降维

七月算法 邹博

2015年11月14日

复习：熵的等式

- $H(Y|X) = H(X,Y) - H(X)$
  - 条件熵定义
- $H(Y|X) = H(Y) - I(X,Y)$
  - 互信息定义展开
  - 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：$H(X|Y) \leqslant H(X)$，$H(Y|X) \leqslant H(Y)$

复习：决策树学习的生成算法

- 建立决策树的关键，即在当前状态下选择哪个属性作为分类依据。根据不同的目标函数，建立决策树主要有一下三种算法。
  - ID3
  - C4.5
  - CART

信息增益

- 概念：当熵和条件熵中的概率由数据估计(特别是极大似然估计)得到时，所对应的熵和条件熵分别称为**经验熵**和**经验条件熵**。
- 信息增益表示得知特征A的信息而使得类X的信息的不确定性减少的程度。
- 定义：特征A对训练数据集D的信息增益g(D,A)，定义为集合D的经验熵H(D)与特征A给定条件下D的经验条件熵H(D|A)之差，即：
  - $g(D,A)=H(D)-H(D|A)$
  - 显然，这即为训练数据集D和特征A的互信息。

基本记号

- 设训练数据集为D，$|D|$表示样本个数。
- 设有K个类$C_k, k=1,2\cdots K$，$|C_k|$为属于类C_k的样本个数，有：$\sum_k |C_k| = |D|$
- 设特征A有n个不同的取值$\{a_1, a_2 \cdots a_n\}$，根据特征A的取值将D划分为n个子集$D_1, D_2 \cdots D_n$，$|D_i|$为D_i的样本个数，有：$\sum_i |D_i| = |D|$
- 记子集D_i中属于类C_k的样本的集合为D_{ik}，$|D_{ik}|$为D_{ik}的样本个数。

信息增益的计算方法

- 计算数据集D的经验熵 $H(D) = -\sum_{k=1}^{K} \frac{C_k}{D} \log \frac{C_k}{D}$
- 遍历所有特征，对于特征A：
  - 计算特征A对数据集D的经验条件熵H(D|A)
  - 计算特征A的信息增益：g(D,A)=H(D) – H(D|A)
  - 选择信息增益最大的特征作为当前的分裂特征

经验条件熵H(D|A)

$$H(D|A)=-\sum_{i,k}p(D_k,A_i)\log p(D_k|A_i)$$

$$=-\sum_{i,k}p(A_i)p(D_k|A_i)\log p(D_k|A_i)$$

$$=-\sum_{i=1}^{n}\sum_{k=1}^{K}p(A_i)p(D_k|A_i)\log p(D_k|A_i)$$

$$=-\sum_{i=1}^{n}p(A_i)\sum_{k=1}^{K}p(D_k|A_i)\log p(D_k|A_i)$$

$$=-\sum_{i=1}^{n}\frac{|D_i|}{|D|}\sum_{k=1}^{K}\frac{|D_{ik}|}{|D_i|}\log\frac{|D_{ik}|}{|D_i|}$$

其他目标

- 信息增益率：$g_r(D,A) = g(D,A) / H(A)$
- 基尼指数：

$$Gini(p)=\sum_{k=1}^{K}p_k(1-p_k)=1-\sum_{k=1}^{K}p_k^2$$

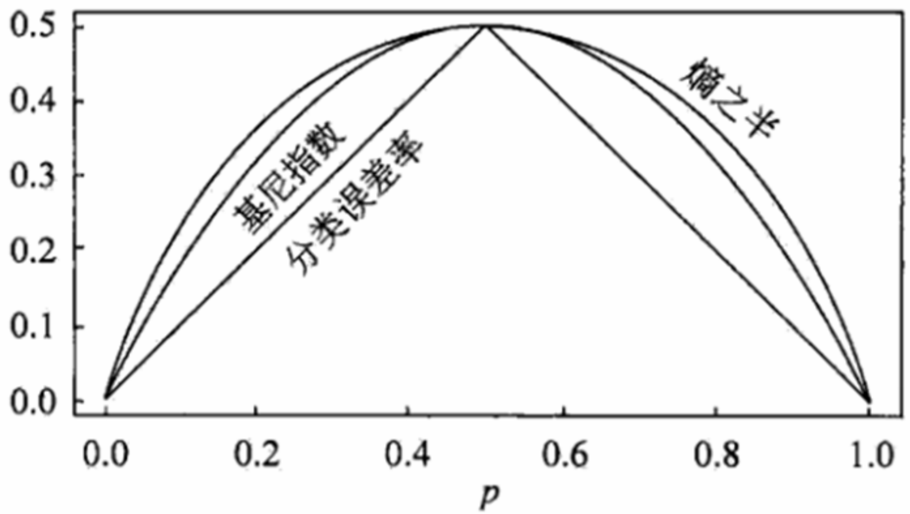
$$=1-\sum_{k=1}^{K}\left(\frac{|C_k|}{D}\right)^2$$

关于基尼指数的讨论(一家之言)

- 考察基尼指数的图像、熵、分类误差率三者之间的关系
  - 将f(x)=-lnx在x=1处一阶展开，忽略高阶无穷小，得到f(x)≈1-x

$$H(X) = -\sum_{k=1}^{K} p_k \ln p_k$$

$$\approx \sum_{k=1}^{K} p_k (1 - p_k)$$

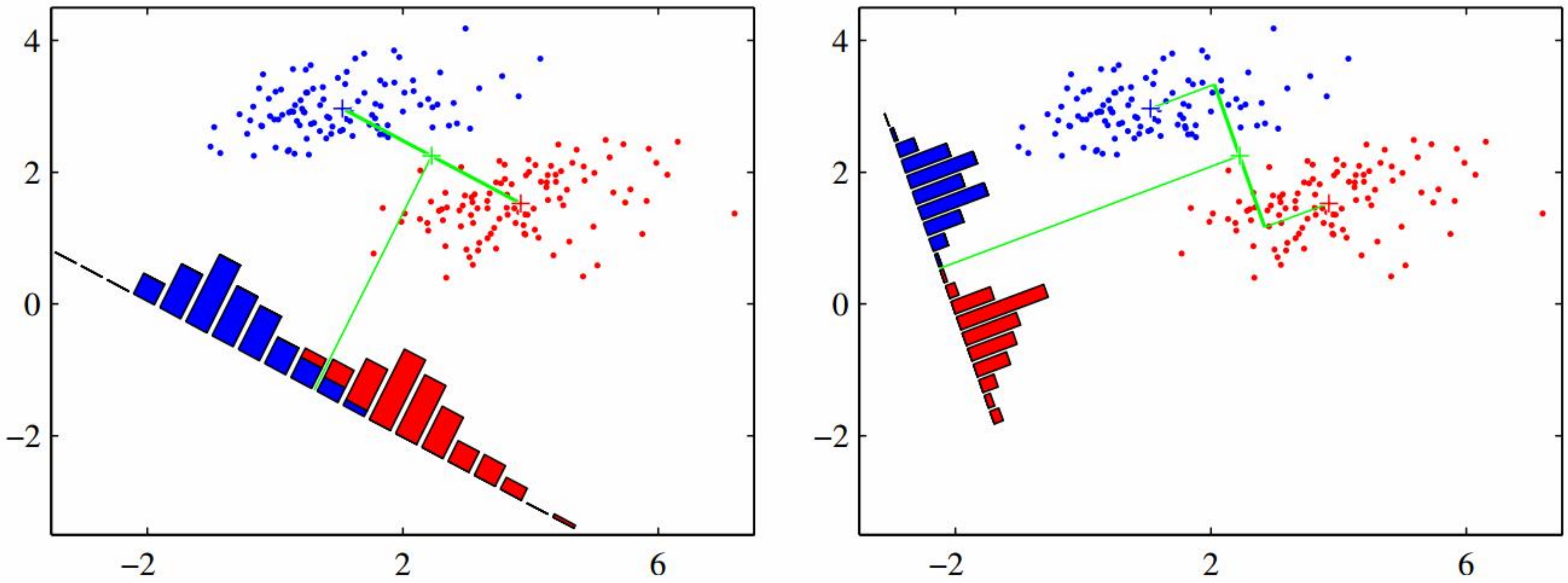


LDA：Linear Discriminant Analysis

- 给定带标记的数据(x_i,c_i)，其中，标记只分两类，即：$c_i=0$或者$c_i=1$，设计分类器，将数据分开。
- 方法：
  - Logistic/Softmax(MaxEnt)
  - SVM
  - 随机森林
  - LDA：Fisher’s linear discriminant
  - 注意：主题模型LDA(Latent Dirichlet Allocation)

LDA的思路

- 假定两类数据线性可分，即：存在一个超平面，将两类数据分开。则：存在某旋转向量，将两类数据投影到1维，并且可分。



LDA的推导

- 假定旋转向量为w，将数据x投影到一维y，得到

$$y = \vec{w}^T \vec{x}$$

- 从而，可以方便的找到阈值w_0，$y \geqslant w_0$时为C_1类，否则为C_2类。

类内均值和方差

- 令C_1有N_1个点，C_2有N_2个点，投影前的类内均值和投影后的类内均值、松散度为：

$$\begin{cases}\vec{m}_1=\dfrac{1}{N_1}\displaystyle\sum_{i=1}^{N_1}\vec{x}_i\\[2ex]\vec{m}_2=\dfrac{1}{N_2}\displaystyle\sum_{i=1}^{N_2}\vec{x}_i\end{cases}\qquad\begin{cases}m_1=w^T\vec{m}_1\\ m_2=w^T\vec{m}_2\end{cases}\qquad\begin{cases}s_1^2=\displaystyle\sum_{i=1}^{N_1}(y_i-m_1)^2\\[2ex] s_2^2=\displaystyle\sum_{i=1}^{N_2}(y_i-m_2)^2\end{cases}$$

- 松散度(scatter)，一般称为散列值，是样本松散程度的度量，值越大，越分散。
- 严格的说，m_2应该写成：$\vec{m}_2=\dfrac{1}{N_2}\displaystyle\sum_{i=N_1+1}^{N_1+N_2}\vec{x}_i$

