

机器学习

Machine Learning

北京航空航天大学计算机学院智能识别与图像处理实验室
IRIP Lab, School of Computer Science and Engineering, Beihang University

黄 迪 刘庆杰

2020年秋季学期
Fall 2020

课前回顾

什么是集成学习?

- 通过将多个学习器进行整合，常可获得比单一学习器显著优越的泛化性能，这对弱分类器尤为明显。
 - 弱分类器：准确率仅比随机猜测略高的分类器
 - 强分类器：识别准确率高并能在多项式时间内完成的分类器

弱分类器 $\xrightarrow{\text{组合}}$ 强分类器

强分类器 $\xrightarrow{\text{组合}}$ 更强的分类器

个体与集成

● 简单分析

关键假设：基学习器的误差相互独立.

现实任务中，个体学习器是为解决同一个问题训练出来的，显然不可能互相独立！

个体学习器的“**准确性**”和“**多样性**”存在冲突.

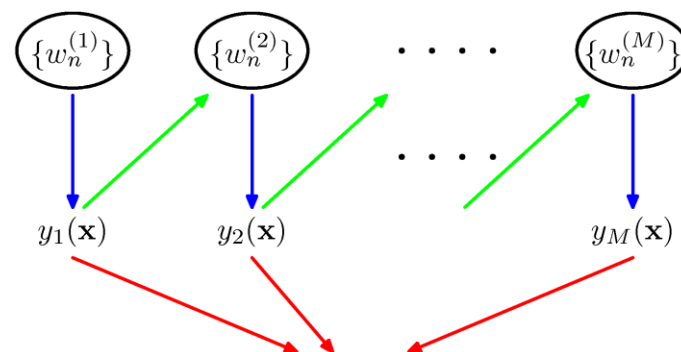
集成学习的研究核心：如何产生“好而不同”的个体学习器.

集成学习方法

- 根据个体学习器生成方式不同，形成两大类方法
 - 串行化方法：个体学习器间存在强依赖关系。
例如：Boosting
 - 并行化方法：个体学习器间不存在强依赖关系。
例如：Bagging、随机森林(Random Forest)

● 一族可将弱学习器提升为强学习器的算法

- 个体学习器存在强依赖关系
- 串行生成
- 每次调整训练数据的样本分布



基本思想：

先从初始数据集训练出一个基学习器，再根据其对训练样本分布进行调整，使**先前做错的样本在后续受到更多关注**，然后基于调整后的样本分布训练下一个基学习器；重复进行直至基学习器数目达到预先指定值。最终将这些基学习器加权结合。

$$Y_M(\mathbf{x}) = \text{sign} \left(\sum_m^M \alpha_m y_m(\mathbf{x}) \right)$$

AdaBoost算法

● Boosting族算法最著名的代表 【1997年Freund和Schapire提出】

f : 真实函数
 $y \in \{-1, +1\}$

输入: 训练集 $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$;
基学习算法 \mathcal{L} ;
训练轮数 T .

过程:

1: $\mathcal{D}_1(\mathbf{x}) = 1/m$.

2: **for** $t = 1, 2, \dots, T$ **do**

3: $h_t = \mathcal{L}(D, \mathcal{D}_t)$;

4: $\epsilon_t = P_{\mathbf{x} \sim \mathcal{D}_t}(h_t(\mathbf{x}) \neq f(\mathbf{x}))$;

5: **if** $\epsilon_t > 0.5$ **then break**

6: $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$;

7:
$$\begin{aligned} \mathcal{D}_{t+1}(\mathbf{x}) &= \frac{\mathcal{D}_t(\mathbf{x})}{Z_t} \times \begin{cases} \exp(-\alpha_t), & \text{if } h_t(\mathbf{x}) = f(\mathbf{x}) \\ \exp(\alpha_t), & \text{if } h_t(\mathbf{x}) \neq f(\mathbf{x}) \end{cases} \\ &= \frac{\mathcal{D}_t(\mathbf{x}) \exp(-\alpha_t f(\mathbf{x}) h_t(\mathbf{x}))}{Z_t} \end{aligned}$$

8: **end for**

输出: $H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \alpha_t h_t(\mathbf{x}) \right)$

初始化样本权值分布

基于分布 \mathcal{D}_t 训练分类器 h_t
估计 h_t 误差

确定 h_t 权重

更新样本分布
(Z_t 规范化因子)

并行化方法 Bagging算法

- 并行式集成学习最著名的代表性方法【1996年Breiman提出】
 - 名字由**B**ootstrap **AGG**regat**ING**缩写而来
 - 基于自助法采样 (Bootstrap Sampling)

给定包含 m 个样本的数据集 D ，对其进行采样产生数据集 D' ：

每次随机从 D 中挑选一个样本，将其拷贝至 D' ，这个过程重复执行 m 次后，就得到了包含 m 个样本的数据集 D' 。显然， D 中有一部分样本会在 D' 中多次出现，而有一部分样本则不会出现。

样本在 m 次采样中始终不被采到的概率是 $(1 - \frac{1}{m})^m$ ，其极限：

将 D' 用做训练集； $D \setminus D'$ 用作测试集

$$\lim_{m \rightarrow \infty} (1 - \frac{1}{m})^m \mapsto \frac{1}{e} \approx 0.368$$

产生不同训练集对集成学习有好处，但改变数据分布会引入偏差。

【1993年Efron和Tibshirani提出】

Bagging算法

- 并行式集成学习最著名的代表性方法【1996年Breiman提出】

基本思想:

利用自助法采样可构造 T 个含 m 个训练样本的采样集，基于每个采样集训练出一个基学习器，再将它们进行结合(在对预测输出结合时，通常对分类任务使用简单投票法，对回归任务使用简单平均法)。

输入: 训练集 $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$;
基学习算法 \mathcal{L} ;
训练轮数 T .

