# 第三章 线性模型

周世斌

中国矿业大学　计算机学院

May. 2022

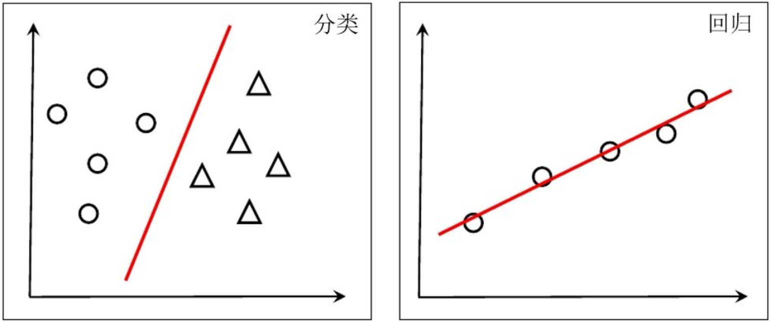
1. 引言
2. 线性模型（最小二乘法求解）
   - 多元线性回归
   - 最优化方法
   - Widrow-Hoff 算法
   - 广义线性模型
3. 主成份分析，降维方法选讲
   - 主成份分析 (Principal Component Analysis)
   - 特征脸
   - 简单的人脸识别系统
4. 线性判别分析
5. 最大间隔准则与支持向量机
6. Shrinkage Methods and Regularization

因此，给定训练样本集合 $\{\mathbf{x}_i, y_i\}_{i=1}^n$，求出决策面

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b$$

就是求参数 $\mathbf{w}, b$，称为线性模型的参数估计，

# 线性模型




线性模型(linear model)试图学得一个通过属性的线性组合来进行预测的函数

$$f(\boldsymbol{x}) = w_1 x_1 + w_2 x_2 + \ldots + w_d x_d + b$$

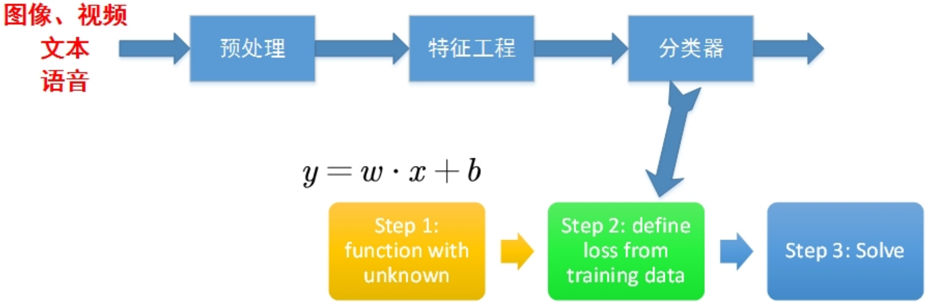
向量形式: $f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x} + b$

简单、基本、可理解性好

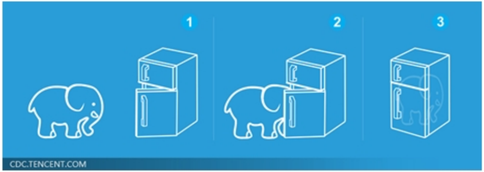
# 学习规则 or 损失函数

- 误差修正学习，最小二乘法
- 最大间隔准则法
- 感知器准则
- hebb 准则
- ...

# Machine Learning is so simple ……




$$y = w \cdot x + b$$



# 线性回归

$$\boxed{f(x) = wx_i + b \quad \text{使得} \quad f(x_i) \simeq y_i}$$

离散属性的处理：若有“序”(order)，则连续化；
否则，转化为 k 维向量

令均方误差最小化，有
$$\begin{aligned}(w^*, b^*) &= \underset{(w,b)}{\arg\min} \sum_{i=1}^{m} \left(f(x_i) - y_i\right)^2 \\ &= \underset{(w,b)}{\arg\min} \sum_{i=1}^{m} \left(y_i - wx_i - b\right)^2\end{aligned}$$

对 $E_{(w,b)} = \sum_{i=1}^{m} \left(y_i - wx_i - b\right)^2$ 进行最小二乘参数估计

# 线性回归

分别对 $w$ 和 $b$ 求导：

$$\frac{\partial E_{(w,b)}}{\partial w} = 2\left(w\sum_{i=1}^{m} x_i^2 - \sum_{i=1}^{m}(y_i - b)\,x_i\right)$$

$$\frac{\partial E_{(w,b)}}{\partial b} = 2\left(mb - \sum_{i=1}^{m}(y_i - wx_i)\right)$$

令导数为 0, 得到闭式(closed-form)解：

$$w = \frac{\sum_{i=1}^{m} y_i\,(x_i - \bar{x})}{\sum_{i=1}^{m} x_i^2 - \frac{1}{m}\left(\sum_{i=1}^{m} x_i\right)^2} \qquad b = \frac{1}{m}\sum_{i=1}^{m}(y_i - wx_i)$$

# 最小二乘法：以回归为例

We want to seek the understanding from data set $\{\mathbf{x}_i, y_i\}_{i=1}^N$. 当 $y$ 为连续量的时候，称为回归。
最小二乘的方法通过最小化偏离数据的误差平方和来选择参数 $(\mathbf{w}, b)$。
误差平方和为

$$E(\mathbf{w}, w_0) = \sum_{i=1}^{N} \left(y_i - \mathbf{w}^{\mathrm{T}} \mathbf{x}_i - b\right)^2$$

为方便起见，就用 $\widehat{\mathbf{w}}$ 代替原来的 $\begin{pmatrix} \mathbf{w} \\ b \end{pmatrix}$。用 $\widehat{\mathbf{x}}_i$ 代替原来的 $\begin{pmatrix} \mathbf{x}_i \\ 1 \end{pmatrix}$ 此时新的 $\widehat{\mathbf{w}}, \widehat{\mathbf{x}}_i$ 为增广参数向量和增广样本向量。此时误差平方和为

$$E(\widehat{\mathbf{w}}) = \sum_{i=1}^{N} \left(y_i - \widehat{\mathbf{w}}^{\mathrm{T}} \widehat{\mathbf{x}}_i\right)^2$$

$$E_{\widehat{\mathbf{w}}} = \left(\mathbf{y} - \mathbf{X}\widehat{\mathbf{w}}\right)^{\mathrm{T}} \left(\mathbf{y} - \mathbf{X}\widehat{\mathbf{w}}\right)$$