Fisher判别准则

- 目标函数： $$J(\vec{w})=\frac{(m_2-m_1)^2}{s_1^2+s_2^2} \Rightarrow J(\vec{w})=\frac{\vec{w}^T S_B \vec{w}}{\vec{w}^T S_W \vec{w}}$$
- 向量表示：

$$\begin{aligned}
(m_2-m_1)^2 &= \left(w^T\vec{m}_2-w^T\vec{m}_1\right)^2 \\
&= \left(w^T(\vec{m}_2-\vec{m}_1)\right)^2 = \left((\vec{m}_2-\vec{m}_1)^T w\right)^2 \\
&= \left((\vec{m}_2-\vec{m}_1)^T w\right)^T \left((\vec{m}_2-\vec{m}_1)^T w\right) \\
&= \left(w^T(\vec{m}_2-\vec{m}_1)\right)\left((\vec{m}_2-\vec{m}_1)^T w\right) \\
&= w^T(\vec{m}_2-\vec{m}_1)(\vec{m}_2-\vec{m}_1)^T w^T \\
&\xleftrightarrow{\text{令}S_b=(\vec{m}_2-\vec{m}_1)(\vec{m}_2-\vec{m}_1)^T} w^T S_b w
\end{aligned}$$

Fisher判别准则

- 目标函数：$$J(\vec{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} \Rightarrow J(\vec{w}) = \frac{\vec{w}^T S_B \vec{w}}{\vec{w}^T S_W \vec{w}}$$
- 其中：$$S_B = (\vec{m}_2 - \vec{m}_1)(\vec{m}_2 - \vec{m}_1)^T$$

$$S_W = \left(\sum_{i=1}^{N_1} (\vec{x}_i - \vec{m}_1)(\vec{x}_i - \vec{m}_1)^T \right) + \left(\sum_{i=1}^{N_2} (\vec{x}_i - \vec{m}_2)(\vec{x}_i - \vec{m}_2)^T \right)$$

- Within-class scatter matrix
- Between-class scatter
- S_w, S_b可以通过样本计算得到(已知)。

目标函数求极值 $J(\vec{w})=\dfrac{\vec{w}^T S_B \vec{w}}{\vec{w}^T S_W \vec{w}}$

- 求驻点：

$$\frac{\partial J(\vec{w})}{\partial \vec{w}}=\left(\frac{\vec{w}^T S_B \vec{w}}{\vec{w}^T S_W \vec{w}}\right)'$$

$$=\frac{\left(\vec{w}^T S_B \vec{w}\right)'\left(\vec{w}^T S_W \vec{w}\right)-\left(\vec{w}^T S_W \vec{w}\right)'\left(\vec{w}^T S_B \vec{w}\right)}{\left(\vec{w}^T S_W \vec{w}\right)^2}$$

$$=\frac{2S_B\vec{w}\left(\vec{w}^T S_W \vec{w}\right)-2S_W\vec{w}\left(\vec{w}^T S_B \vec{w}\right)}{\left(\vec{w}^T S_W \vec{w}\right)^2}\overset{令}{=}0$$

$$\Rightarrow S_B\vec{w}\left(\vec{w}^T S_W \vec{w}\right)=S_W\vec{w}\left(\vec{w}^T S_B \vec{w}\right)$$

$$\Rightarrow S_B\vec{w}\propto S_W\vec{w}$$

Fisher判别投影向量公式

- 以上推导得到 $S_B\vec{w} \propto S_W\vec{w}$
- 根据S_B的计算公式 $S_B = (\vec{m}_2 - \vec{m}_1)(\vec{m}_2 - \vec{m}_1)^T$
- 得：

$$\begin{aligned} S_B\vec{w} &= (\vec{m}_2 - \vec{m}_1)(\vec{m}_2 - \vec{m}_1)^T \vec{w} \\ &= (\vec{m}_2 - \vec{m}_1)\left((\vec{m}_2 - \vec{m}_1)^T \vec{w}\right) \propto (\vec{m}_2 - \vec{m}_1) \end{aligned}$$

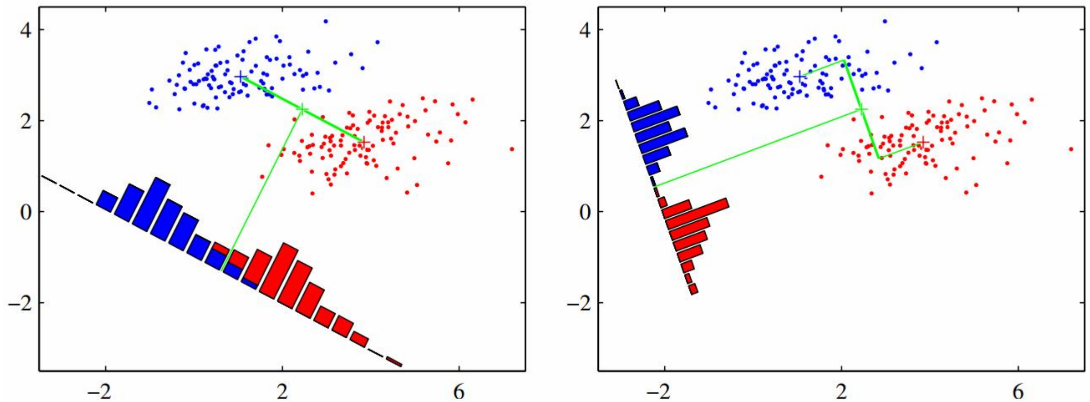
- 从而：

$$S_W\vec{w} \propto S_B\vec{w} \propto \vec{m}_2 - \vec{m}_1$$

- 若S_W可逆，则：$\vec{w} \propto S_W^{-1}(\vec{m}_2 - \vec{m}_1)$

LDA与分类 $\vec{w} \propto S_W^{-1}(\vec{m}_2 - \vec{m}_1)$

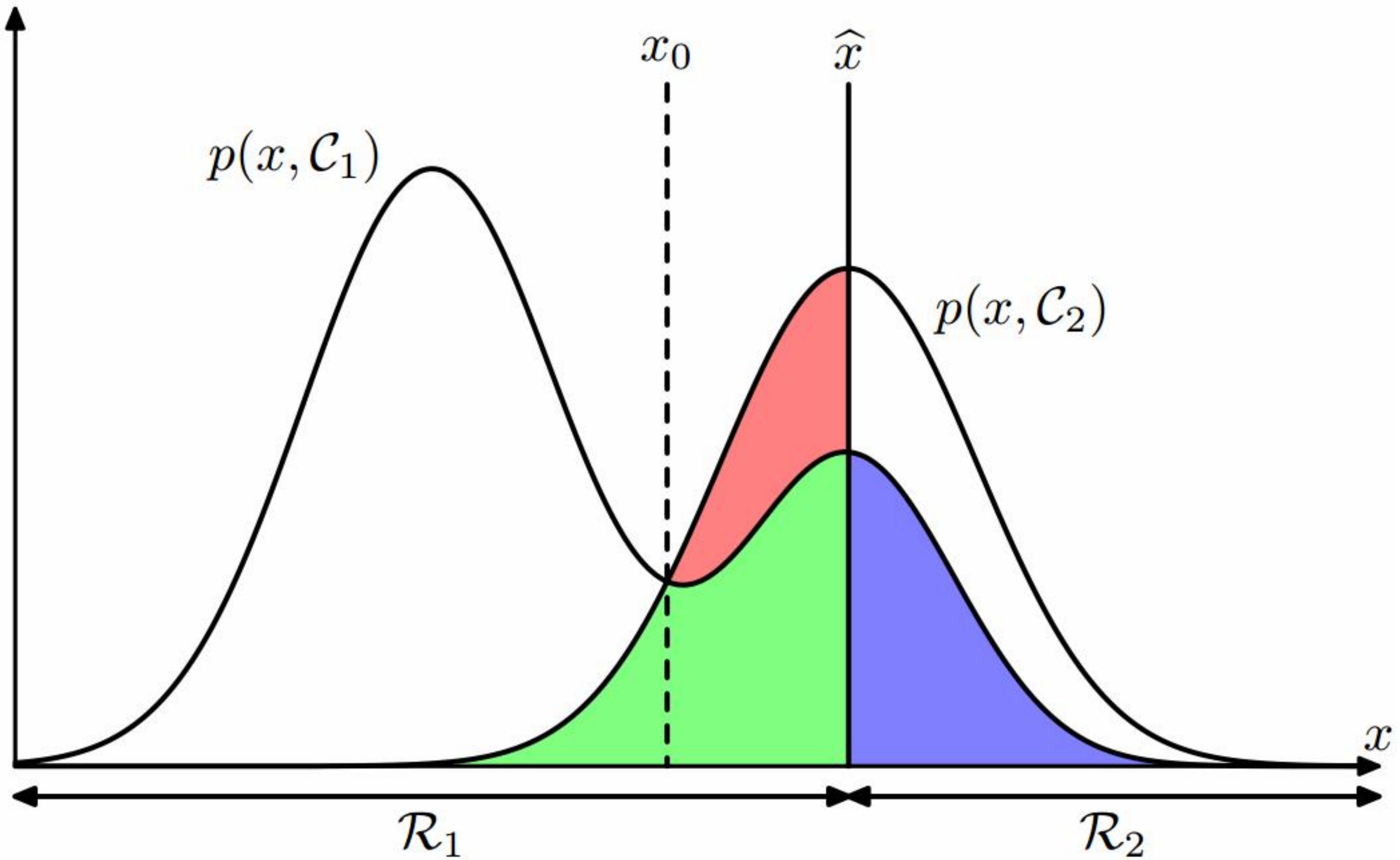
- 线性判别分析(Fisher's linear discriminant)
  - 严格的说，它只是给出了数据的特定投影方向
- 投影后，数据可以方便的找到阈值w_0，$y \geqslant w_0$时为C_1类，否则为C_2类。
  - 思考：一维数据下，可以如何分类？