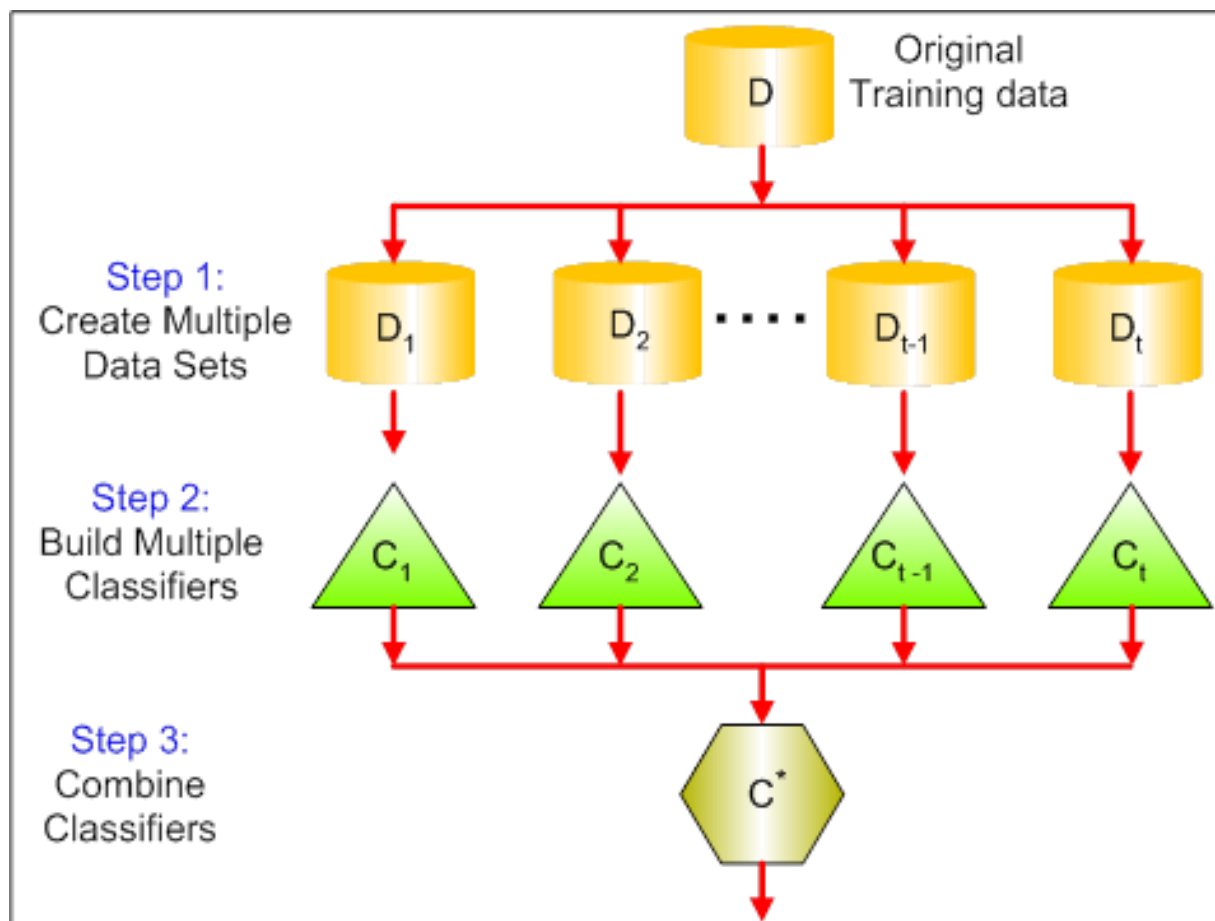
过程:

```
1: for  $t = 1, 2, \dots, T$  do  
2:    $h_t = \mathcal{L}(D, \mathcal{D}_{bs})$   
3: end for
```

输出: $H(\mathbf{x}) = \arg \max_{y \in \mathcal{Y}} \sum_{t=1}^T \mathbb{I}(h_t(\mathbf{x}) = y)$

Bagging算法

- 并行式集成学习最著名的代表性方法【1996年Breiman提出】



并行化方法 随机森林算法

- 是Bagging方法的一种扩展变体 【2001年Breiman提出】
 - Random Forest, 简称RF
 - 以决策树为基学习器
 - 训练过程引入随机属性选择

基本思想：

对基决策树的每个结点，先从该结点的(d 个)属性集合中随机选择一个包含 k 个属性的子集，再从这个子集选择一个最优属性用于划分，一般情况下推荐 $k=\log_2 d$ 。

随机森林算法

- 是Bagging方法的一种扩展变体 【2001年Breiman提出】

Input: Data set $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$;
Feature subset size K .

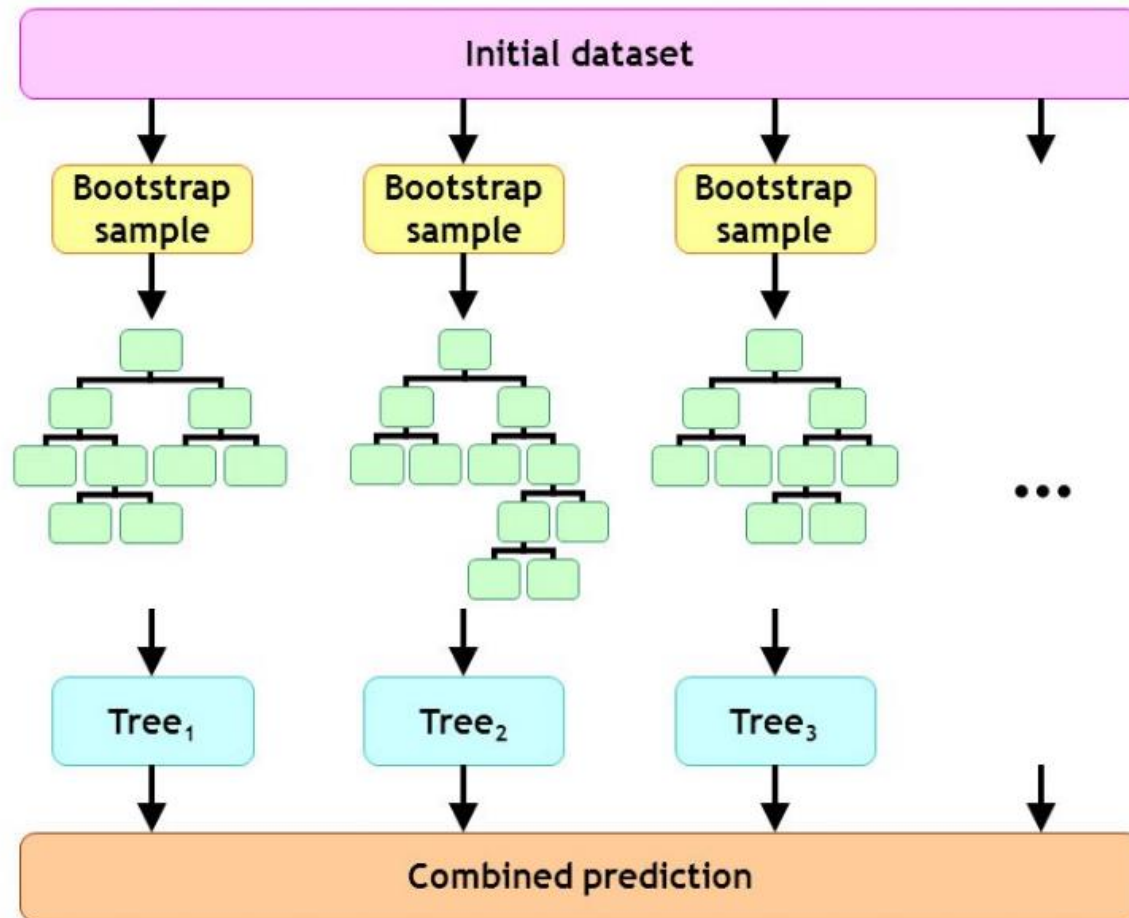
Process:

1. $N \leftarrow$ create a tree node based on D ;
2. **if** *all instances in the same class* **then return** N
3. $\mathcal{F} \leftarrow$ the set of features that can be split further;
4. **if** \mathcal{F} *is empty* **then return** N
5. $\tilde{\mathcal{F}} \leftarrow$ select K features from \mathcal{F} randomly;
6. $N.f \leftarrow$ the feature which has the best split point in $\tilde{\mathcal{F}}$;
7. $N.p \leftarrow$ the best split point on $N.f$;
8. $D_l \leftarrow$ subset of D with values on $N.f$ smaller than $N.p$;
9. $D_r \leftarrow$ subset of D with values on $N.f$ no smaller than $N.p$;
10. $N_l \leftarrow$ call the process with parameters (D_l, K) ;
11. $N_r \leftarrow$ call the process with parameters (D_r, K) ;
12. **return** N