# 多元线性回归

$$\boxed{f(\boldsymbol{x}_i) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b \quad \text{使得} \quad f(\boldsymbol{x}_i) \simeq y_i}$$

$$\boldsymbol{x}_i = (x_{i1}; x_{i2}; \ldots; x_{id}) \qquad y_i \in \mathbb{R}$$

把 $\boldsymbol{w}$ 和 $b$ 吸收入向量形式 $\hat{\boldsymbol{w}} = (\boldsymbol{w}; b)$，数据集表示为

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1d} & 1 \\ x_{21} & x_{22} & \cdots & x_{2d} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{md} & 1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{x}_1^{\mathrm{T}} & 1 \\ \boldsymbol{x}_2^{\mathrm{T}} & 1 \\ \vdots & \vdots \\ \boldsymbol{x}_m^{\mathrm{T}} & 1 \end{pmatrix} \qquad \boldsymbol{y} = (y_1; y_2; \ldots; y_m)$$

# 多元线性回归

同样采用最小二乘法求解，有

$$\hat{\boldsymbol{w}}^* = \arg\min_{\hat{\boldsymbol{w}}} \left(\boldsymbol{y} - \mathbf{X}\hat{\boldsymbol{w}}\right)^{\mathrm{T}} \left(\boldsymbol{y} - \mathbf{X}\hat{\boldsymbol{w}}\right)$$

令 $E_{\hat{\boldsymbol{w}}} = \left(\boldsymbol{y} - \mathbf{X}\hat{\boldsymbol{w}}\right)^{\mathrm{T}} \left(\boldsymbol{y} - \mathbf{X}\hat{\boldsymbol{w}}\right)$ ，对 $\hat{\boldsymbol{w}}$ 求导：

$$\frac{\partial E_{\hat{\boldsymbol{w}}}}{\partial \hat{\boldsymbol{w}}} = 2\mathbf{X}^{\mathrm{T}} \left(\mathbf{X}\hat{\boldsymbol{w}} - \boldsymbol{y}\right)$$ 令其为零可得 $\hat{\boldsymbol{w}}$

然而，麻烦来了：涉及矩阵求逆！

- 若 $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ 满秩或正定，则 $\hat{\boldsymbol{w}}^* = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\boldsymbol{y}$
- 若 $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ 不满秩，则可解出多个 $\hat{\boldsymbol{w}}$

此时需求助于归纳偏好，或引入 正则化 (regularization) $\longrightarrow$

# 矩阵导数, Ref. PRML

## C.3 矩阵的导数

有时，我们需要考虑向量和矩阵关于标量的导数。向量$\boldsymbol{a}$关于标量$x$的导数本身是一个向量，它的分量为

$$\left(\frac{\partial \boldsymbol{a}}{\partial x}\right)_i = \frac{\partial a_i}{\partial x} \tag{C.16}$$

矩阵的导数的定义与此类似。关于向量和矩阵的导数也可以被定义。例如

$$\left(\frac{\partial x}{\partial \boldsymbol{a}}\right)_i = \frac{\partial x}{\partial a_i} \tag{C.17}$$

类似地

$$\left(\frac{\partial \boldsymbol{a}}{\partial \boldsymbol{b}}\right)_{ij} = \frac{\partial a_i}{\partial b_j} \tag{C.18}$$

写出矩阵的各个元素，下面的性质很容易证明

$$\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{x}^T\boldsymbol{a}) = \frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{a}^T\boldsymbol{x}) = \boldsymbol{a} \tag{C.19}$$

# 矩阵导数

类似地

$$\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{A}\boldsymbol{B}) = \frac{\partial \boldsymbol{A}}{\partial x}\boldsymbol{B} + \boldsymbol{A}\frac{\partial \boldsymbol{B}}{\partial x} \tag{C.20}$$

矩阵的逆矩阵的导数可以表示为

$$\frac{\partial}{\partial x}(\boldsymbol{A}^{-1}) = -\boldsymbol{A}^{-1}\frac{\partial \boldsymbol{A}}{\partial x}\boldsymbol{A}^{-1} \tag{C.21}$$

使用公式（C.20）对方程$\boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{I}$求微分，然后右乘$\boldsymbol{A}^{-1}$即可证明。并且

$$\frac{\partial}{\partial x}\ln|\boldsymbol{A}| = \mathrm{Tr}\left(\boldsymbol{A}^{-1}\frac{\partial \boldsymbol{A}}{\partial x}\right) \tag{C.22}$$

这个我们稍后会证明。如果我们把$x$选成$\boldsymbol{A}$中的元素，那么我们有

$$\frac{\partial}{\partial A_{ij}}\mathrm{Tr}(\boldsymbol{A}\boldsymbol{B}) = B_{ji} \tag{C.23}$$

# 矩阵导数

写出矩阵的下标即可证明这个等式。我们可以把这个结论写成更加简洁的形式

$$\frac{\partial}{\partial \boldsymbol{A}} \operatorname{Tr}(\boldsymbol{A}\boldsymbol{B}) = \boldsymbol{B}^T \tag{C.24}$$

使用这种记号，我们有下列性质

$$\frac{\partial}{\partial \boldsymbol{A}} \operatorname{Tr}(\boldsymbol{A}^T\boldsymbol{B}) = \boldsymbol{B} \tag{C.25}$$

$$\frac{\partial}{\partial \boldsymbol{A}} \operatorname{Tr}(\boldsymbol{A}) = \boldsymbol{I} \tag{C.26}$$

$$\frac{\partial}{\partial \boldsymbol{A}} \operatorname{Tr}(\boldsymbol{A}\boldsymbol{B}\boldsymbol{A}^T) = \boldsymbol{A}(\boldsymbol{B} + \boldsymbol{B}^T) \tag{C.27}$$

这些也可以通过写出矩阵下标的方式证明出。我们也有

$$\frac{\partial}{\partial \boldsymbol{A}} \ln |\boldsymbol{A}| = (\boldsymbol{A}^{-1})^T \tag{C.28}$$