Code

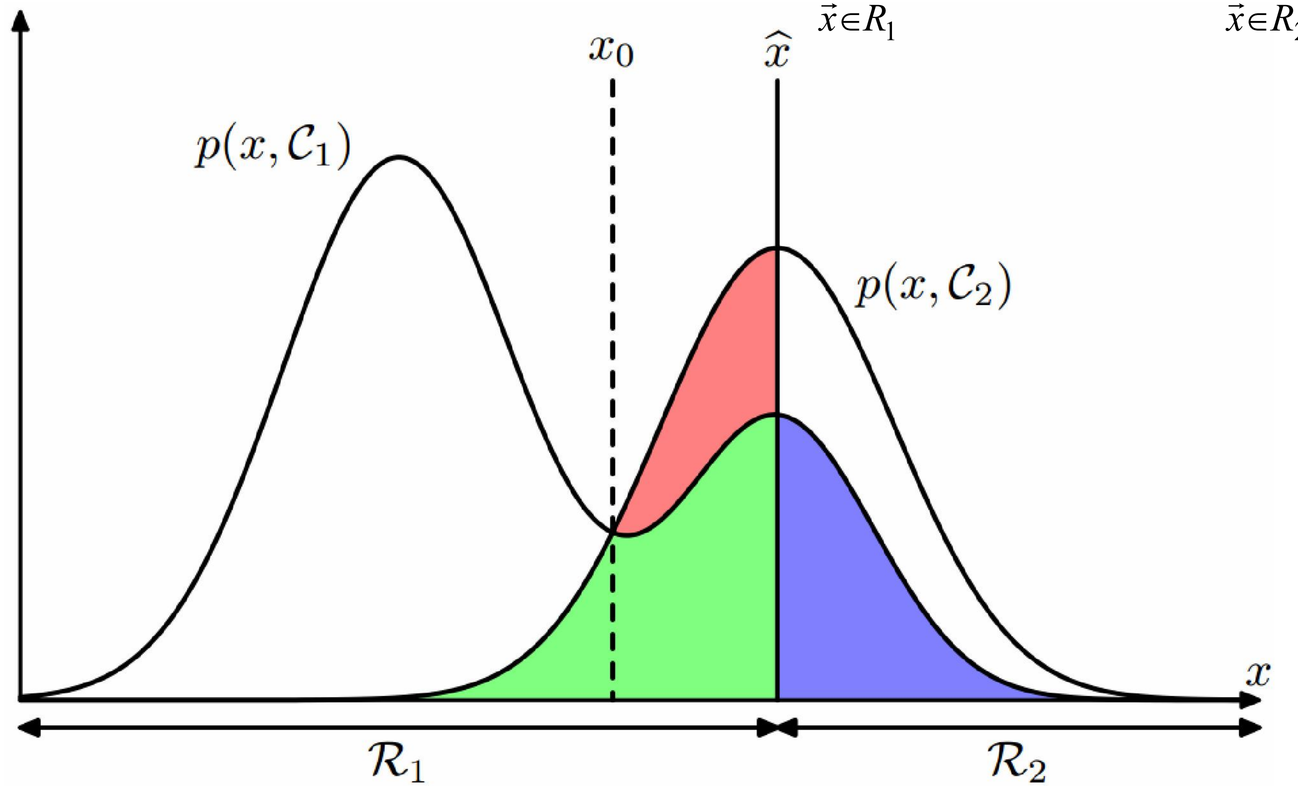
```
def lda(data):
    n = len(data[0]) - 1
    m1 = [0 for x in range(n)]
    m2 = [0 for x in range(n)]
    m = [0 for x in range(n)]
    number1 = 0
    number2 = 0
    for d in data:
        if d[n] == 1:
            add(m1, d)
            number1 += 1
        elif d[n] == 2:
            add(m2, d)
            number2 += 1
    divide(m1, number1)
    divide(m2, number2)
    print m1,m2

    sw = [[] for x in range(n)]
    for i in range(n):
        sw[i] = [0 for x in range(n)]
    calc_sw(data, sw, m1, 1)
    calc_sw(data, sw, m2, 2)
    normal_matrix(sw)
    print "Sw矩阵：", sw
    r = linalg.inv(sw)
    print "逆矩阵：", r
    diff(m1, m2, m)
    normal_vector(m)
    m = multiply(r, m)
    normal_vector(m)
    return m
```

分类step1：极大似然估计

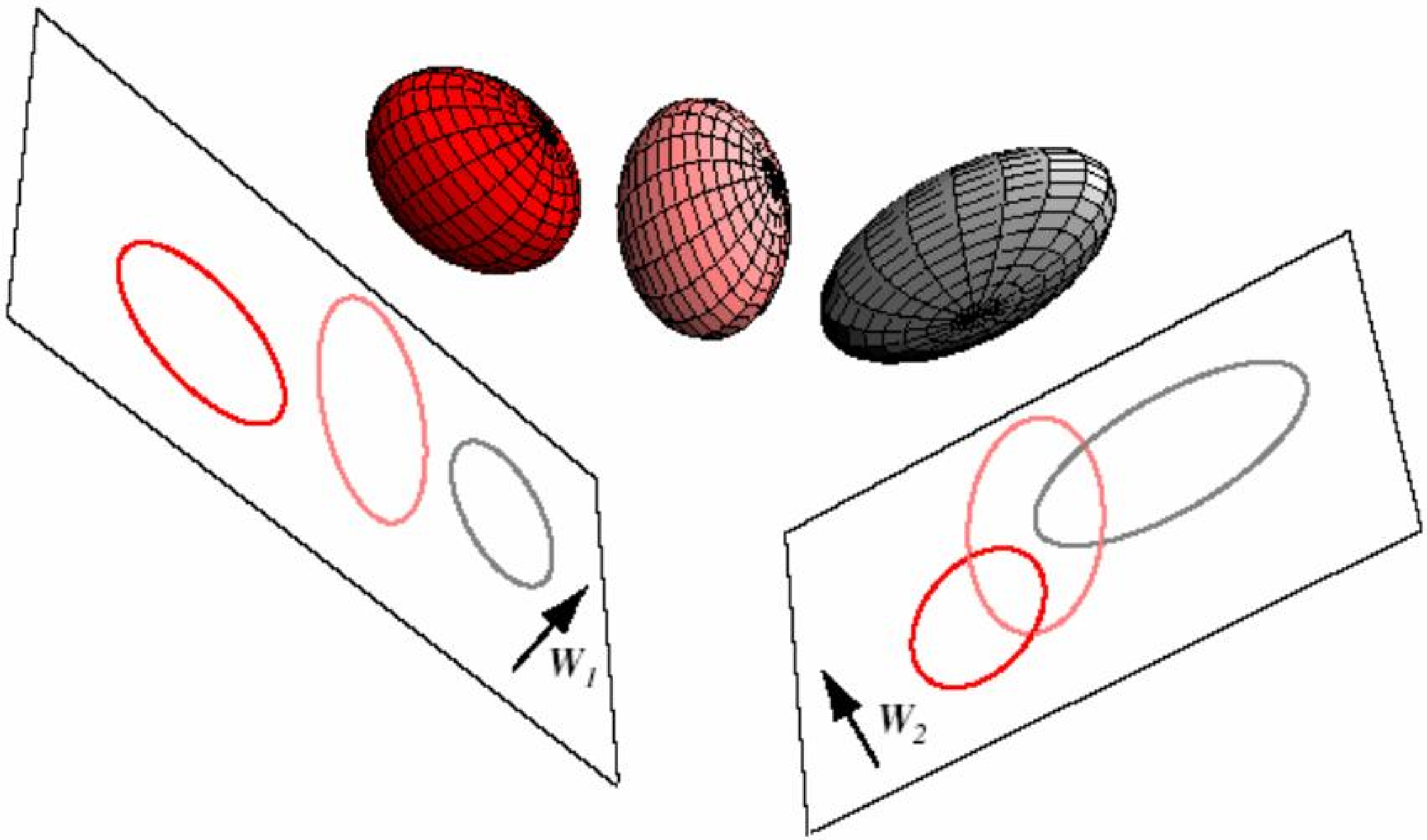


分类step2：误判率准则

$$p(error) = p(\vec{x} \in R_1, C_2) + p(\vec{x} \in R_2, C_1) = \int_{\vec{x} \in R_1} p(\vec{x}, C_2) d\vec{x} + \int_{\vec{x} \in R_2} p(\vec{x}, C_1) d\vec{x}$$

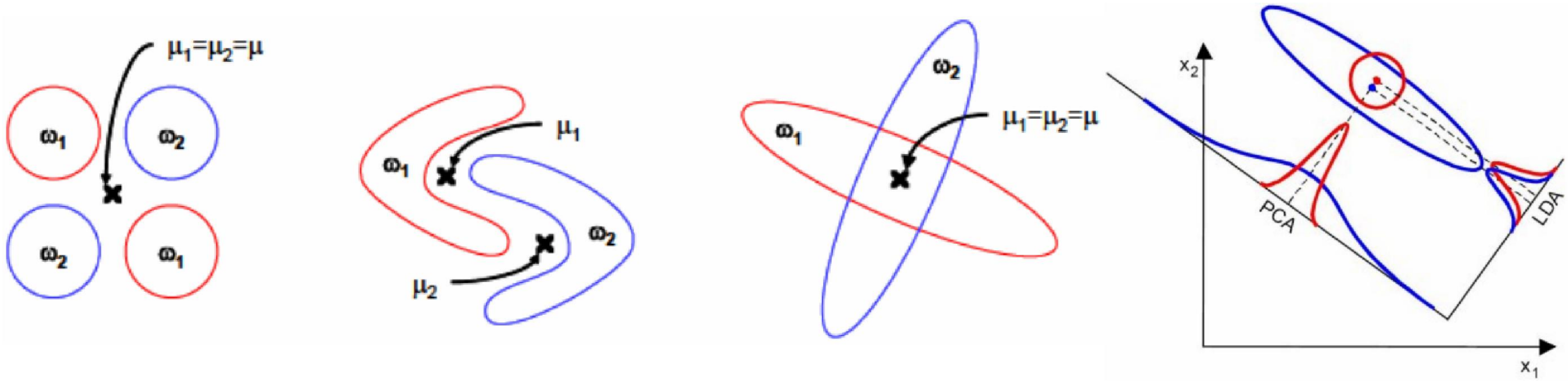


使用LDA将样本投影到平面上



LDA特点

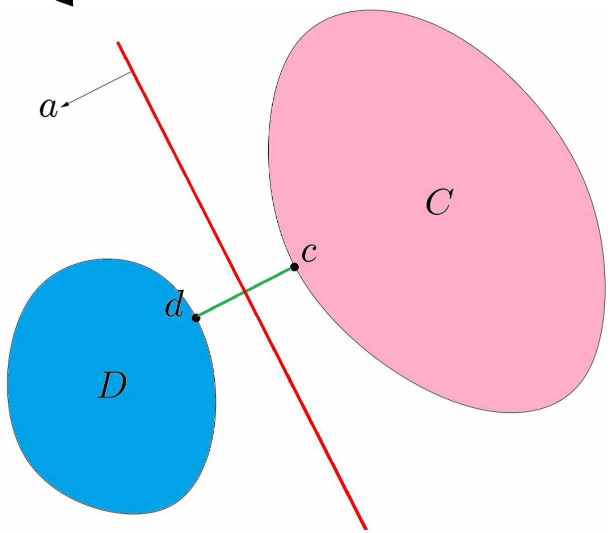
- 由LDA的计算公式看出，LDA是强依赖均值的。如果类别之间的均值相差不大或者需要方差等高阶矩来分类，效果一般。
- 若均值无法有效代表概率分布，LDA效果一般。
  - LDA适用于类别是高斯分布的分类。