Output: A random decision tree

随机森林算法

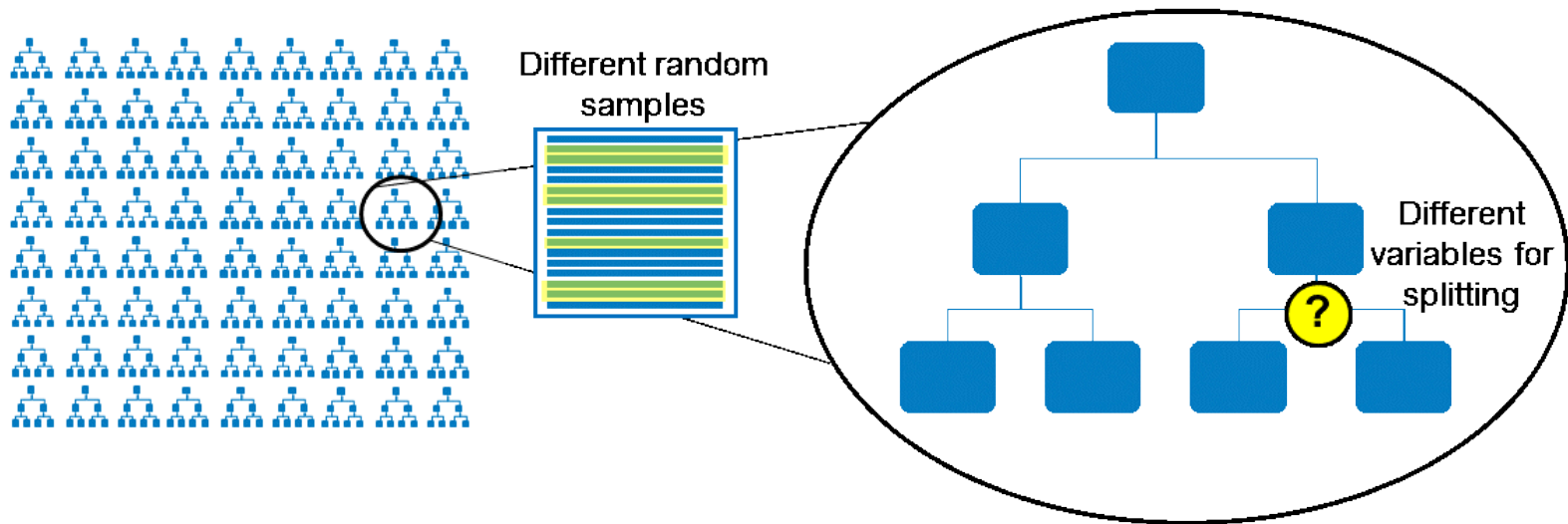
- 是Bagging方法的一种扩展变体 【2001年Breiman提出】



随机森林算法

● 算法特点

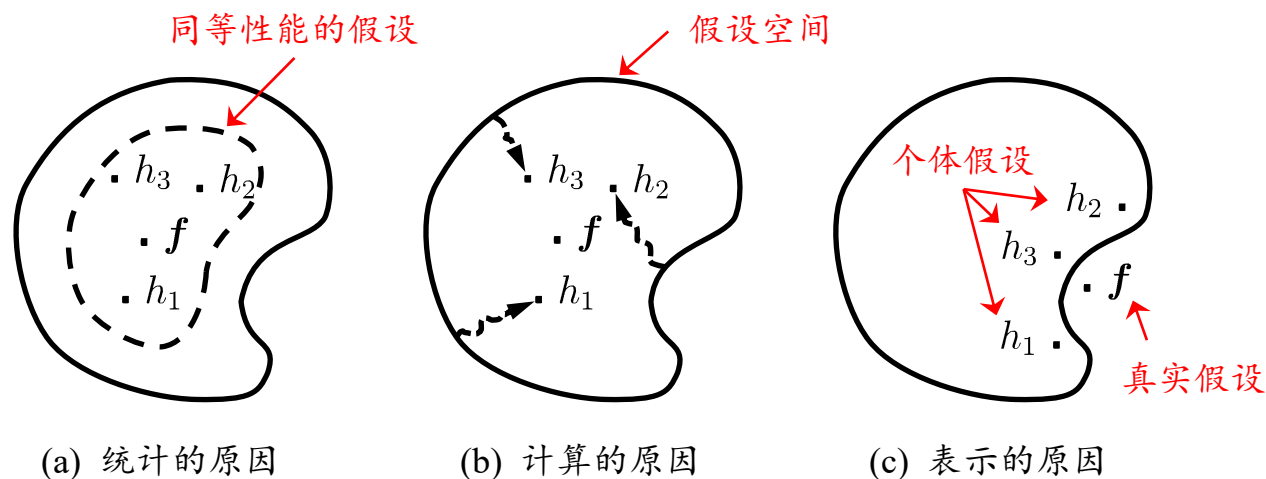
- 基学习器多样性通过**样本扰动**和**属性扰动**实现
- 性能强大，被誉为“**代表集成学习技术水平的方法**”
- 算法简单、容易实现、计算开销小



结合策略

● 学习器的组合有三个方面的好处

- 统计方面：减小误选假设空间导致泛化性能不佳的几率
- 计算方面：降低陷入坏局部极小点影响泛化性能的风险
- 表示方面：扩大假设空间学习对于真实空间更好的近似



结合策略——平均法

- 平均法(Averaging)是数值型输出最常见的结合策略

- 简单平均法(Simple Averaging) 个体学习器性能相近时适用

$$H(\mathbf{x}) = \frac{1}{T} \sum_{i=1}^T h_i(\mathbf{x}).$$

- 加权平均法(Weighted Averaging) 个体学习器性能迥异时适用

【1993年Perrone和Cooper正式将其用于集成学习】

$$H(\mathbf{x}) = \sum_{i=1}^T w_i h_i(\mathbf{x}), \quad w_i \geq 0 \quad \text{and} \quad \sum_{i=1}^T w_i = 1.$$

加权平均法是集成学习的基本出发点，各种结合方法都可视为其特例或变体，不同的集成学习方法是通过不同的方式确定加权平均法中基学习器的权重。

结合策略—投票法

硬投票和软投票

- 投票法(Voting)是标签型输出最常见的结合策略

标记集合 $\{c_1, c_2, \dots, c_N\}$, h_i 在样本 x 上的预测 $\{h_i^1(x), h_i^2(x), \dots, h_i^N(x)\}$

- 绝对多数投票法(Majority Voting): **得票超半数**

$$H(x) = \begin{cases} c_j & \text{if } \sum_{i=1}^T h_i^j(x) > \frac{1}{2} \sum_{k=1}^l \sum_{i=1}^T h_i^k(x) \\ \text{rejection} & \text{otherwise.} \end{cases}$$

- 相对多数投票法(Plurality Voting): **得票最多**

$$H(x) = c_{\arg \max_j \sum_{i=1}^T h_i^j(x)}$$

- 加权投票法(Weighted Voting): **加权后得票最多**

$$H(x) = c_{\arg \max_j \sum_{i=1}^T w_i h_i^j(x)}$$

结合策略—学习法

- 当训练数据很多时采用另一个学习期进行结合

初级学习器 vs. 次级学习器或元学习器(Meta-learner)

- Stacking是学习法的典型代表【1992年Wolpert提出】

Input: Data set $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$;
First-level learning algorithms $\mathcal{L}_1, \dots, \mathcal{L}_T$;
Second-level learning algorithm \mathcal{L} .