根据公式（C.22）和公式（C.24）即可证得。

# 多元线性回归

进一步得到

$$E_{\widehat{\mathbf{w}}} = \left(\mathbf{y} - \mathbf{X}\widehat{\mathbf{w}}\right)^{\mathrm{T}}\left(\mathbf{y} - \mathbf{X}\widehat{\mathbf{w}}\right)$$

☞思考，$\mathbf{X}$ 是什么？(Design Matrix)

$$\frac{\partial E}{\partial \widehat{\mathbf{w}}} = -2\mathbf{X}^{\mathrm{T}}\left(\mathbf{y} - \mathbf{X}\widehat{\mathbf{w}}\right) = -2\mathbf{X}^{\top}\mathbf{y} + 2\mathbf{X}^{\mathrm{T}}\mathbf{X}\widehat{\mathbf{w}} = 0$$

$$\Longrightarrow \widehat{\mathbf{w}} = \left(\mathbf{X}^{\top}\mathbf{X}\right)^{-1}\mathbf{X}^{\top}\mathbf{y}$$

☞此算法的缺陷在哪里？

```
from numpy import arange,array,ones,linalg
import matplotlib.pyplot as plt

xi = arange(0,9)
A = array([ xi, ones(9)])
# linearly generated sequence
y = [19, 20, 20.5, 21.5, 22, 23, 23, 25.5, 24]
w = linalg.lstsq(A.T,y)[0] # obtaining the parameters

# plotting the line
line = w[0]*xi+w[1] # regression line

plt.scatter(xi, y,  color='black')
plt.plot(xi, line, color='red', linewidth=2)
plt.xticks(())
plt.yticks(())
plt.show()
```

# 梯度下降法

求向量函数 $f(\mathbf{x})$（凸函数）的极小值， $\min f(\mathbf{x})$ 。
由 Taylor 公式，$f(\mathbf{x})$ 可表示为

$$f(\mathbf{x}) = f(\mathbf{x}_k) + \mathbf{g}_k^\top (\mathbf{x} - \mathbf{x}_k) + o(\|\mathbf{x} - \mathbf{x}_k\|)$$

这里 $\mathbf{g}_k$ 是梯度，$\mathbf{g}_k = \nabla f(\mathbf{x}_k)$，设搜索点为 $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$，$\alpha_k > 0$

$$\Longrightarrow f(\mathbf{x}_{k+1}) = f(\mathbf{x}_k) + \alpha_k \mathbf{g}_k^\top \mathbf{d}_k + o(\|\mathbf{x}_{k+1} - \mathbf{x}_k\|)$$

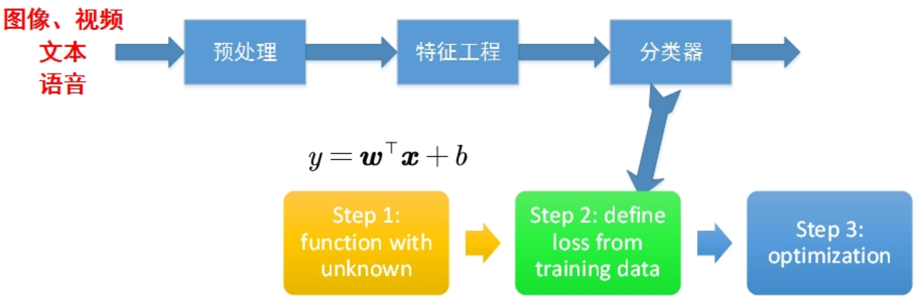
显然当 $\mathbf{d}_k = -\mathbf{g}_k$ 时，$f(\mathbf{x}_{k+1}) \le f(\mathbf{x}_k)$。
当 $k \to \infty$ 时，使得 $f(\mathbf{x}_{k+1}) \to \min f(\mathbf{x})$。则每一步的搜索点为

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k, \qquad \alpha_k > 0$$

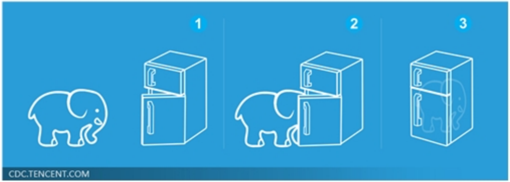
# 梯度下降法

☞ 习题：$\min\ f(\mathbf{x}) = \frac{1}{2}x_1^2 + \frac{9}{2}x_2^2$。初始点 $(9,1)^\top$，$\alpha_k = 0.2$ ，求三个迭代点。

# Machine Learning is so simple ……




$$y = \boldsymbol{w}^\top \boldsymbol{x} + b$$



# Widrow-Hoff 方法

在 20 世纪 60 年代，注意力主要放在如何构造简单的迭代程序来训练线性学习器。Widrow-Hoff 方法（也称为 Adaline 算法）能收敛到最小二乘解

$$\mathbf{g}_k = \frac{\partial L}{\partial \mathbf{w}} = -2\mathbf{X}^{\mathrm{T}}\left(\mathbf{y} - \mathbf{X}\mathbf{w}\right) = \left(\mathbf{x}_1 \quad \cdots \mathbf{x}_n\right)\begin{pmatrix} e_1 \\ \vdots \\ e_n \end{pmatrix} = \sum_{i=1}^{n} e_i \mathbf{x}_i$$

☞ 问题：$e_i$ 是什么？

根据梯度下降法

$$\mathbf{w} \longleftarrow \mathbf{w} - \Delta\mathbf{w}, \quad \Delta\mathbf{w} = \eta \sum_{i=1}^{n} e_i \mathbf{x}_i$$

$\eta > 0$ 为正参数。注意此处的 $\mathbf{w}, \mathbf{x}_i$ 为增广参数向量和增广样本向量。

# Widrow-Hoff 算法

对于线性模型

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0$$

给定训练集 $\{\mathbf{x}_i, y_i\}_{i=1}^{n}$ 和学习率 $\eta > 0$

$\mathbf{w} \leftarrow \mathbf{0};\ w_0 \leftarrow 0$

Repeat

For i=1 to n

$$\begin{pmatrix} \mathbf{w} \\ w_0 \end{pmatrix} \longleftarrow \begin{pmatrix} \mathbf{w} \\ w_0 \end{pmatrix} - \eta\left(\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0 - y_i\right)\begin{pmatrix} \mathbf{x}_i \\ 1 \end{pmatrix}$$