LDA与线性回归的关系

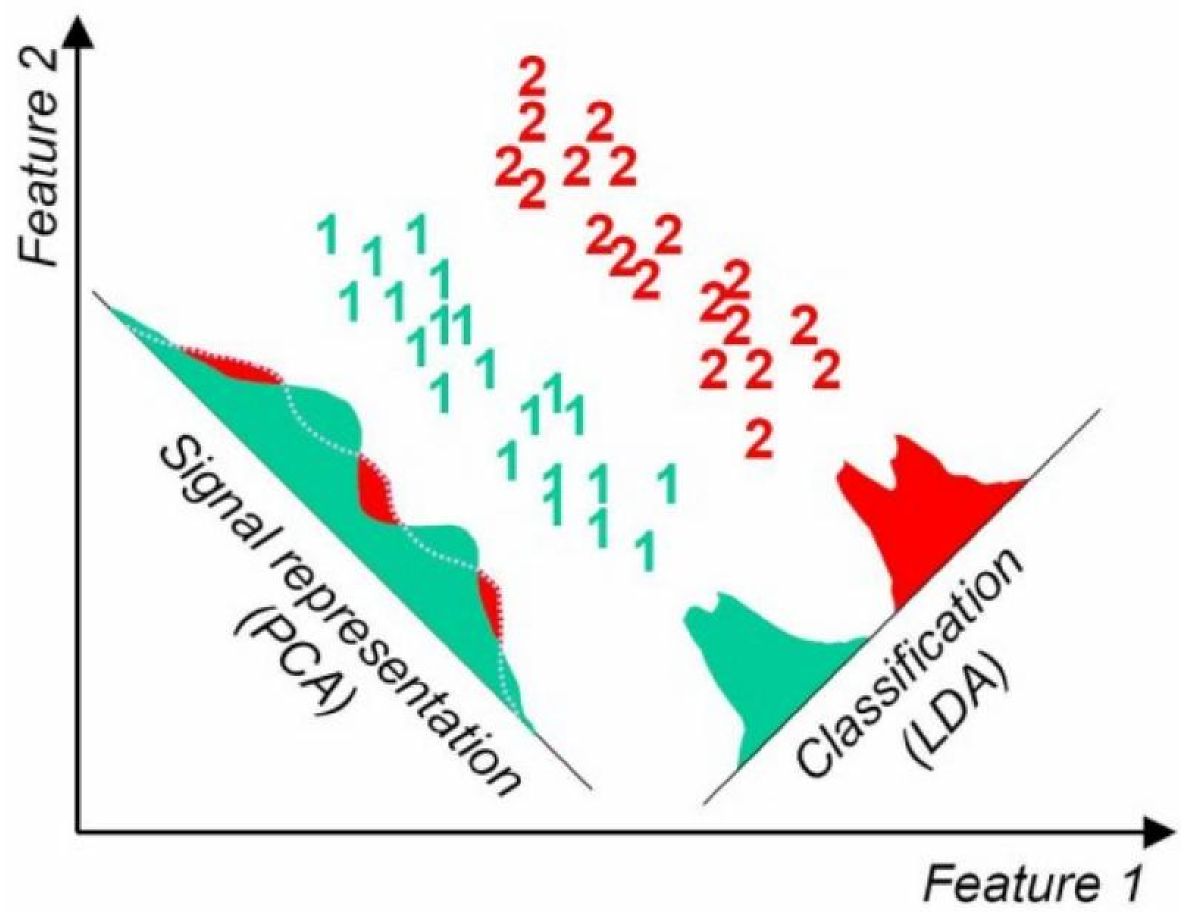
- 假定正样本$(x,1)^{(i)}$个数为N_1，负样本$(x,-1)^{(i)}$个数为N_2；将标记加权成正样本$(x,1/N_1)^{(i)}$，负样本$(x,-1/N_2)^{(i)}$，则使用线性回归得到的决策面方向与LDA相同。

$$\begin{cases}(x,1)^{(i)} \Rightarrow \left(x,\dfrac{1}{N_1}\right)^{(i)} \\ (x,-1)^{(i)} \Rightarrow \left(x,-\dfrac{1}{N_2}\right)^{(i)}\end{cases}$$



LDA与PCA

- LDA：
  - 分类性能最好的方向
- PCA：
  - 样本点投影具有最大方差的方向



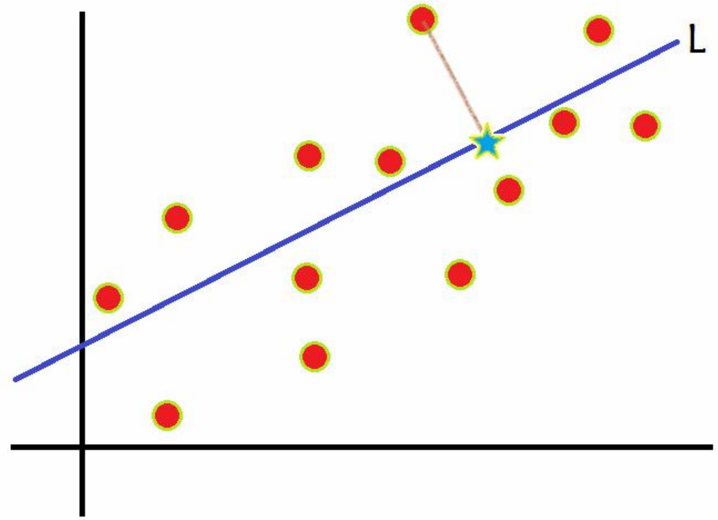
复习：实对称阵特征向量的正交性

- 实对阵矩阵的不同特征值对应的特征向量一定是正交的
- 证明：
- 令实对称矩阵为A，它的两个不同的特征值$\lambda_1 \lambda_2$对应的特征向量分别是$\mu_1 \mu_2$
- 则有：$A\mu_1=\lambda_1\mu_1$， $A\mu_2=\lambda_2\mu_2$
- $(A\mu_1)^T=(\lambda_1\mu_1)^T$，从而：$\mu_1^TA=\lambda_1\mu_1^T$
- 所以：$\mu_1^TA\mu_2=\lambda_1\mu_1^T\mu_2$
- 同时，$\mu_1^TA\mu_2=\mu_1^T(A\mu_2)=\mu_1^T\lambda_2\mu_2=\lambda_2\mu_1^T\mu_2$
- 所以，$\lambda_1\mu_1^T\mu_2=\lambda_2\mu_1^T\mu_2$
- 故：$(\lambda_1-\lambda_2)\mu_1^T\mu_2=0$
- 而$\lambda_1\neq\lambda_2$，所以$\mu_1^T\mu_2=0$，即：μ_1，μ_2正交。

问题的提出

- 实际问题往往需要研究多个特征，而这些特征存在一定的相关性。
  - 数据量增加了问题的复杂性。
- 将多个特征综合为少数几个代表性特征：
  - 既能够代表原始特征的绝大多数信息，
  - 组合后的特征又互不相关，降低相关性。
  - 主成分
- 即主成分分析。

考察降维后的样本方差



- 对于n个特征的m个样本，将每个样本写成行向量，得到矩阵A

$$A=\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m-1,1} & a_{m-1,2} & \cdots & a_{m-1,n} \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}=\begin{pmatrix} a_1^T \\ a_2^T \\ \vdots \\ a_{m-1}^T \\ a_m^T \end{pmatrix}$$

- 思路：寻找样本的主方向**u**：将m个样本值投影到某直线L上，得到m个位于直线L上的点，计算m个投影点的方差。认为方差最大的直线方向是主方向。
  - 假定样本是去均值化的；若没有去均值化，则计算m个样本的均值，将样本真实值减去均值。

计算投影样本点的方差

- 取投影直线L的延伸方向**u**，计算A×**u**的值

$$A \cdot u = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m-1,1} & a_{m-1,2} & \cdots & a_{m-1,n} \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \cdot u = \begin{pmatrix} a_1^T \\ a_2^T \\ \vdots \\ a_{m-1}^T \\ a_m^T \end{pmatrix} \cdot u = \begin{pmatrix} a_1^T \cdot u \\ a_2^T \cdot u \\ \vdots \\ a_{m-1}^T \cdot u \\ a_m^T \cdot u \end{pmatrix}$$

- 求向量A×**u**的方差

$$Var\left(A \cdot u\right) = \left(Au - E\right)^T \left(Au - E\right) = \left(Au\right)^T \left(Au\right) = u^T A^T Au$$

- 目标函数： $J\left(u\right) = \frac{1}{2} u^T A^T Au$

目标函数 $J(u)=\frac{1}{2}u^T A^T Au$

- 由于u数乘得到的方向和u相同，因此，增加u是单位向量的约束，即：$\|u\|_2=1$
- 从而：$\|u\|_2=1 \Rightarrow u^T u=1$
- 建立Lagrange方程：

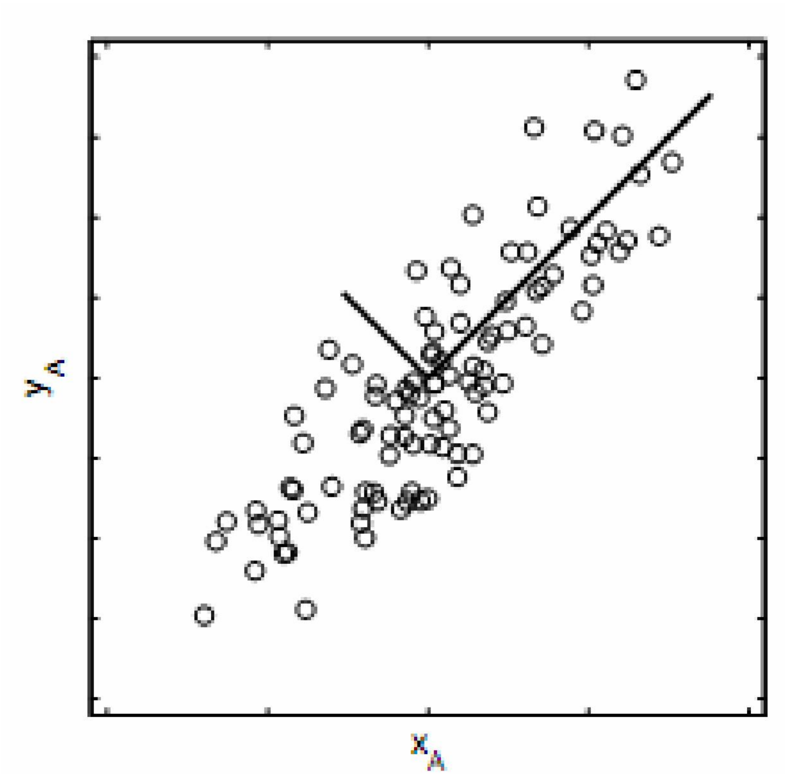
$$L(u)=\frac{1}{2}u^T A^T Au-\lambda\left(u^T u-1\right)$$

$$\frac{\partial L(u)}{\partial u}=A^T Au-\lambda u \overset{令}{=} 0 \Rightarrow \left(A^T A\right)u=\lambda u$$

