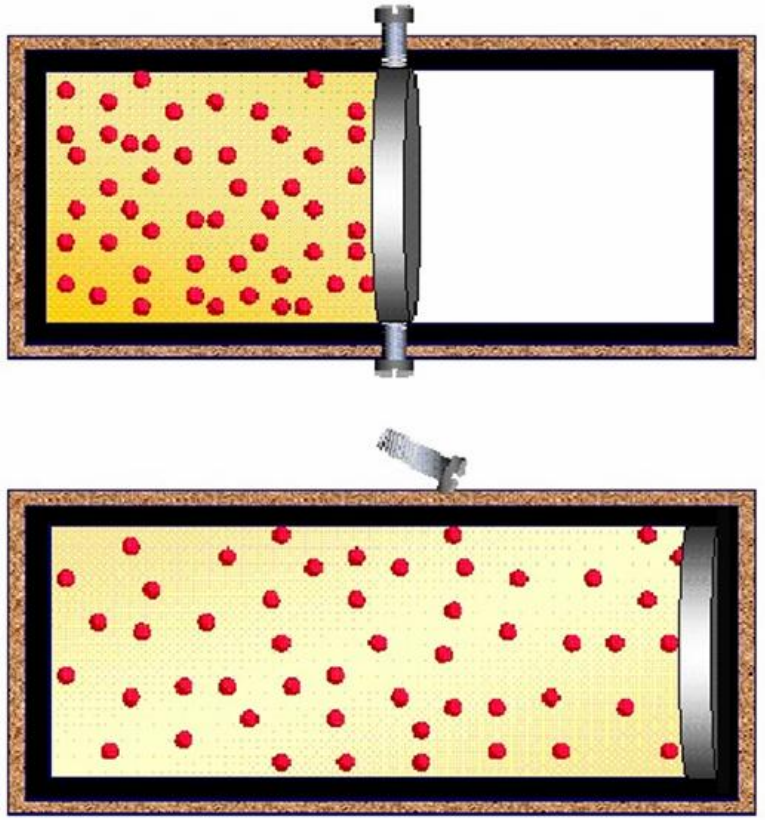
$$0 \le H(X) \le \log|X|$$

自封闭系统的运动总是倒向均匀分布

- 密封箱子中间放一隔板
- 隔板左边空间注入烟，右边真空

去掉隔板会怎样？

- 左边的烟就会自然（自发）地向右边扩散，最后均匀地占满整个箱体



联合熵和条件熵

- 两个随机变量X，Y的联合分布，可以形成联合熵Joint Entropy，用H(X,Y)表示
- H(X,Y) – H(X)
  - (X,Y)发生所包含的熵，减去X单独发生包含的熵：在X发生的前提下，Y发生“新”带来的熵
  - 该式子定义为X发生前提下，Y的熵：
    - 条件熵H(Y|X)

推导条件熵的定义式

$$
\begin{aligned}
& H(X,Y)-H(X) \\
&= -\sum_{x,y} p(x,y)\log p(x,y) + \sum_{x} p(x)\log p(x) \\
&= -\sum_{x,y} p(x,y)\log p(x,y) + \sum_{x}\left(\sum_{y} p(x,y)\right)\log p(x) \\
&= -\sum_{x,y} p(x,y)\log p(x,y) + \sum_{x,y} p(x,y)\log p(x) \\
&= -\sum_{x,y} p(x,y)\log \frac{p(x,y)}{p(x)} \\
&= -\sum_{x,y} p(x,y)\log p(y\mid x)
\end{aligned}
$$

相对熵

- 相对熵，又称互熵，交叉熵，鉴别信息，Kullback熵，Kullback-Leible散度等
- 设p(x)、q(x)是X中取值的两个概率分布，则p对q的相对熵是

$$D(p\,\|\,q)=\sum_{x}p(x)\log\frac{P(x)}{q(x)}=E_{p(x)}\log\frac{p(x)}{q(y)}$$

- 两点说明：
  - 在一定程度上，相对熵可以度量两个随机变量的“距离”
  - 一般的，D(p||q) ≠ D(q||p)