End for

Until 收敛条件满足

返回 $(\mathbf{w}, w_0)$

# 广义线性模型

对于样例 $(\boldsymbol{x}, y)$, $y \in \mathbb{R}$, 若希望线性模型的预测值逼近真实标记，

则得到线性回归模型 $y = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$

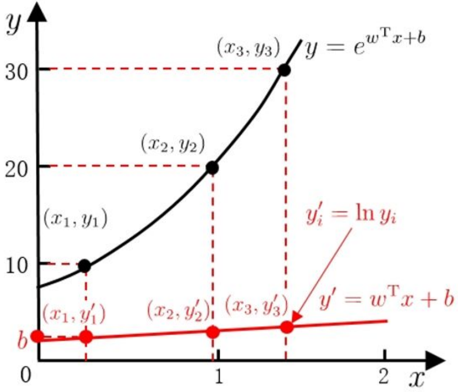
令预测值逼近 y 的衍生物？

若令 $\ln y = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$

则得到对数线性回归

(log-linear regression)

实际是在用 $e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b}$ 逼近 y



# 广义线性模型

In the linear regression models, the model prediction $f(\mathbf{x})$ was given by a linear function of the parameters $\mathbf{w}$. In the simplest case, the model is also linear in the input variables and therefore takes the form

$$f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b$$

so that $f$ is a real number. For classification problems, however, we wish to predict discrete class labels. To achieve this, we consider a generalization of this model in which we transform the linear function of w using a nonlinear function $g(\cdot)$ so that

$$f(\mathbf{x}) = g(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b)$$

# 广义线性模型

In the machine learning literature $g(\cdot)$ is known as an activation function, whereas its inverse is called a link function in the statistics literature. The decision surfaces correspond to $f(\mathbf{x}) = \text{constant}$, so that

$$\mathbf{w}^{\mathrm{T}}\mathbf{x} + b = \text{constant}$$

and hence the decision surfaces are linear functions of $\mathbf{x}$, even if the function $g(\cdot)$ is nonlinear.

# 广义线性模型

一般形式: $y = g^{-1}\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b\right)$

单调可微的 联系函数 (link function)

令 $g(\cdot) = \ln(\cdot)$ 则得到 对数线性回归

$$\ln y = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$$

… …

# 二分类任务

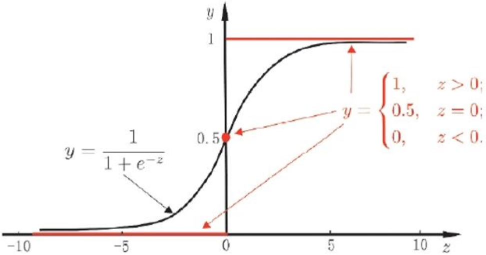
线性回归模型产生的实值输出 $z = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$

期望输出 $y \in \{0, 1\}$

找 $z$ 和 $y$ 的联系函数

理想的“单位阶跃函数”
(unit-step function)

$$y = \begin{cases} 0, & z < 0; \\ 0.5, & z = 0; \\ 1, & z > 0, \end{cases}$$




性质不好，
需找“替代函数”
(surrogate function)

常用
单调可微、任意阶可导

$$y = \frac{1}{1 + e^{-z}}$$

对数几率函数
(logistic function)
简称“对率函数”

# The Curse of Dimensionality

Our geometrical intuitions, formed through a life spent in a space of three dimensions, can fail badly when we consider spaces of higher dimensionality. As a simple example, consider a sphere of radius $r = 1$ in a space of $d$ dimensions, and ask what is the fraction of the volume of the sphere that lies between radius $r = 1 - \epsilon$ and $r = 1$. We can evaluate this fraction by noting that the volume of a sphere of radius r in $d$ dimensions must scale as $r^d$, and so we write

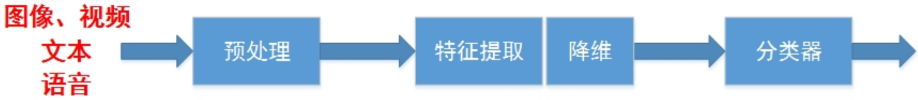
$$V_d(r) = k_d \cdot r^d$$

where the constant $k_d$ depends only on $d$. Thus the required fraction is given by

$$\frac{V_d(1) - V_d(1-\epsilon)}{V_d(1)} = 1 - (1-\epsilon)^d$$

Thus, in spaces of high dimensionality, most of the volume of a sphere is concentrated in a thin shell near the surface!

# 机器学习系统



# 主成份分析（Principal Component Analysis,PCA）算法

计算样本集合（设样本集合中的样本都为 $p$ 维的）

$$\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$$

的协方差矩阵

$$\Sigma = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\top$$

计算协方差矩阵的特征值与特征向量（已归一化、施密特正交化）

$$\lambda_1, \lambda_2, \ldots, \lambda_p$$
$$\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_p$$

不失一般性，可假设 $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p$。取前 $m$ 个特征向量，则给定样本 $\mathbf{x}$（$p$ 维），其降维后的结果（从 $p$ 维降到 $m$ 维）为

$$\mathbf{x} = (\alpha_1, \alpha_2, \ldots, \alpha_p)^\top \Longrightarrow (\beta_1, \beta_2, \ldots, \beta_m)^\top = (\mathbf{x}^\top\boldsymbol{\xi}_1, \mathbf{x}^\top\boldsymbol{\xi}_2, \ldots, \mathbf{x}^\top\boldsymbol{\xi}_m)^\top;$$

# PCA 算法推导过程

给定 $p$ 维欧氏空间（已确定原点和坐标轴）中点

$$\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$$

选取过原点的投影轴 $U$（单位向量为 $\mathbf{u}$）使得所有点到投影轴的距离平方之和最小。

☞思考：这有什么意义？符合那条准则？

☞思考：在基于欧氏距离的表示中（特征选择），如何给出此问题的表达式？

# PCA 算法推导过程

最小平方误差准则（MSE），最小二乘法

$$\min_{\mathbf{u}} \sum_{i=1}^{n} \|\mathbf{x}_i - (\mathbf{x}_i^\top \mathbf{u})\mathbf{u}\|^2$$