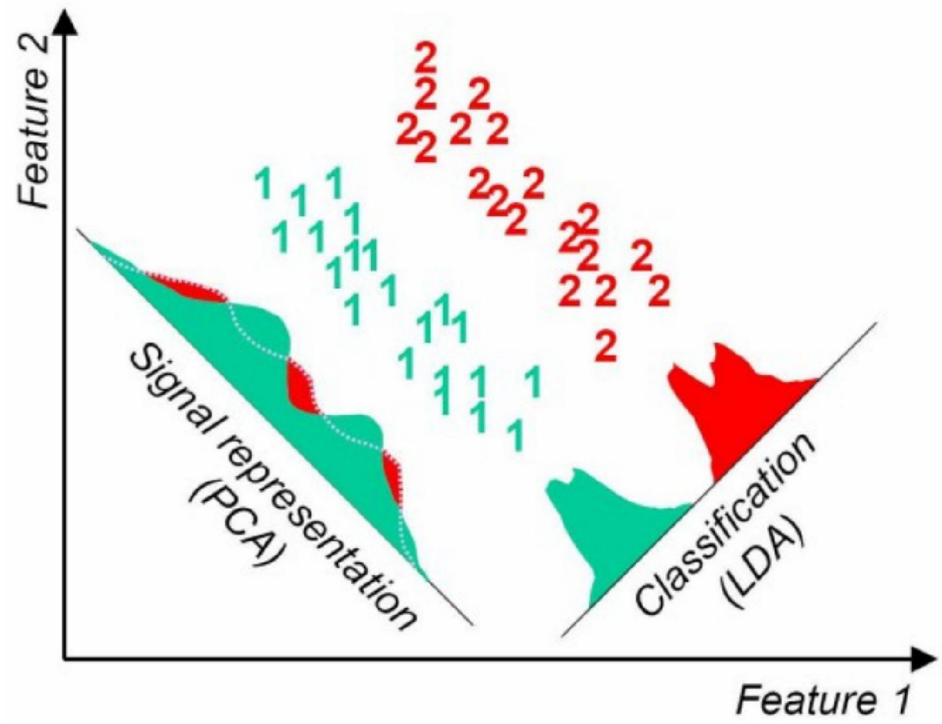
方差和特征值 $A^T Au = \lambda u$

- 若A中的样本都是去均值化的，则A^TA与A的协方差矩阵仅相差系数n-1
  - A^TA常常称为散列矩阵(scatter matrix)
- 根据上式，**u**是A^TA的一个特征向量，λ的值的大小为原始观测数据的特征在向量**u**的方向上投影值的方差。
- 以上即为主成分分析PCA的核心推导过程。

PCA的两个特征向量



y_A
x_A

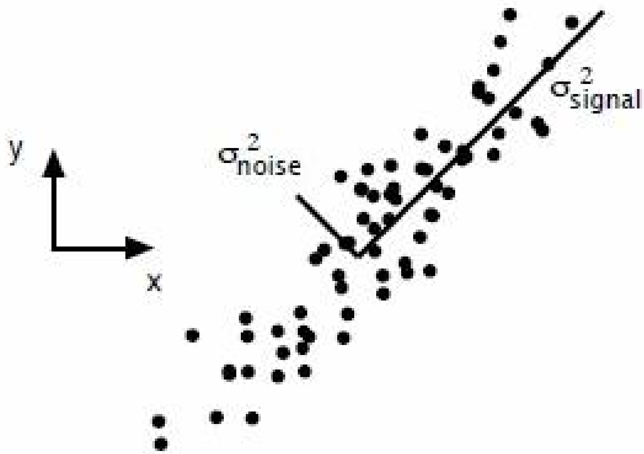


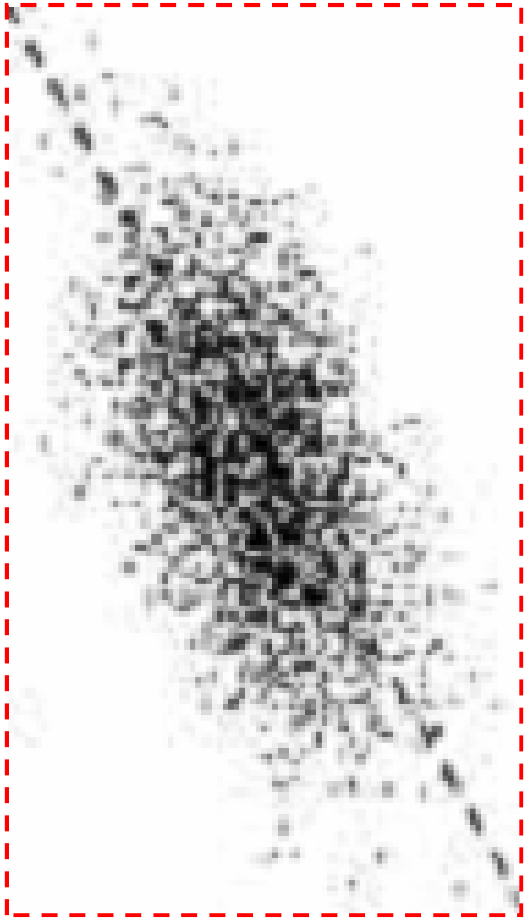
Feature 2
Feature 1
Signal representation (PCA)
Classification (LDA)

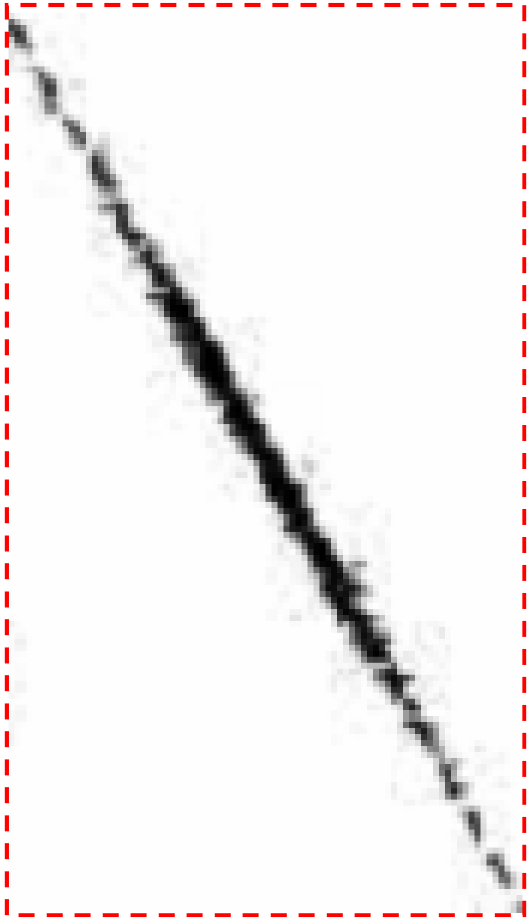
PCA的重要应用

- OBB树
  - Oriented Bounding Box
  - GIS中的空间索引
- 特征提取
- 数据压缩
  - 降维
  - 对原始观测数据A在λ值前k大的特征向量u上投影后，获得一个A(m×n)Q(n×k) 的序列，再加上特征向量矩阵Q，即将A原来的m×n个数据压缩到m×k+k×n个数据。

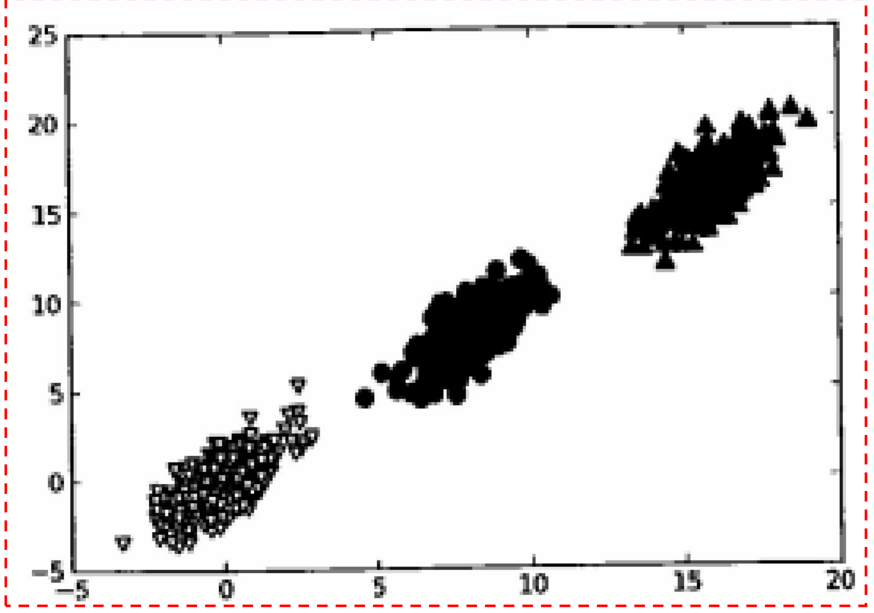
PCA的重要应用——去噪



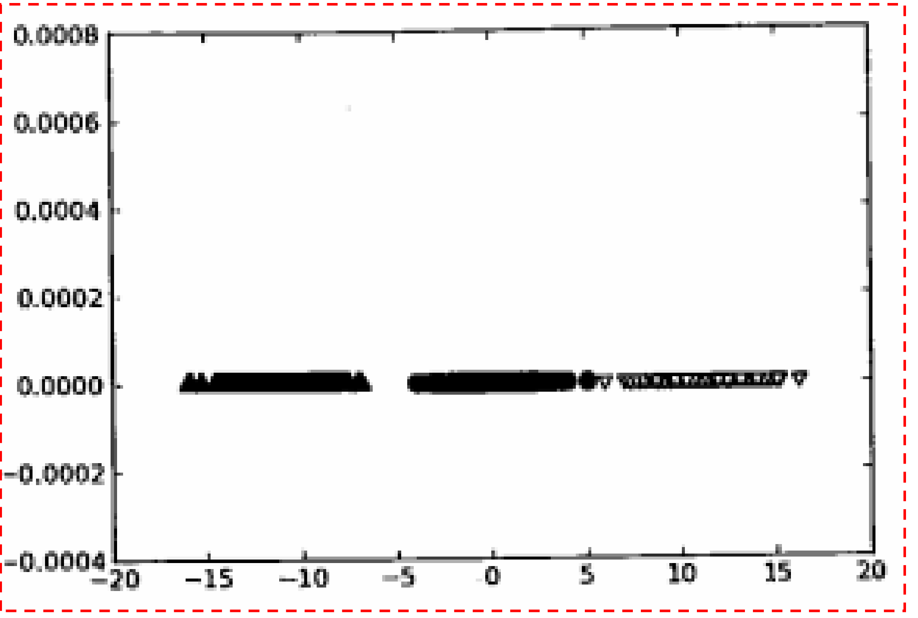




PCA的重要应用——降维



25
20
15
10
5
0
−5
−5
0
5
10
15
20



0.0008
0.0006
0.0004
0.0002
0.0000
−0.0002
−0.0004
−20
−15
−10
−5
0
5
10
15
20

PCA总结

- 实对称阵的特征值一定是实数，不同特征值对应的特征向量一定正交，重数为r的特征值一定有r个线性无关的特征向量；
- 样本矩阵的协方差矩阵必然一定是对称阵，协方差矩阵的元素即各个特征间相关性的度量；
  - 具体实践中考虑是否去均值化；
- 将协方差矩阵C的特征向量组成矩阵P，可以将C合同为对角矩阵D，对角阵D的对角元素即为A的特征值。
  - $P^TCP=D$
  - 协方差矩阵的特征向量，往往单位化，即特征向量的模为1，从而，P是标准正交阵：$P^TP=I$。
  - 即将特征空间线性加权，使得加权后的特征组合间是不相关的。选择若干最大的特征值对应的特征向量(即新的特征组合)，即完成了PCA的过程。