Process:

```
1. for  $t = 1, \dots, T$ :    % Train a first-level learner by applying the
2.    $h_t = \mathcal{L}_t(D)$ ;    % first-level learning algorithm  $\mathcal{L}_t$ 
3. end
4.  $D' = \emptyset$ ;          % Generate a new data set
5. for  $i = 1, \dots, m$ :
6.   for  $t = 1, \dots, T$ :
7.     $z_{it} = h_t(\mathbf{x}_i)$ ;
8.   end
9.    $D' = D' \cup ((z_{i1}, \dots, z_{iT}), y_i)$ ;
10. end
11.  $h' = \mathcal{L}(D')$ ;      % Train the second-level learner  $h'$  by applying
                           % the second-level learning algorithm  $\mathcal{L}$  to the
                           % new data set  $D'$ .
```

Output: $H(\mathbf{x}) = h'(h_1(\mathbf{x}), \dots, h_T(\mathbf{x}))$

从初始数据集训练初级学习器

生成次级数据集：初级学习器的输出被当作样例输入特征，继承初始样本标记。

从次级数据集训练次级学习器

数据降维

Dimensionality Reduction

机器学习算法

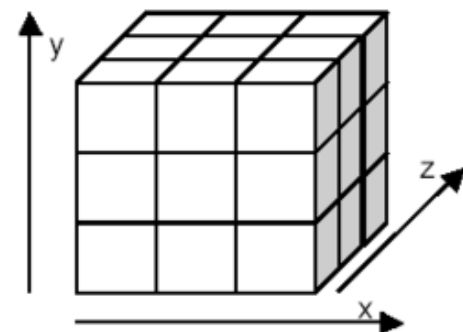
机器学习主要问题

		<i>Supervised Learning</i>	<i>Unsupervised Learning</i>
<i>Discrete</i>		Classification or Categorization	Clustering
	<i>Continuous</i>	Regression	Dimensionality Reduction

数据维数

● 维数(又称维度)

- 数学中：独立参数的数目
- 物理中：独立时空坐标的数目



“点是0维、直线是1维、平面是2维、体是3维”

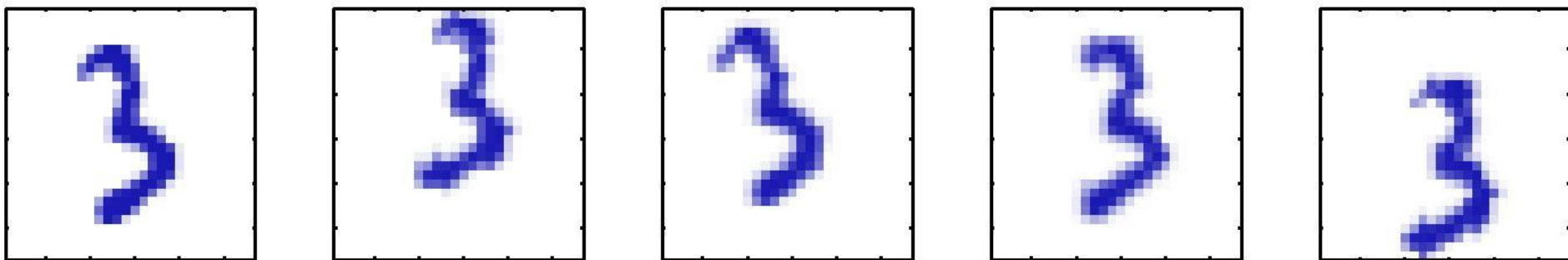


点基于点是0维
点基于直线是1维
点基于平面是2维
点基于体是3维



在点上定位一个点,不需要参数
在直线上定位一个点,需要1个参数
在平面上定位一个点,需要2个参数
在体上定位一个点,需要3个参数

数据维数



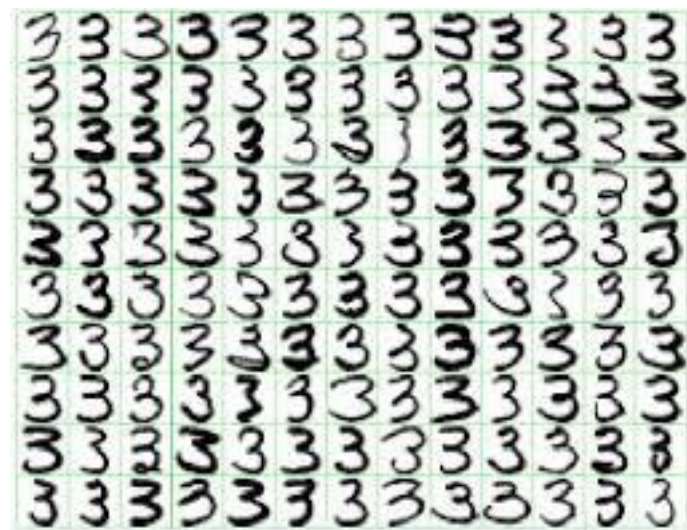
64×64像素数字放入100×100像素白板

● 维数

- 水平/垂直的平移变化
- 旋转变化
- 尺度变化
- 形状变化(不同人的写作习惯)
- 光照变化
- ...

潜变量

(Latent Variables)



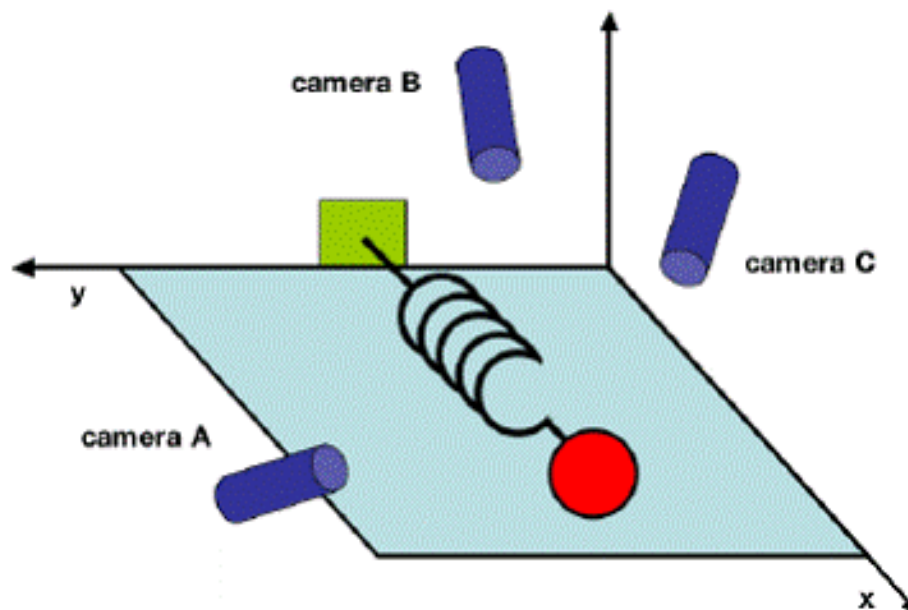
数据降维

● 为什么要降维？

在原始的高维空间中，包含冗余信息和噪声信息，会在实际应用中引入误差，影响准确率；而降维可以提取数据内部的本质结构，减少冗余信息和噪声信息造成的误差，提高应用中的精度。

● 一个简单的例子

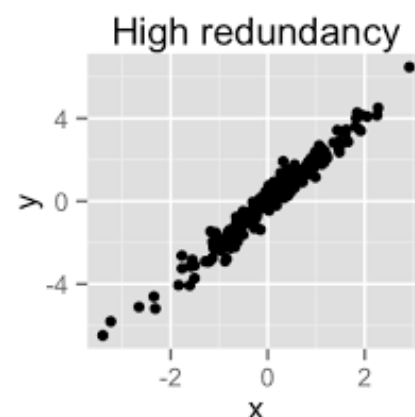
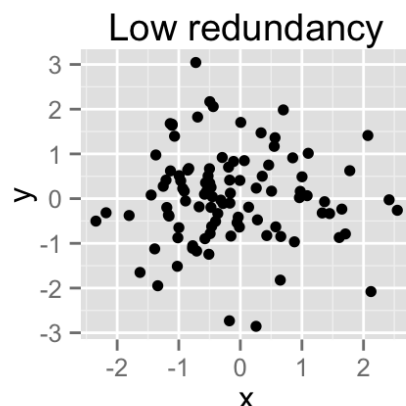
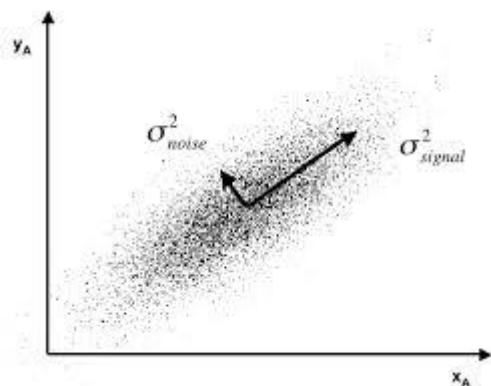
- 沿 x 轴拉小球
- 实际环境