由勾股定理

$$\min_{\mathbf{u}} \sum_{i=1}^{n} \|\mathbf{x}_i - (\mathbf{x}_i^\top \mathbf{u})\mathbf{u}\|^2 \Longrightarrow \min_{\mathbf{u}} \sum_{i=1}^{n} \left[\|\mathbf{x}_i\|^2 - (\mathbf{x}_i^\top \mathbf{u})^2\right]$$

$$\min_{\mathbf{u}} \sum_{i=1}^{n} \left[\|\mathbf{x}_i\|^2 - (\mathbf{x}_i^\top \mathbf{u})^2\right] \Longrightarrow \max_{\mathbf{u}} \sum_{i=1}^{n} (\mathbf{x}_i^\top \mathbf{u})^2$$

☞思考：为什么？

$$\max_{\mathbf{u}} \sum_{i=1}^{n} (\mathbf{x}_i^\top \mathbf{u})^2 \Longrightarrow \max_{\mathbf{u}} \mathbf{u}^\top \mathbf{X}^\top \mathbf{X} \mathbf{u}$$

☞思考：将所有点的向量 $\mathbf{x}$ 如何组成矩阵 $\mathbf{X}$？

# PCA 算法推导过程

$$\mathbf{A} = \mathbf{X}^\top \mathbf{X}$$

特征值与特征向量（已归一化、施密特正交化）

$$\lambda_1, \lambda_2, \ldots, \lambda_p$$
$$\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_p$$

不失一般性，可假设 $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p$。
投影轴 $U$（单位向量为 $\mathbf{u}$）可表示为：

$$\mathbf{u} = b_1\boldsymbol{\xi}_1 + b_2\boldsymbol{\xi}_2 + \ldots + b_p\boldsymbol{\xi}_p$$

其中

$$b_1^2 + b_2^2 + \ldots + b_p^2 = 1$$

☞思考：为什么？为什么？

# PCA 算法推导过程

$$\mathbf{A} = \begin{pmatrix} \boldsymbol{\xi}_1 & \cdots & \boldsymbol{\xi}_p \end{pmatrix} \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_p \end{pmatrix} \begin{pmatrix} \boldsymbol{\xi}_1^\top \\ \vdots \\ \boldsymbol{\xi}_p^\top \end{pmatrix}$$

$$\mathbf{u}^\top \mathbf{X}^\top \mathbf{X} \mathbf{u} = \mathbf{u}^\top \mathbf{A} \mathbf{u} = \lambda_1 b_1^2 + \ldots + \lambda_p b_p^2$$

☞ 思考：为什么？

$$\Longrightarrow \max_{\mathbf{u}} \mathbf{u}^\top \mathbf{A} \mathbf{u} \Longrightarrow \boldsymbol{\xi}_1^\top \mathbf{A} \boldsymbol{\xi}_1 = \lambda_1$$

# PCA 算法推导过程

假如向两个投影轴投影，使得点集到投影轴的距离之和最小

$$\min_{\mathbf{u}} \sum_{i=1}^{n} \|\mathbf{x}_i - (\mathbf{x}_i^\top \mathbf{u}_1)\mathbf{u}_1 - (\mathbf{x}_i^\top \mathbf{u})\mathbf{u}\|^2$$

$$\Longrightarrow \min_{\mathbf{u}} \sum_{i=1}^{n} \left[\|\mathbf{x}_i\|^2 - (\mathbf{x}_i^\top \mathbf{u}_1)^2 - (\mathbf{x}_i^\top \mathbf{u})^2\right]$$

$$\Longrightarrow \min_{\mathbf{u}} \sum_{i=1}^{n} \left[\|\mathbf{x}_i\|^2 - \lambda_1 - (\mathbf{x}_i^\top \mathbf{u})^2\right]$$

$$\Longrightarrow \min_{\mathbf{u}} \sum_{i=1}^{n} \left[\|\mathbf{x}_i\|^2 - \lambda_1 - (\mathbf{x}_i^\top \mathbf{u})^2\right]$$

# PCA 算法推导过程

$$
\begin{aligned}
&\Longrightarrow \min_{\mathbf{u}} \sum_{i=1}^{n} \left[ \|\mathbf{x}_i\|^2 - \lambda_1 - (\mathbf{x}_i^\top \mathbf{u})^2 \right] \\
&\Longrightarrow \max_{\mathbf{u}} \sum_{i=1}^{n} (\mathbf{x}_i^\top \mathbf{u})^2 \\
&\Longrightarrow \max_{\mathbf{u} \perp \boldsymbol{\xi}_1} \mathbf{u}^\top \mathbf{X}^\top \mathbf{X} \mathbf{u}^2 \\
&\Longrightarrow \boldsymbol{\xi}_2^\top \mathbf{A} \boldsymbol{\xi}_2^\top = \lambda_2
\end{aligned}
$$

如此下去，得到 $m$ 个最大特征值和相应的特征向量，也就是向这些特征向量投影，得到的距离平方和最小。

# PCA 算法推导过程

☞思考：

计算样本集合（样本集合中的样本都为 p 维的）

$$\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$$

的协方差矩阵

$$\Sigma = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\top$$

与前面的矩阵

$$\mathbf{A} = \mathbf{X}^\top \mathbf{X}$$

的比较。

# 主成份

1. 主轴：称 $\boldsymbol{\xi}_i$ 为 $\mathbf{X}$ 第 $i$ 个主轴向量
2. 主坐标：称

$$\mathbf{x}_i^\top \boldsymbol{\xi}_1, \mathbf{x}_i^\top \boldsymbol{\xi}_2, \ldots, \mathbf{x}_i^\top \boldsymbol{\xi}_p$$