关于PCA的进一步考察

- 若A是m×n阶矩阵，不妨认为m>n，则A^TA是n×n阶方阵。根据下式计算：

$$\left(A^T \cdot A\right)v_i = \lambda_i v_i \Rightarrow \begin{cases} \sigma_i = \sqrt{\lambda_i} \\ u_i = \dfrac{1}{\sigma_i} A \cdot v_i \end{cases} \Rightarrow A = U\Sigma V^T$$

- 从而，将矩阵A可以写成U、V两个方阵和对角矩阵D的乘积，这一过程，称作奇异值分解SVD。

考察结果

$$\left(A^T \cdot A\right)v_i = \lambda_i v_i \Rightarrow \begin{cases} \sigma_i = \sqrt{\lambda_i} \\ u_i = \dfrac{1}{\sigma_i} A \cdot v_i \end{cases}$$

$$U\Sigma V^T = A \cdot \left(\frac{v_1}{\sigma_1}, \frac{v_2}{\sigma_2}, \cdots, \frac{v_n}{\sigma_n}\right) \cdot \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \end{bmatrix} \cdot \begin{pmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_n^T \end{pmatrix}$$

$$= A \cdot \left(v_1, v_2, \cdots, v_n\right) \cdot \begin{bmatrix} \frac{1}{\sigma_1} & & & \\ & \frac{1}{\sigma_2} & & \\ & & \ddots & \\ & & & \frac{1}{\sigma_n} \end{bmatrix} \cdot \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \end{bmatrix} \cdot \begin{pmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_n^T \end{pmatrix} = A\left(v_1, v_2, \cdots, v_n\right) \cdot \begin{pmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_n^T \end{pmatrix}$$

$$= A$$

SVD的提法

$$\left(A^T \cdot A\right)v_i = \lambda_i v_i \Rightarrow \begin{cases} \sigma_i = \sqrt{\lambda_i} \\ u_i = \dfrac{1}{\sigma_i} A \cdot v_i \end{cases} \Rightarrow A = U\Sigma V^T$$

- 奇异值分解(Singular Value Decomposition)是一种重要的矩阵分解方法，可以看做对称方阵在任意矩阵上的推广。
  - Singular：突出的、奇特的、非凡的
  - 似乎更应该称之为“优值分解”
- 假设A是一个m×n阶实矩阵，则存在一个分解使得：

$$A_{m\times n} = U_{m\times m}\Sigma_{m\times n}V^T_{n\times n}$$

  - 通常将奇异值由大而小排列。这样，Σ便能由A唯一确定了。
- 与特征值、特征向量的概念相对应：
  - Σ对角线上的元素称为矩阵A的奇异值；
  - U的第i列称为A的关于σ_i的左奇异向量；
  - V的第i列称为A的关于σ_i的右奇异向量。

SVD举例 $A_{m\times n}=U_{m\times m}\Sigma_{m\times n}V_{n\times n}^T$

- 已知4×5阶实矩阵A，求A的SVD分解：

$$A=\begin{bmatrix}1&0&0&0&2\\0&0&3&0&0\\0&0&0&0&0\\0&4&0&0&0\end{bmatrix}$$

$$U=\begin{bmatrix}0&0&1&0\\0&1&0&0\\0&0&0&1\\1&0&0&0\end{bmatrix}\quad \Sigma=\begin{bmatrix}4&0&0&0&0\\0&3&0&0&0\\0&0&\sqrt{5}&0&0\\0&0&0&0&0\end{bmatrix}\quad V^T=\begin{bmatrix}0&1&0&0&0\\0&0&1&0&0\\\sqrt{0.2}&0&0&0&\sqrt{0.8}\\0&0&0&1&0\\\sqrt{0.8}&0&0&0&-\sqrt{0.2}\end{bmatrix}$$

- 矩阵U和V都是单位正交方阵：$U^TU=I$， $V^TV=I$

奇异值分解不是唯一的

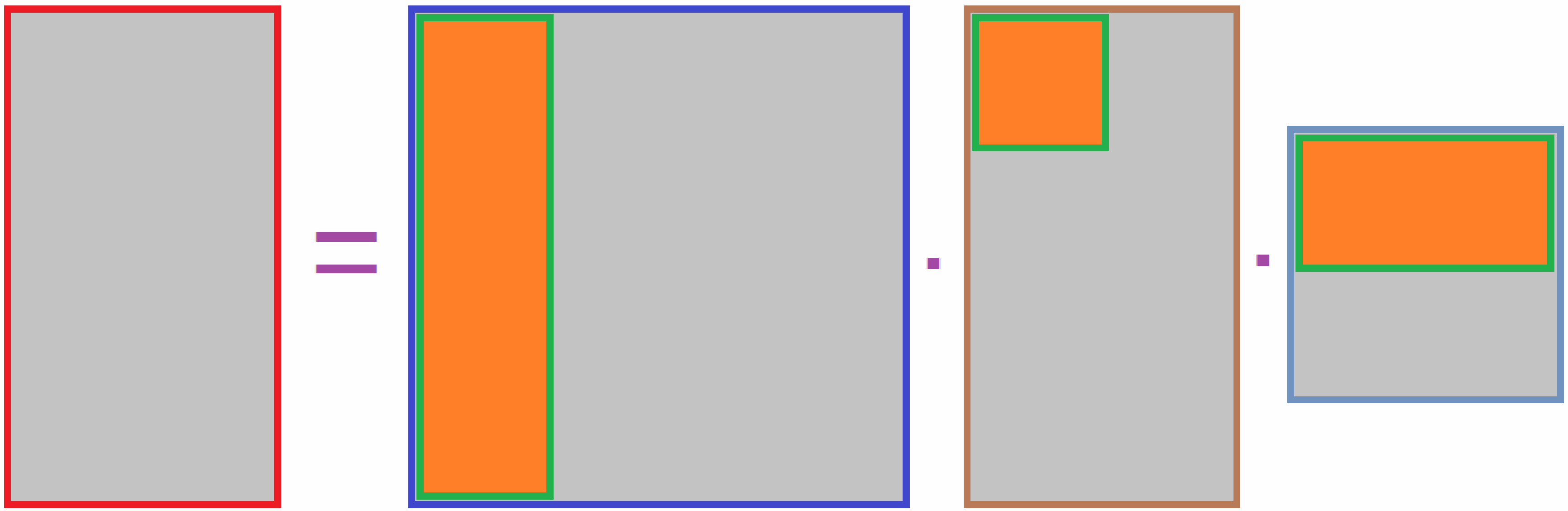
- 由于Σ有一个对角元是零，故这个奇异值分解值不是唯一的。

$$U=\begin{bmatrix}0&0&1&0\\0&1&0&0\\0&0&0&1\\1&0&0&0\end{bmatrix}\quad \Sigma=\begin{bmatrix}4&0&0&0&0\\0&3&0&0&0\\0&0&\sqrt{5}&0&0\\0&0&0&0&0\end{bmatrix}\quad V^T=\begin{bmatrix}0&1&0&0&0\\0&0&1&0&0\\\sqrt{0.2}&0&0&0&\sqrt{0.8}\\0&0&0&1&0\\\sqrt{0.8}&0&0&0&-\sqrt{0.2}\end{bmatrix}$$

$$U=\begin{bmatrix}0&0&1&0\\0&1&0&0\\0&0&0&1\\1&0&0&0\end{bmatrix}\quad \Sigma=\begin{bmatrix}4&0&0&0&0\\0&3&0&0&0\\0&0&\sqrt{5}&0&0\\0&0&0&0&0\end{bmatrix}\quad V^T=\begin{bmatrix}0&1&0&0&0\\0&0&1&0&0\\\sqrt{0.2}&0&0&0&\sqrt{0.8}\\\sqrt{0.4}&0&0&\sqrt{0.5}&-\sqrt{0.1}\\\sqrt{0.4}&0&0&\sqrt{0.5}&-\sqrt{0.1}\end{bmatrix}$$

SVD的四个矩阵 $A_{m\times n} = U_{m\times m}\Sigma_{m\times n}V^T_{n\times n}$

- 实际中，往往只保留Σ前k个较大的数



求伪逆

- 奇异值分解可以被用来计算矩阵的伪逆。若矩阵A的奇异值分解为$A=U\Sigma V^T$，那么A的伪逆为$A^+=V\Sigma^+U^T$
- 其中Σ^+是Σ的伪逆，是将主对角线上每个非零元素都求倒数之后再转置得到的。

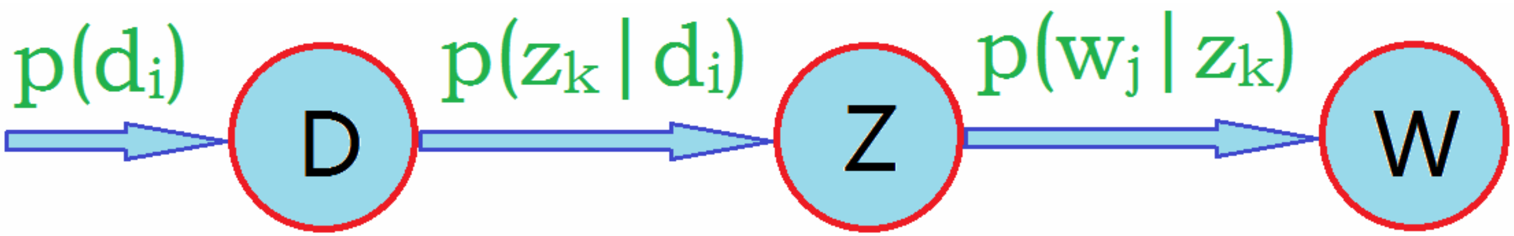
  - 求伪逆通常可以用来求解**最小二乘法**问题。

广义逆矩阵(伪逆)