数据降维

● 一个简单的例子(续)

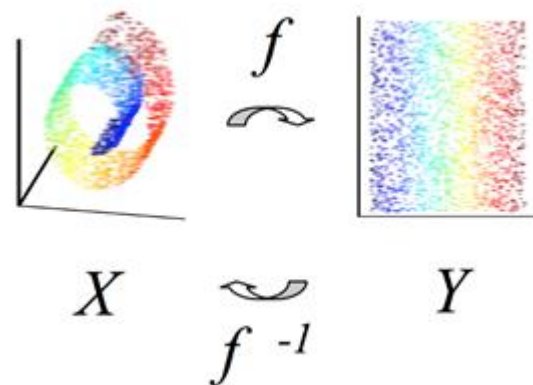
- 噪声
- 冗余



● 降维

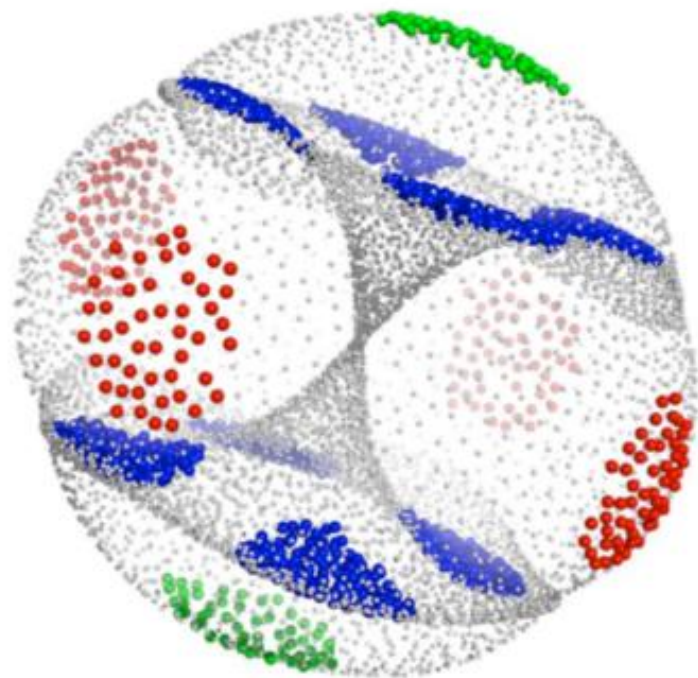
利用某种映射将原高维度空间的数据点投影到低维度的空间:

$$f : X \rightarrow Y$$



降维方法

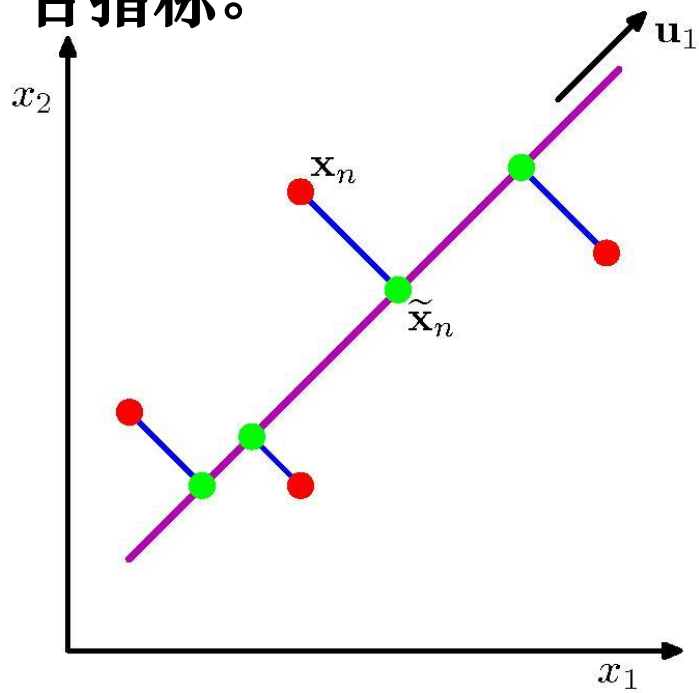
- 主成分分析
Principal Component Analysis
- 等距映射
Isometric Mapping
- 局部线性嵌入
Locally Linear Embedding
- ...



主成分分析-概述

- K. Pearson (1901年论文) 针对非随机变量
- H. Hotelling (1933年论文) 推广到随机向量

主成分分析(Principal Component Analysis, PCA), 将原有众多具有一定相关性的指标重新组合成一组少量互相无关的综合指标。



使得降维后样本的**方差尽可能大**

使得降维后数据的**均方误差尽可能小**

主成分分析-算法

● 最大方差思想

使用较少的数据维度保留住较多的原数据特性

将 **D** 维数据集 $\{\mathbf{x}_n\}, n = 1, 2, \dots, N$ 降为 **M** 维, $M < D$

首先考虑 $M = 1$, 定义这个空间的投影方向为 **D** 维向量 \mathbf{u}_1

出于方便且不失一般性, 令 $\mathbf{u}_1^T \mathbf{u}_1 = 1$

每个数据点 \mathbf{x}_n 在新空间中表示为标量 $\mathbf{u}_1^T \mathbf{x}_n$

样本均值在新空间中表示为 $\mathbf{u}_1^T \bar{\mathbf{x}}$, 其中 $\bar{\mathbf{x}} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n$

投影后样本方差表示为 $\frac{1}{N} \sum_{n=1}^N \{\mathbf{u}_1^T \mathbf{x}_n - \mathbf{u}_1^T \bar{\mathbf{x}}\}^2 = \boxed{\mathbf{u}_1^T \mathbf{S} \mathbf{u}_1}$ 最大

其中原样本方差 $\mathbf{S} = \frac{1}{N} \sum_{n=1}^N (\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T$

主成分分析-算法

- 最大方差思想

使用较少的数据维度保留住较多的原数据特性

目标是**最大化** $\mathbf{u}_1^T \mathbf{S} \mathbf{u}_1$, *s.t.* $\mathbf{u}_1^T \mathbf{u}_1 = 1$

利用拉格朗日乘子法 $\mathbf{u}_1^T \mathbf{S} \mathbf{u}_1 + \lambda_1(1 - \mathbf{u}_1^T \mathbf{u}_1)$

对 \mathbf{u}_1 求导置零得到 $\mathbf{S} \mathbf{u}_1 = \lambda_1 \mathbf{u}_1$

\mathbf{u}_1 是S的特征向量

进一步得到 $\mathbf{u}_1^T \mathbf{S} \mathbf{u}_1 = \lambda_1$

**\mathbf{u}_1 是S最大特征值对应的特征向量时
方差取到极大值，称 \mathbf{u}_1 为第一主成分**

主成分分析-算法

● 最大方差思想

使用较少的数据维度保留住较多的原数据特性

考虑更一般性的情况($M > 1$), 新空间中数据方差最大的最佳投影方向由协方差矩阵 S 的 M 个特征向量 u_1, \dots, u_M 定义, 其分别对应 M 个最大的特征值 $\lambda_1, \dots, \lambda_M$