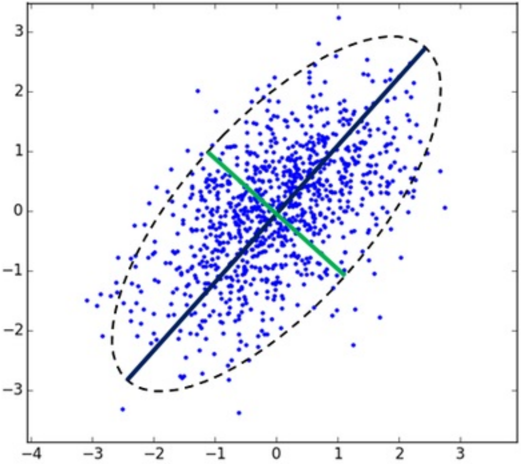
   为 $\mathbf{x}_i$ 主坐标
3. 主成份：$n$ 个样本点的第 $j$ 个主坐标形成的向量

$$\mathbf{y}_{(j)} = \mathbf{X}\boldsymbol{\xi}_j = \begin{pmatrix} \mathbf{x}_1^\top \boldsymbol{\xi}_j \\ \vdots \\ \mathbf{x}_n^\top \boldsymbol{\xi}_j \end{pmatrix}$$

☞思考，主成份分析损失的信息有多少？

一句话概括PCA的话就是找到方差在该方向上投影最大的那些方向，比如下边这个图是用$\begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$作为些协方差矩阵产生的高斯分布样本：

$$\begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix} \Rightarrow$$



# 例子

采用主成分分析方法对下面 4 个输入向量进行分析。

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \qquad \mathbf{x}_2 = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}, \qquad \mathbf{x}_3 = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \qquad \mathbf{x}_4 = \begin{bmatrix} 1 \\ 4 \\ 1 \end{bmatrix}$$

# 第一步检验 $\mathbf{x}$ 是否满足均值为零的条件

计算：

$$\bar{\mathbf{x}} = \frac{1}{4}\begin{bmatrix} 1+2+0+1 \\ 0+3+1+4 \\ 1+1+1+1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

为满足均值为零的条件，将输入向量转换为 $\mathbf{x}' = \mathbf{x} - \bar{\mathbf{x}}$ ，有：

$$\mathbf{x}'_1 = \begin{bmatrix} 0 \\ -2 \\ 0 \end{bmatrix}, \quad \mathbf{x}'_2 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \quad \mathbf{x}'_3 = \begin{bmatrix} -1 \\ -1 \\ 0 \end{bmatrix}, \quad \mathbf{x}'_4 = \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix}$$

# 第二步求 $\mathbf{x}$ 的协方差矩阵和特征值

$$\Sigma = \frac{1}{4}\left[\mathbf{x}'_1\mathbf{x}'^{\top}_1 + \mathbf{x}'_2\mathbf{x}'^{\top}_2 + \mathbf{x}'_3\mathbf{x}'^{\top}_3 + \mathbf{x}'_4\mathbf{x}'^{\top}_4\right]$$
$$= \begin{bmatrix} 0.5 & 0.5 & 0 \\ 0.5 & 2.5 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

求出 $\Sigma$ 的三个特征值从大到小排序： $\lambda_1 = 2.618$ , $\lambda_2 = 0.382$ , $\lambda_3 = 0$ ，对应的特征向量为：

$$\boldsymbol{\xi}_1 = \begin{bmatrix} 0.2298 \\ 0.9732 \\ 0 \end{bmatrix}, \qquad \boldsymbol{\xi}_2 = \begin{bmatrix} -0.9732 \\ 0.2298 \\ 0 \end{bmatrix}, \qquad \boldsymbol{\xi}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

# 第三步降维处理

将输入向量压缩到一维，各输入向量的第一个主成分用式 $y_1 = \boldsymbol{\xi}_1^\top \mathbf{x}'$ 计算，得到：

$$\mathbf{y}_1 = \begin{bmatrix}-1.9465\end{bmatrix},\ \mathbf{y}_2 = \begin{bmatrix}1.2030\end{bmatrix},\ \mathbf{y}_3 = \begin{bmatrix}-1.2030\end{bmatrix},\ \mathbf{y}_4 = \begin{bmatrix}1.9465\end{bmatrix}$$

将输入向量压缩到二维，各输入向量的第一个主成分和第二个主成分，$\begin{matrix} y_1 = \boldsymbol{\xi}_1^\top \mathbf{x}' \\ y_2 = \boldsymbol{\xi}_2^\top \mathbf{x}' \end{matrix}$ 计算，得到：

$$\mathbf{y}_1 = \begin{bmatrix}-1.9465\\-0.4595\end{bmatrix},\ \mathbf{y}_2 = \begin{bmatrix}1.2030\\-0.7435\end{bmatrix},\ \mathbf{y}_3 = \begin{bmatrix}-1.2030\\0.7435\end{bmatrix},\ \mathbf{y}_4 = \begin{bmatrix}1.9465\\0.4595\end{bmatrix}$$

# PCA 与 SVD

我们也经常看到很多的开源代码用 SVD（Singular value decomposition，奇异值分解）实现 PCA，那么 PCA 与 SVD 关系是什么呢？

☞ 奇异值：矩阵 $\mathbf{X}^\top\mathbf{X}$ 的 $n$ 个特征值 $\lambda_i$ 的平方根 $\sigma_i = \sqrt{\lambda_i}$ 称为矩阵 $\mathbf{X}$ 的奇异值。

如果对$X$做奇异值矩阵分解（SVD分解）：

$$X = USV^\top$$

对角阵$S$对角线上的元素是奇异值，$U$和$V$是正交矩阵：$U^\top U = I, V^\top V = I$。把$X$的奇异值分解代入协方差矩阵：

$$C = \frac{1}{n} X^\top X = \frac{1}{n} VS^\top U^\top USV^\top = V\frac{S^2}{n}V^\top$$

$d \times d$正交矩阵$V$的每一列是特征向量，不难发现特征值与奇异值之间存在着对应关系$\lambda_i = S_{ii}^2/n$。对$X$主成分方向进行投影：

$$XV_k = USV^\top V_k = U_k S_k$$

$U_k$包含$U$的前k列，$S_k$包含$S$左上角的$k \times k$个元素。

# numpy.linalg.svd 函数

函数：np.linalg.svd(a,full_matrices=1,compute_uv=1)。

参数：

- a是一个形如(M,N)矩阵
- full_matrices的取值是为0或者1，默认值为1，这时u的大小为(M,M)，v的大小为(N,N)。否则u的大小为(M,K)，v的大小为(K,N)，K=min(M,N)。
- compute_uv的取值是为0或者1，默认值为1，表示计算u,s,v。为0的时候只计算s。