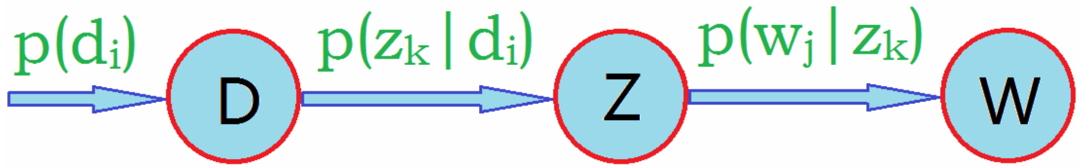
- 若A为非奇异矩阵,则线性方程组Ax=b的解为x=A^(-1)b，其中A的A的逆矩阵A^(-1)满足A^(-1)A=AA^(-1)=I(I为单位矩阵)。若A是奇异阵或长方阵, $x=A^+b$。A^+叫做A的伪逆阵。
- 1955年R.彭罗斯证明了对每个m×n阶矩阵A,都存在惟一的n×m阶矩阵X，满足：①AXA=A；②XAX=X；③(AX)* = I；④(XA)* = I。通常称X为A的穆尔-彭罗斯广义逆矩阵,简称M-P逆，记作A^+。
- 在矛盾线性方程组Ax = b的最小二乘解中, $x=A^+b$是范数最小的一个解。
  - 统一前文使用极大似然得到的公式: $\theta = (X^TX)^{-1}X^T\vec{y}$

SVD与pLSA

- 基于概率统计的pLSA模型(probabilistic latent semantic analysis, 概率隐语义分析)，增加了主题模型，形成简单的贝叶斯网络，可以使用EM算法学习模型参数。

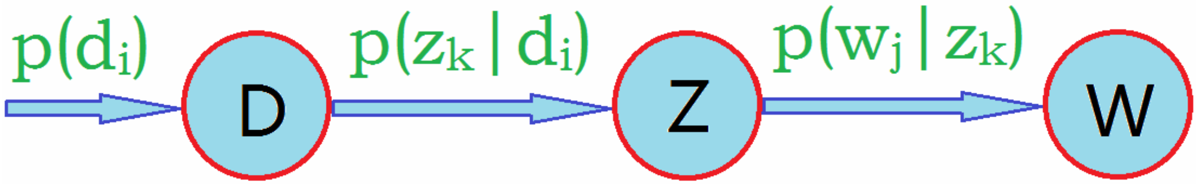


附：参数含义



- D代表文档，Z代表主题(隐含类别)，W代表单词；
  - $P(d_i)$表示文档d_i的出现概率，
  - $P(z_k|d_i)$表示文档d_i中主题z_k的出现概率，
  - $P(w_j|z_k)$表示给定主题z_k出现单词w_j的概率。
- 每个主题在所有词项上服从多项分布，每个文档在所有主题上服从多项分布。
- 整个文档的生成过程是这样的：
  - 以$P(d_i)$的概率选中文档d_i；
  - 以$P(z_k|d_i)$的概率选中主题z_k；
  - 以$P(w_j|z_k)$的概率产生一个单词w_j。

pLSA模型



- 观察数据为(d_i, w_j)对，主题z_k是隐含变量。
- (d_i, w_j)的联合分布为

$$P(d_i, w_j) = P(w_j \mid d_i) P(d_i)$$

$$P(w_j \mid d_i) = \sum_{k=1}^{K} P(w_j \mid z_k) P(z_k \mid d_i)$$

- 而 $P(w_j \mid z_k), P(z_k \mid d_i)$ 对应了两组多项分布，而计算每个文档的主题分布，就是该模型的任务目标。
- 事实上，上式即为矩阵相乘的公式。因此pLSA可以看做是概率化的矩阵分解。$A_{m \times n} = U_{m \times m} \Sigma_{m \times n} V^T_{n \times n}$

SVD举例

- 假定Ben、Tom、John、Fred对6种产品进行了评价，评分越高，代表对该产品越喜欢。0表示未评价。

	Ben	Tom	John	Fred
Season 1	5	5	0	5
Season 2	5	0	3	4
Season 3	3	4	0	3
Season 4	0	0	5	3
Season 5	5	4	4	5
Season 6	5	4	5	5

评分矩阵

	Ben	Tom	John	Fred
Season 1	5	5	0	5
Season 2	5	0	3	4
Season 3	3	4	0	3
Season 4	0	0	5	3
Season 5	5	4	4	5
Season 6	5	4	5	5

$$A=\begin{bmatrix}5&5&0&5\\5&0&3&4\\3&4&0&3\\0&0&5&3\\5&4&4&5\\5&4&5&5\end{bmatrix}$$

SVD分解 $A_{m\times n}=U_{m\times m}\Sigma_{m\times n}V_{n\times n}^T$

$$A=\begin{bmatrix}5&5&0&5\\5&0&3&4\\3&4&0&3\\0&0&5&3\\5&4&4&5\\5&4&5&5\end{bmatrix}\quad U_{6\times 6}=\begin{bmatrix}-0.4472&-0.5373&-0.0064&-0.5037&-0.3857&-0.3298\\-0.3586&0.2461&0.8622&-0.1458&0.0780&0.2002\\-0.2925&-0.4033&-0.2275&-0.1038&0.4360&0.7065\\-0.2078&0.6700&-0.3951&-0.5888&0.0260&0.0667\\-0.5099&0.0597&-0.1097&0.2869&0.5946&-0.5371\\-0.5316&0.1887&-0.1914&0.5341&-0.5485&0.2429\end{bmatrix}$$

$$\Sigma_{6\times 4}=\begin{bmatrix}17.7139&0&0&0\\0&6.3917&0&0\\0&0&3.0980&0\\0&0&0&1.3290\\0&0&0&0\\0&0&0&0\end{bmatrix}\quad V_{4\times 4}^T=\begin{bmatrix}-0.5710&-0.2228&0.6749&0.4109\\-0.4275&-0.5172&-0.6929&0.2637\\-0.3846&0.8246&-0.2532&0.3286\\-0.5859&0.0532&0.0140&-0.8085\end{bmatrix}$$

SVD分解，取k=2 $A_{m\times n}=U_{m\times m}\Sigma_{m\times n}V^T_{n\times n}$

$$A=\begin{bmatrix}5&5&0&5\\5&0&3&4\\3&4&0&3\\0&0&5&3\\5&4&4&5\\5&4&5&5\end{bmatrix}$$

$$U_{6\times 6}=\begin{bmatrix}-0.4472&-0.5373&-0.0064&-0.5037&-0.3857&-0.3298\\-0.3586&0.2461&0.8622&-0.1458&0.0780&0.2002\\-0.2925&-0.4033&-0.2275&-0.1038&0.4360&0.7065\\-0.2078&0.6700&-0.3951&-0.5888&0.0260&0.0667\\-0.5099&0.0597&-0.1097&0.2869&0.5946&-0.5371\\-0.5316&0.1887&-0.1914&0.5341&-0.5485&0.2429\end{bmatrix}$$

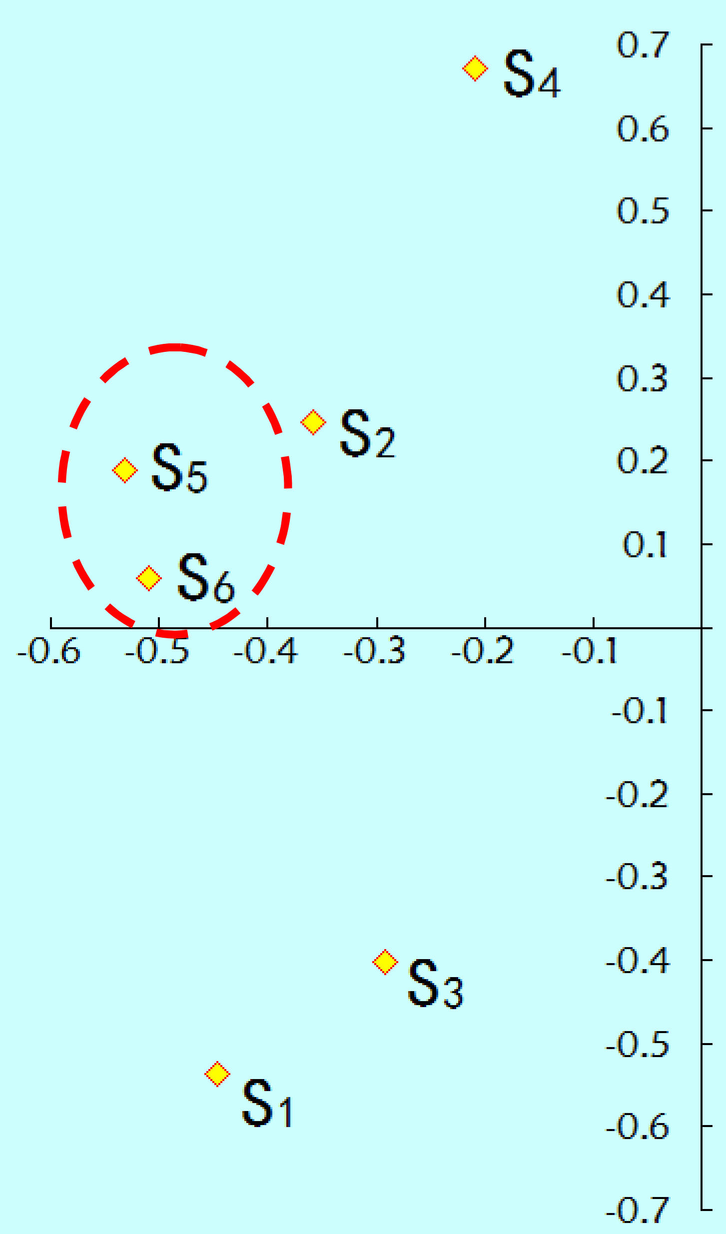
$$\Sigma_{6\times 4}=\begin{bmatrix}17.7139&0&0&0\\0&6.3917&0&0\\0&0&3.0980&0\\0&0&0&1.3290\\0&0&0&0\\0&0&0&0\end{bmatrix}$$

$$V^T_{4\times 4}=\begin{bmatrix}-0.5710&-0.2228&0.6749&0.4109\\-0.4275&-0.5172&-0.6929&0.2637\\-0.3846&0.8246&-0.2532&0.3286\\-0.5859&0.0532&0.0140&-0.8085\end{bmatrix}$$