首先获得方差最大的1维, 生成该维的补空间;

继续在补空间中获得方差最大的1维, 生成新的补空间;

依次循环下去得到 M 维的空间。

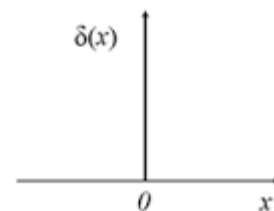
主成分分析-算法

● 最小均方误差思想

使原数据与降维后的数据(在原空间中的重建)的误差最小

定义一组正交的 D 维基向量 $\{\mathbf{u}_i\}, i = 1, \dots, D$, 满足

$$\mathbf{u}_i^T \mathbf{u}_j = \delta_{ij}$$



由于基是完全的, 每个数据点可以表示为基向量的线性组合

$$\mathbf{x}_n = \sum_{i=1}^D \alpha_{ni} \mathbf{u}_i$$

相当于进行了坐标变换

$$\{\mathbf{x}_{n1}, \dots, \mathbf{x}_{nD}\} \xrightarrow{\{\mathbf{u}_i\}} \{\alpha_{n1}, \dots, \alpha_{nD}\}$$



$$\alpha_{nj} = \mathbf{x}_n^T \mathbf{u}_j$$

主成分分析-算法

● 最小均方误差思想

使原数据与降维后的数据(在原空间中的重建)的误差最小

那么 $\mathbf{x}_n = \sum_{i=1}^D (\mathbf{x}_n^T \mathbf{u}_i) \mathbf{u}_i$

在 M 维变量 ($M < D$) 生成的空间中对其进行表示

$$\tilde{\mathbf{x}}_n = \sum_{i=1}^M z_{ni} \mathbf{u}_i + \sum_{i=M+1}^D b_i \mathbf{u}_i$$

独特的

共享的

目标最小化失真度 $J = \frac{1}{N} \sum_{n=1}^N \|\mathbf{x}_n - \tilde{\mathbf{x}}_n\|^2$

导数置零得到 $z_{nj} = \mathbf{x}_n^T \mathbf{u}_j, j = 1, \dots, M$

$$b_j = \bar{\mathbf{x}}^T \mathbf{u}_j, j = M + 1, \dots, D$$

主成分分析-算法

● 最小均方误差思想

使原数据与降维后的数据(在原空间中的重建)的误差最小

$$\text{有 } \mathbf{x}_n - \tilde{\mathbf{x}}_n = \sum_{i=M+1}^D \{(\mathbf{x}_n - \bar{\mathbf{x}})^T \mathbf{u}_i\} \mathbf{u}_i$$

$$J = \frac{1}{N} \sum_{n=1}^N \sum_{i=M+1}^D (\mathbf{x}_n^T \mathbf{u}_i - \bar{\mathbf{x}}^T \mathbf{u}_i)^2 = \sum_{i=M+1}^D \mathbf{u}_i^T \mathbf{S} \mathbf{u}_i$$

$$\text{拉格朗日乘子法 } \tilde{J} = \sum_{i=M+1}^D \mathbf{u}_i^T \mathbf{S} \mathbf{u}_i + \sum_{i=M+1}^D \lambda_i (1 - \mathbf{u}_i^T \mathbf{u}_i)$$