返回值：

- 总共有三个返回值u,s,v
- u大小为(M,M)，s大小为(M,N)，v大小为(N,N)。
- A = u*s*v
- 其中s是对矩阵a的**奇异值分解**。s除了对角元素不为0，其他元素都为0，并且对角元素从大到小排列。s中有n个奇异值，一般排在后面的比较接近0，所以仅保留比较大的r个奇异值。

# numpy.linalg.svd 函数实例

```
>>> from numpy import *
>>> data = mat([[1,2,3],[4,5,6]])
>>> U,sigma,VT = np.linalg.svd(data)
>>> print U
[[-0.3863177  -0.92236578]
 [-0.92236578  0.3863177 ]]
>>> print sigma
[9.508032   0.77286964]
>>> print VT
[[-0.42866713 -0.56630692 -0.7039467 ]
 [ 0.80596391  0.11238241 -0.58119908]
 [ 0.40824829 -0.81649658  0.40824829]]
```

# 流形学习

流形学习，全称流形学习方法 (Manifold Learning)，自 2000 年在著名的科学杂志《Science》被首次提出以来，已成为信息科学领域的研究热点。在理论和应用上，流形学习方法都具有重要的研究意义。假设数据是均匀采样于一个高维欧氏空间中的低维流形，流形学习就是从高维采样数据中恢复低维流形结构，即找到高维空间中的低维流形，并求出相应的嵌入映射，以实现维数约简或者数据可视化。它是从观测到的现象中去寻找事物的本质，找到产生数据的内在规律。

流形学习方法是模式识别中的基本方法，分为线性流形学习算法和非线性流形学习算法，

- 非线性流形学习算法包括等距映射（Isomap），拉普拉斯特征映射（Laplacian eigenmaps，LE），局部线性嵌入 (Locally-linear embedding，LLE) 等。
- 而线性方法常见的有，主成分分析（Principal component analysis，PCA），多维尺度变换（Multidimensional scaling，MDS）等。

# 特征脸 (Eigenface)

其中涉及到一个矩阵的计算问题
样本集合（样本集合中的样本都为 $p$ 维的）

$$\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$$

的协方差矩阵

$$\Sigma = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\top = \mathbf{X}^\top \mathbf{X}$$

（思考：$\mathbf{X}$ 是什么）矩阵是 $p \times p$ 维的，求出特征值与特征向量计算量很大

# 特征脸 (Eigenface)

设

$$\mathbf{X}\mathbf{X}^\top$$

的特征值与特征向量为：

$$\lambda_1, \lambda_2, \ldots, \lambda_n$$

$$\zeta_1, \zeta_2, \ldots, \zeta_n$$

$$\mathbf{X}\mathbf{X}^\top \zeta_i = \lambda_i \zeta_i$$

$$\Longrightarrow \mathbf{X}^\top \mathbf{X}\mathbf{X}^\top \zeta_i = \lambda_i \mathbf{X}^\top \zeta_i$$

不失一般性，可假设 $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$，所以，$\mathbf{X}^\top \zeta_i$ 是的 $\mathbf{X}^\top \mathbf{X}$ 特征向量，$\lambda_i$ 是 $\mathbf{X}^\top \mathbf{X}$ 的特征值。则，$\mathbf{X}^\top \zeta_i$ 是特征脸，记为 $\boldsymbol{\eta}_i$

# 特征脸 (Eigenface)

假设有 $m$ 个特征脸，则可将新的人脸图片 $\mathbf{x}$，降维为：

$$\begin{pmatrix} \mathbf{x}^\top \boldsymbol{\eta}_1 \\ \vdots \\ \mathbf{x}^\top \boldsymbol{\eta}_m \end{pmatrix}$$

因此，解 $\mathbf{X}^\top\mathbf{X}$ 的特征值特征向量，变成 $\mathbf{X}\mathbf{X}^\top$ 的特征值特征向量。

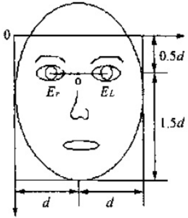
☞ 思考，节约工作量大约有多大？

# 一个简单的人脸识别系统

1. 预处理：人脸图像的分割以及主要器官的定位。图像处理的任务。
2. 特征提取：Eigenface 方法
3. 分类器的设计：K-NN 分类器

# 预处理

需要对人脸图像进行一些列的预处理，以达到位置校准和灰度归一的目的。要进行必要的裁剪和归一化处理。



# 预处理

图 9.7 规一化后的部分人脸图像

# 特征提取

由前面讨论可知 $\mathbf{X}\mathbf{X}^\top$ 的特征值 $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$
所以，$\mathbf{X}^\top\zeta_i$ 是 $\mathbf{X}^\top\mathbf{X}$ 相应于特征值 $\lambda_i$ 的特征向量，也是特征脸，由于这些图像很像人脸，所以称为特征脸。

图 9.8 "特征脸"图像

# 分类器的设计，K 近邻学习器

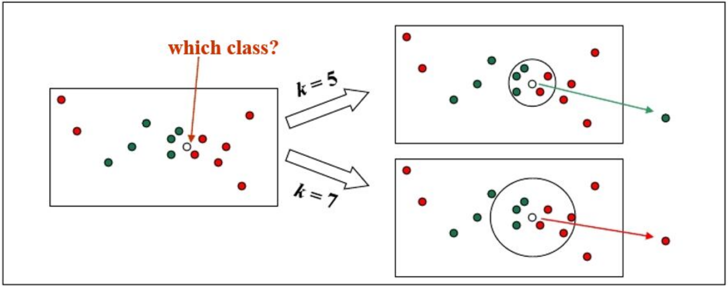
$k$ 近邻 ($k$-Nearest Neighbor, $k$NN)

懒惰学习 (lazy learning) 的代表

基本思路：

近朱者赤，近墨者黑

(投票法；平均法 )




关键：　$k$ 值选取；距离计算

# 分类器的设计

K-NN 算法的基本思路是：在给定新样本 $\mathbf{x}_i$ 后，选择在训练样本集中与该新样本距离最近 (最相似) 的 $K$ 篇样本，根据这 $K$ 篇样本所属的类别判定新样本所属的类别。其中关键步骤为：