产品矩阵的压缩

$$U_{6\times6}=\begin{bmatrix}-0.4472 & -0.5373 & -0.0064 & -0.5037\\ -0.3586 & 0.2461 & 0.8622 & -0.1458\\ -0.2925 & -0.4033 & -0.2275 & -0.1038\\ -0.2078 & 0.6700 & -0.3951 & -0.5888\\ -0.5099 & 0.0597 & -0.1097 & 0.2869\\ -0.5316 & 0.1887 & -0.1914 & 0.5341\end{bmatrix}$$

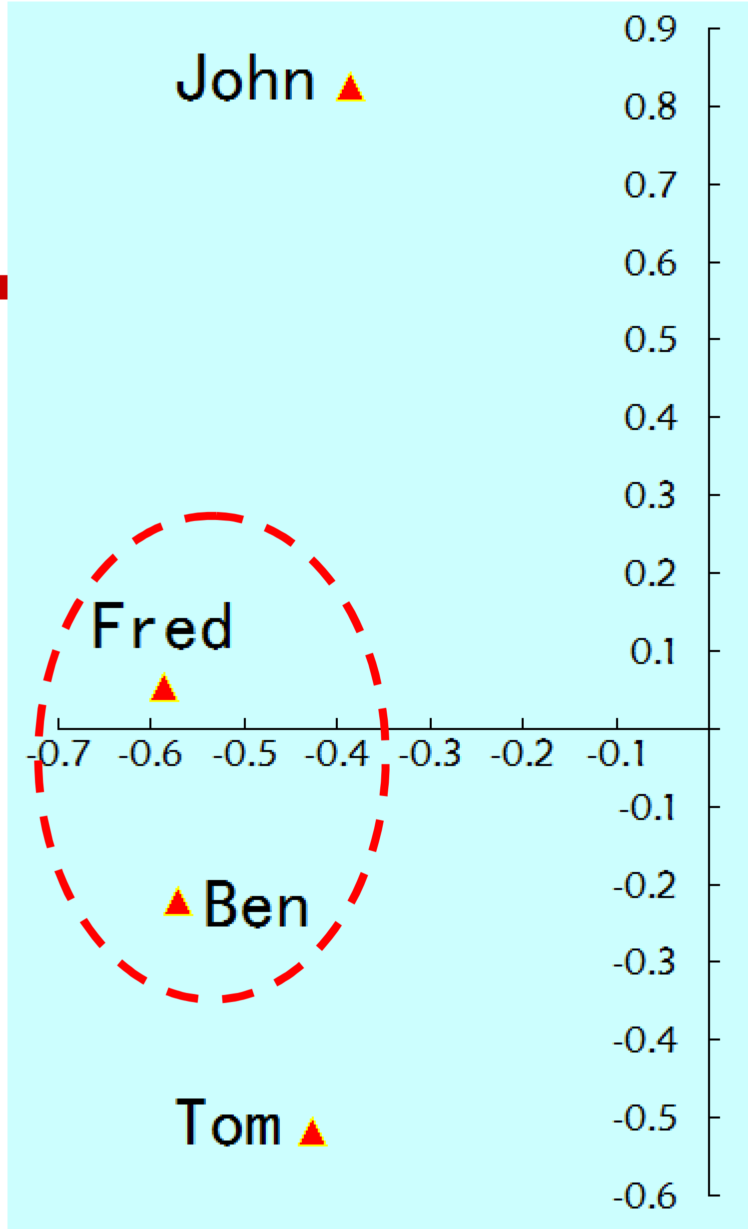
$$A=\begin{bmatrix}5 & 5 & 0 & 5\\ 5 & 0 & 3 & 4\\ 3 & 4 & 0 & 3\\ 0 & 0 & 5 & 3\\ 5 & 4 & 4 & 5\\ 5 & 4 & 5 & 5\end{bmatrix}$$



用户矩阵的压缩

$$V_{4\times4}^{T}=\begin{bmatrix} -0.5710 & -0.2228 & 0.6749 & 0.4109 \\ -0.4275 & -0.5172 & -0.6929 & 0.2637 \\ -0.3846 & 0.8246 & -0.2532 & 0.3286 \\ -0.5859 & 0.0532 & 0.0140 & -0.8085 \end{bmatrix}$$

$$A=\begin{bmatrix} 5 & 5 & 0 & 5 \\ 5 & 0 & 3 & 4 \\ 3 & 4 & 0 & 3 \\ 0 & 0 & 5 & 3 \\ 5 & 4 & 4 & 5 \\ 5 & 4 & 5 & 5 \end{bmatrix}$$



新用户的个性化推荐

- 对于新用户，如何对其做个性化推荐呢？
  - 将A扩展后重新计算SVD，然后聚类用户？
  - 事实上

$$A = U \cdot \Sigma \cdot V^T$$
$$\Rightarrow U^T A = U^T U \cdot \Sigma \cdot V^T$$
$$\Rightarrow U^T A = \Sigma \cdot V^T$$
$$\Rightarrow \Sigma^{-1} U^T A = \Sigma^{-1} \Sigma \cdot V^T$$
$$\Rightarrow \Sigma^{-1} U^T A = V^T$$
$$\Rightarrow \left(\Sigma^{-1} U^T A\right)^T = V$$
$$\Rightarrow A^T U \Sigma^{-1} = V$$

新用户的个性化推荐 $V = A^T \cdot U \cdot \Sigma^{-1}$

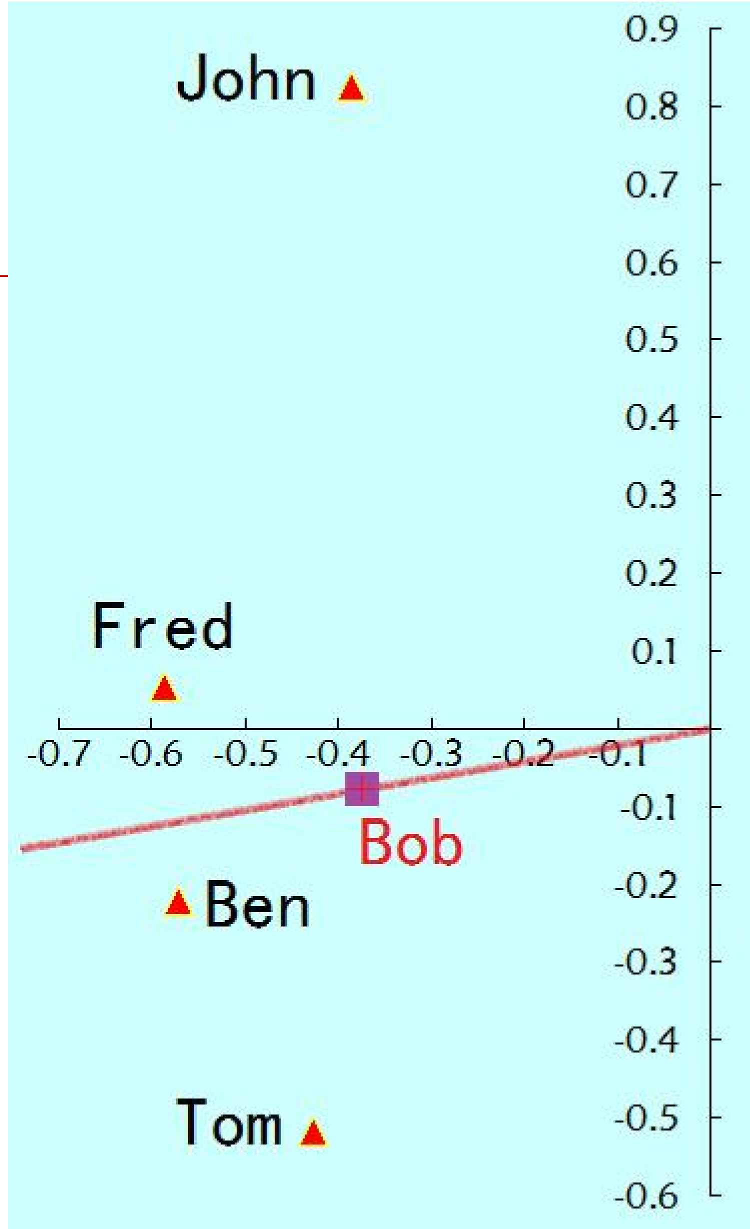
- 假设有个Bob的新用户，对6个产品的评分为 $(5,5,0,0,0,5)^{\mathrm{T}}$，则：

$$V = a^T \cdot U \cdot \Sigma^{-1} = (5,5,0,0,0,5) \cdot \begin{bmatrix} -0.4472 & -0.5373 \\ -0.3586 & 0.2461 \\ -0.2925 & -0.4033 \\ -0.2078 & 0.6700 \\ -0.5099 & 0.0597 \\ -0.5316 & 0.1887 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{17.7139} & 0 \\ 0 & \frac{1}{6.3917} \end{bmatrix}$$

$$= (-0.3775, -0.0802)$$

个性化推荐

- 计算新加的Bob和现有用户的距离：余弦距离(一定意义下即相关系数)，最近的是Ben。
- Ben：5 5 3 0 5 5
- Bob：5 5 0 0 0 5
  - 因此，可顺次推荐S5、S3



PCA和SVD总结

- 矩阵对向量的乘法，对应于对该向量的旋转、伸缩。如果对某向量只发生了伸缩而无旋转变化，则该向量是该矩阵的特征向量，伸缩比即为特征值。
  - PCA用来提取一个场的主要信息(即主成分分量)，而SVD一般用来分析两个场的相关关系。两者在具体的实现方法上也有不同，SVD是通过矩阵奇异值分解的方法分解两个场的协方差矩阵的，而PCA是通过分解一个场的协方差矩阵。
  - PCA可用于特征的压缩、降维；当然也能去噪等；如果将矩阵转置后再用PCA，相当于去除相关度过大的样本数据——但不常见；SVD能够对一般矩阵分解，并可用于个性化推荐等内容。

参考文献

- Pattern Recognition and Machine Learning Chapter 10, Christopher M. Bishop, Springer-Verlag, 2006
- Machine Learning:A Probabilistic Perspective, Kevin P. Murphy, The MIT Press, 2012
- Prof. Andrew Ng, Machine Learning, Stanford University

我们在这里

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！