求导得到 $\mathbf{S} \mathbf{u}_i = \lambda_i \mathbf{u}_i$

J 最小时取 $D-M$ 个最小的特征值

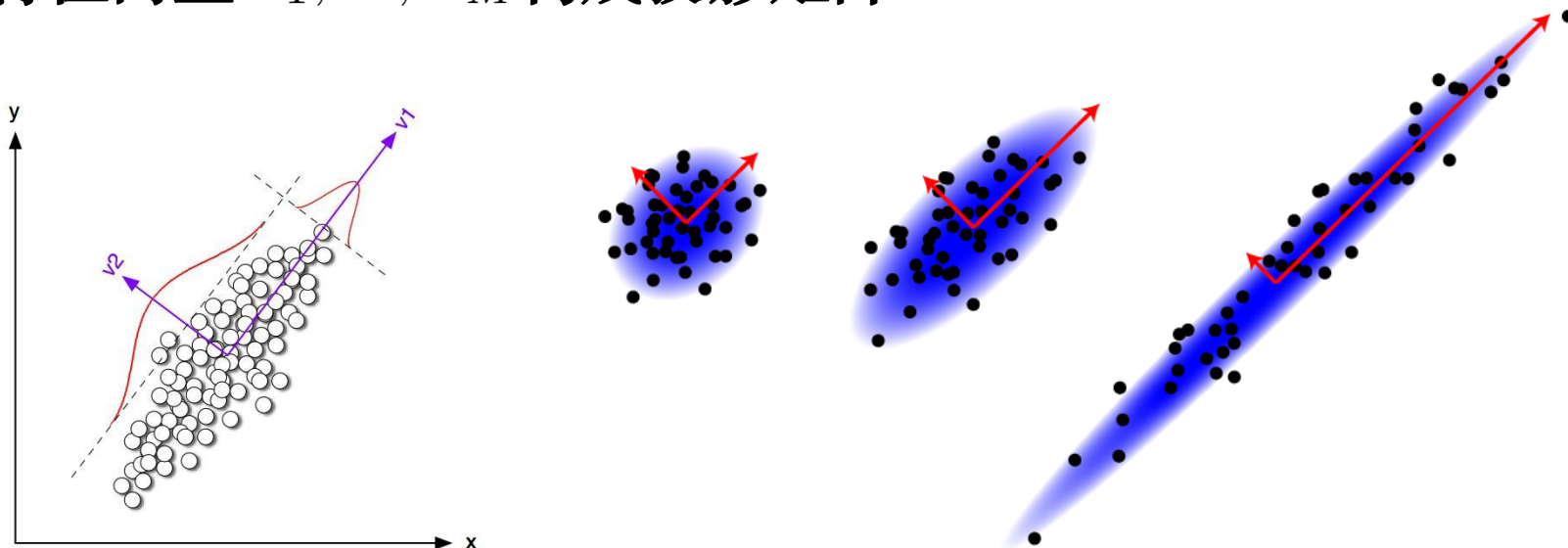
对应失真度为 $J = \sum_{i=M+1}^D \lambda_i$

主子空间对应 M 个最大特征值

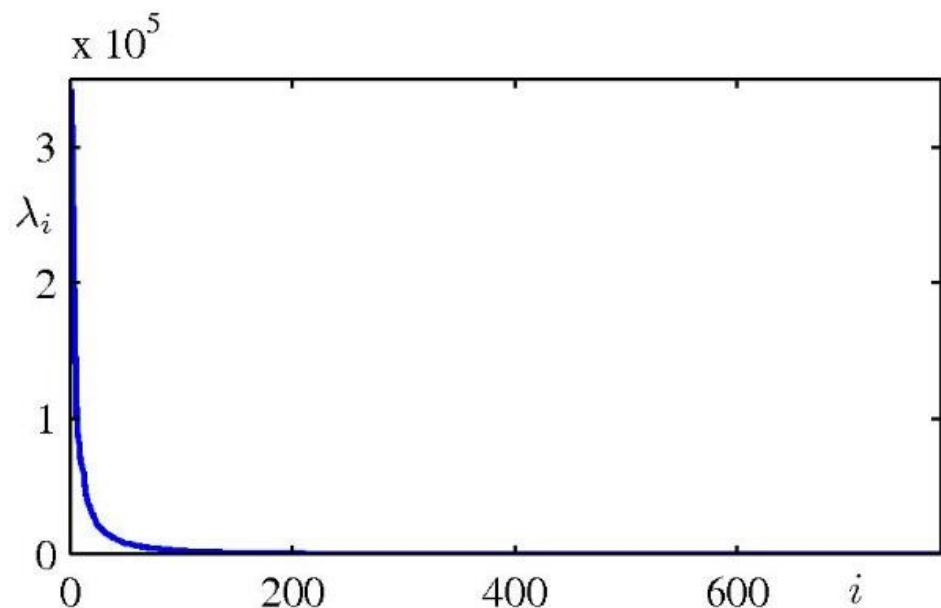
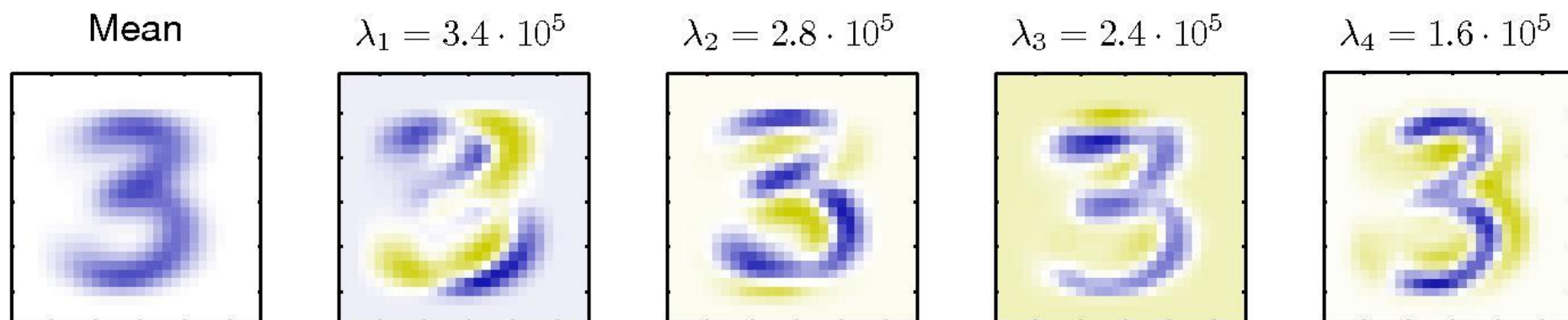
主成分分析-算法

● 计算步骤

- ① 计算给定样本 $\{\mathbf{x}_n\}, n = 1, 2, \dots, N$ 的均值 $\bar{\mathbf{x}}$ 和协方差矩阵 \mathbf{S} ;
- ② 计算 \mathbf{S} 的特征向量与特征值;
- ③ 将特征值从大到小排列, 前 M 个特征值 $\lambda_1, \dots, \lambda_M$ 所对应的特征向量 $\mathbf{u}_1, \dots, \mathbf{u}_M$ 构成投影矩阵。



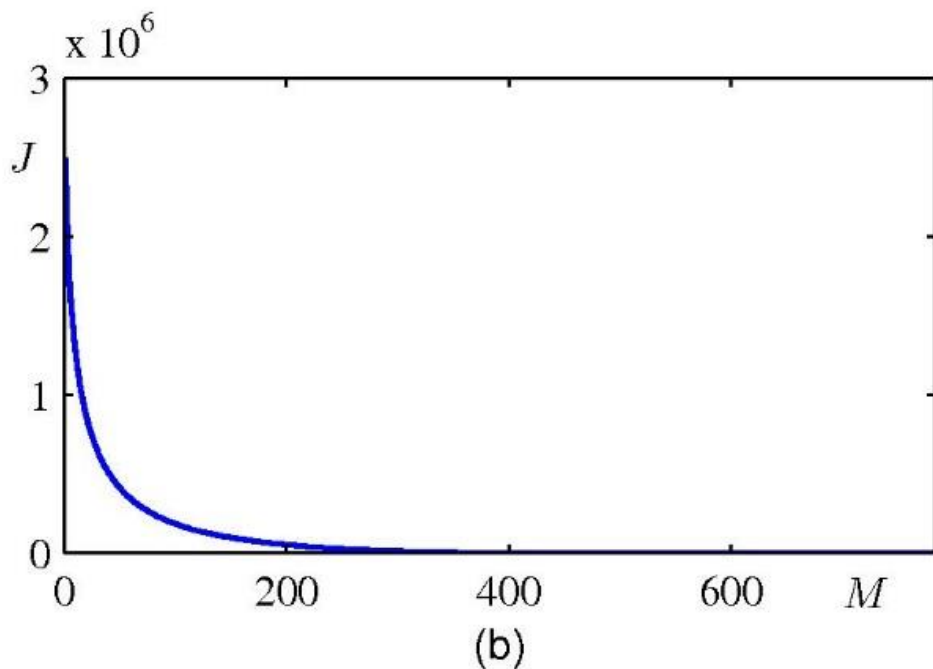
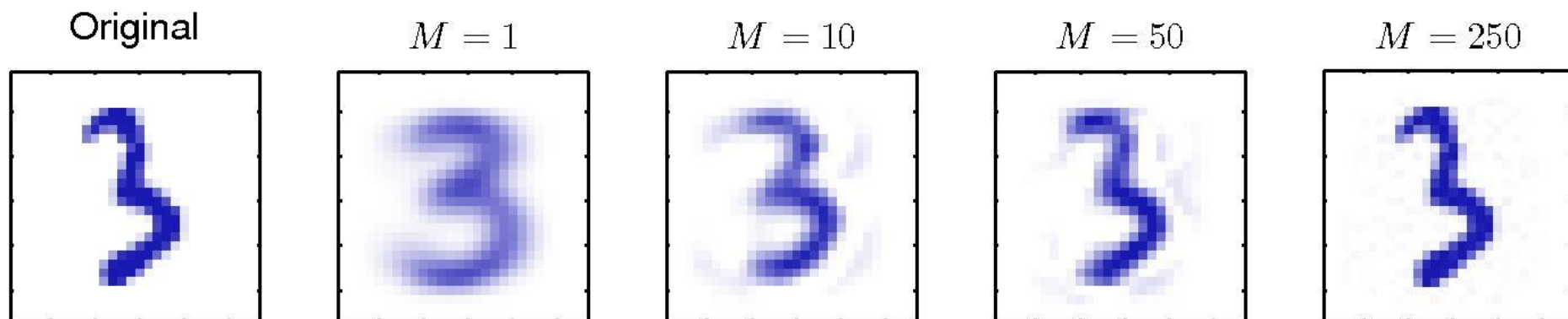
主成分分析-应用



特征值分布谱
特征值由大到小排列

(a)