- 确定新样本的向量表示。
- 在训练样本集中选出与新样本最相似的 $K$ 个样本，相似度计算公式：

$$\mathrm{sim}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sum_{k=1}^{p} x_{ki} \times x_{kj}}{\sqrt{\sum_{k=1}^{p} x_{ki}^2 \times \sum_{k=1}^{p} x_{kj}^2}} = \frac{\mathbf{x}_i^\top \mathbf{x}_j}{\|\mathbf{x}_i\| \, \|\mathbf{x}_j\|} = \frac{\langle \mathbf{x}_i, \mathbf{x}_j \rangle}{\|\mathbf{x}_i\| \, \|\mathbf{x}_j\|}$$

# 分类器的设计

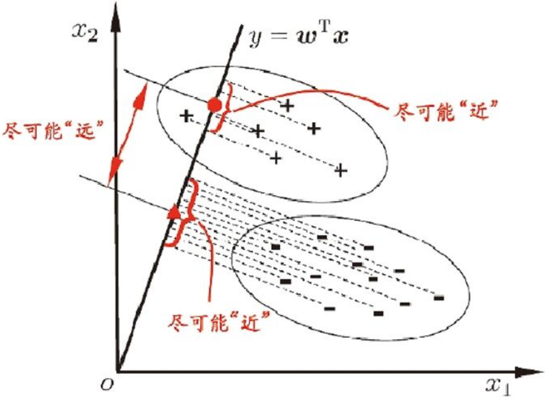
- 在新样本 $\mathbf{x}_i$ 的 $K$ 个邻居中，计算每类的类别状态值：

$$\mathrm{CSV}_c(\mathbf{x}_i) = \sum_{\mathbf{x}\in\mathrm{KNN}} \mathrm{sim}(\mathbf{x}_i, \mathbf{x})\mathrm{y}(\mathbf{x}, c_j) \quad (1)$$

$$\mathrm{y}(\mathbf{x}, c_j) = \begin{cases} 1 & \mathbf{x} \in c_j \\ 0 & \mathbf{x} \notin c_j \end{cases} \quad (2)$$

- 比较类的类别状态值，将样本分到类别状态值最大的类别中。

# 线性判别分析 (Linear Discriminant Analysis)




由于将样例投影到一条直线（低维空间），因此也被视为一种“监督降维”技术 降维 ⟶

# LDA 的目标

给定数据集 $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{m}$

第 $i$ 类示例的集合 $X_i$

第 $i$ 类示例的均值向量 $\boldsymbol{\mu}_i$

第 $i$ 类示例的协方差矩阵 $\boldsymbol{\Sigma}_i$

两类样本的中心在直线上的投影：$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\mu}_0$ 和 $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\mu}_1$

两类样本的协方差：$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\Sigma}_0\boldsymbol{w}$ 和 $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\Sigma}_1\boldsymbol{w}$

同类样例的投影点尽可能接近 → $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\Sigma}_0\boldsymbol{w} + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\Sigma}_1\boldsymbol{w}$ 尽可能小

异类样例的投影点尽可能远离 → $\|\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\mu}_0 - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\mu}_1\|_2^2$ 尽可能大

于是，最大化

$$J = \frac{\left\|\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\mu}_0 - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\mu}_1\right\|_2^2}{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\Sigma}_0\boldsymbol{w} + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\Sigma}_1\boldsymbol{w}} = \frac{\boldsymbol{w}^{\mathrm{T}}\left(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1\right)\left(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1\right)^{\mathrm{T}}\boldsymbol{w}}{\boldsymbol{w}^{\mathrm{T}}\left(\boldsymbol{\Sigma}_0 + \boldsymbol{\Sigma}_1\right)\boldsymbol{w}}$$

# LDA 的目标

类内散度矩阵 (within-class scatter matrix)

$$
\begin{aligned}
\mathbf{S}_w &= \mathbf{\Sigma}_0 + \mathbf{\Sigma}_1 \\
&= \sum_{\boldsymbol{x} \in X_0} (\boldsymbol{x} - \boldsymbol{\mu}_0)(\boldsymbol{x} - \boldsymbol{\mu}_0)^{\mathrm{T}} + \sum_{\boldsymbol{x} \in X_1} (\boldsymbol{x} - \boldsymbol{\mu}_1)(\boldsymbol{x} - \boldsymbol{\mu}_1)^{\mathrm{T}}
\end{aligned}
$$

类间散度矩阵 (between-class scatter matrix)

$$
\mathbf{S}_b = (\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)^{\mathrm{T}}
$$

LDA的目标：最大化广义瑞利商 (generalized Rayleigh quotient)

$$
\boxed{J = \frac{\boldsymbol{w}^{\mathrm{T}} \mathbf{S}_b \boldsymbol{w}}{\boldsymbol{w}^{\mathrm{T}} \mathbf{S}_w \boldsymbol{w}}}
$$

$\boldsymbol{w}$ 成倍缩放不影响 $J$ 值
仅考虑方向

# 求解思路

令 $\boldsymbol{w}^{\mathrm{T}}\mathbf{S}_w\boldsymbol{w}=1$，最大化广义瑞利商等价形式为

$$\min_{\boldsymbol{w}} \; -\boldsymbol{w}^{\mathrm{T}}\mathbf{S}_b\boldsymbol{w}$$
$$\text{s.t. } \boldsymbol{w}^{\mathrm{T}}\mathbf{S}_w\boldsymbol{w}=1$$

运用拉格朗日乘子法，有 $\mathbf{S}_b\boldsymbol{w}=\lambda\mathbf{S}_w\boldsymbol{w}$

$\mathbf{S}_b\boldsymbol{w}$ 的方向恒为 $\boldsymbol{\mu}_0-\boldsymbol{\mu}_1$，不妨令 $\mathbf{S}_b\boldsymbol{w}=\lambda(\boldsymbol{\mu}_0-\boldsymbol{\mu}_1)$

于是 $\quad \boldsymbol{w}=\mathbf{S}_w^{-1}(\boldsymbol{\mu}_0-\boldsymbol{\mu}_1)$

实践中通常是进行奇异值分解 $\mathbf{S}_w=\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathrm{T}}$ $\longrightarrow$ 附录A

然后 $\mathbf{S}_w^{-1}=\mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^{\mathrm{T}}$