互信息

- 两个随机变量X，Y的互信息，定义为X，Y的联合分布和独立分布乘积的相对熵。
- I(X,Y)=D(P(X,Y) || P(X)P(Y))

$$I(X,Y)=\sum_{x,y}p(x,y)\log\frac{p(x,y)}{p(x)p(y)}$$

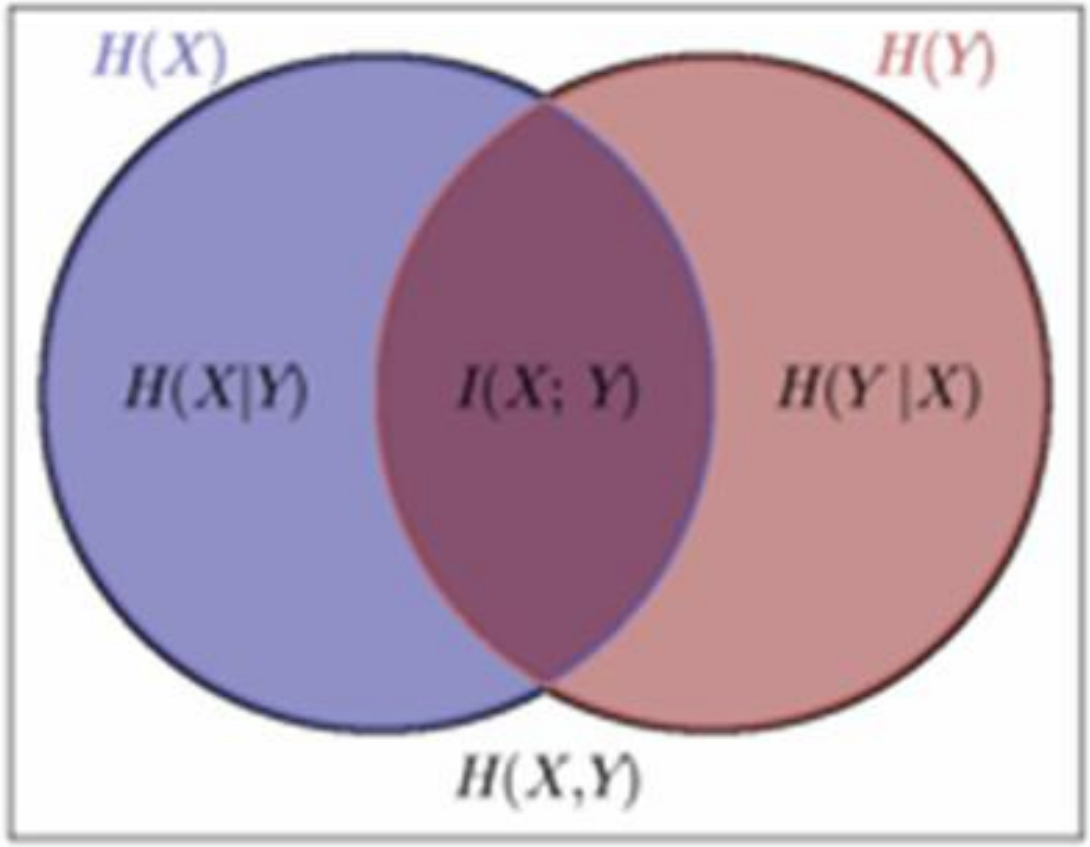
计算H(Y)-I(X,Y)

$$
\begin{aligned}
& H(Y) - I(X,Y) \\
&= -\sum_{y} p(y) \log p(y) - \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \\
&= -\sum_{y} \left(\sum_{x} p(x,y) \right) \log p(y) - \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \\
&= -\sum_{x,y} p(x,y) \log p(y) - \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \\
&= -\sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)} \\
&= -\sum_{x,y} p(x,y) \log p(y \mid x) \\
&= H(Y \mid X)
\end{aligned}
$$

整理得到的等式

- $H(Y|X) = H(X,Y) - H(X)$
  - 条件熵定义
- $H(Y|X) = H(Y) - I(X,Y)$
  - 根据互信息定义展开得到
  - 有些文献将$I(X,Y)=H(Y) – H(Y|X)$作为互信息的定义式
- 对偶式
  - $H(X|Y)= H(X,Y) - H(Y)$
  - $H(X|Y)= H(X) - I(X,Y)$
- $I(X,Y)= H(X) + H(Y) - H(X,Y)$
  - 有些文献将该式作为互信息的定义式
- 试证明：$H(X|Y) \leqslant H(X)$，　$H(Y|X) \leqslant H(Y)$

强大的Venn图：帮助记忆



参考文献

- Kevin P. Murphy, Machine Learning: A Probabilistic Perspective, MIT Press, 2012
- Elements of Information Theory (Cover & Thomas)
- Linear and Nonlinear Programming (Nash & Sofer)
- 统计学习方法，李航著，清华大学出版社，2012年

感谢大家！

恳请大家批评指正！