主成分分析-应用



失真度分布谱
随 M 取值由小到大排列

主成分分析-应用

特征脸(Eigenfaces)#1~#8





主成分分析-应用

特征脸(Eigenfaces)#101~#108



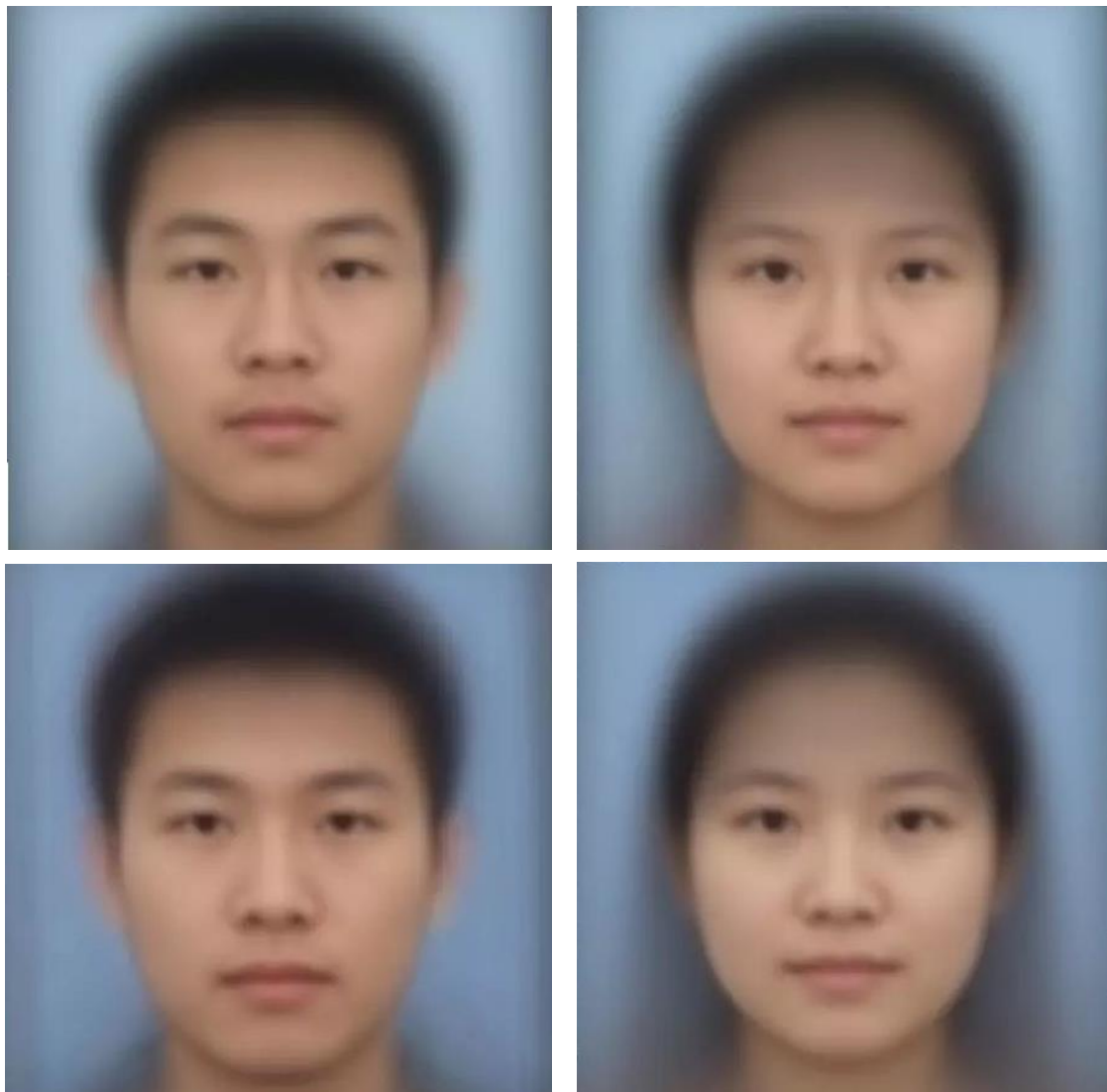


主成分分析-应用

特征脸(Eigenfaces)#501~#508



主成分分析-应用



主成分分析-应用

● 利用PCA处理高维数据

在实际应用中，样本维数可能很高，远大于样本的个数
在人脸识别中，1000张人脸图像，每张图像 100×100 像素

D 维空间， N 个样本点， $N < D$

\mathbf{X} 是 $N \times D$ 维的数据矩阵，其行向量为 $(\mathbf{x}_n - \bar{\mathbf{x}})^T$

$$\mathbf{S} = \frac{1}{N} \sum_{n=1}^N (\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T \text{ 可以写为 } \mathbf{S} = N^{-1} \mathbf{X}^T \mathbf{X}$$

S 维数? $D \times D$ 维 10000×10000

$$\frac{1}{N} \mathbf{X}^T \mathbf{X} \mathbf{u}_i = \lambda_i \mathbf{u}_i \longrightarrow \frac{1}{N} \mathbf{X} \mathbf{X}^T (\mathbf{X} \mathbf{u}_i) = \lambda_i (\mathbf{X} \mathbf{u}_i)$$

令 $\mathbf{v}_i = \mathbf{X} \mathbf{u}_i$ ，得到 $\boxed{\frac{1}{N} \mathbf{X} \mathbf{X}^T} \mathbf{v}_i = \lambda_i \mathbf{v}_i$ $N \times N$ 维

主成分分析-应用

- 利用PCA处理高维数据

在实际应用中，样本维数可能很高，远大于样本的个数
在人脸识别中，1000张人脸图像，每张图像 100×100 像素

对 $\frac{1}{N} \mathbf{X} \mathbf{X}^T$ 求的**特征值** λ_i 和**特征向量** \mathbf{v}_i

$$\frac{1}{N} \mathbf{X} \mathbf{X}^T \mathbf{v}_i = \lambda_i \mathbf{v}_i$$

$$\longrightarrow \frac{1}{N} \mathbf{X}^T \mathbf{X} (\mathbf{X}^T \mathbf{v}_i) = \lambda_i (\mathbf{X}^T \mathbf{v}_i) \quad \text{S的特征向量}$$

调整尺度 $\mathbf{u}_i \propto \mathbf{X}^T \mathbf{v}_i$ 满足 $\|\mathbf{u}_i\| = 1$

$$\longrightarrow \mathbf{u}_i = \frac{1}{(N\lambda_i)^{1/2}} \mathbf{X}^T \mathbf{v}_i$$

奇异值分解(Singular Value Decomposition, SVD)

概率主成分分析

● PCA的概率表示

隐藏变量 z 以如下形式产生 D 维观测变量 x

$$x = Wz + \mu + \epsilon \longrightarrow \text{高斯噪声}$$

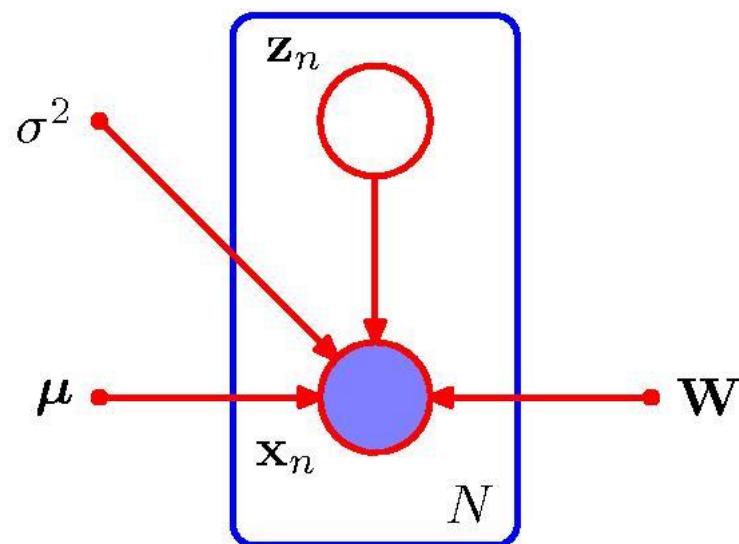
其中, z 为 M 维的隐藏变量, 且满足高斯分布

$$p(z) = \mathcal{N}(z|0, I)$$

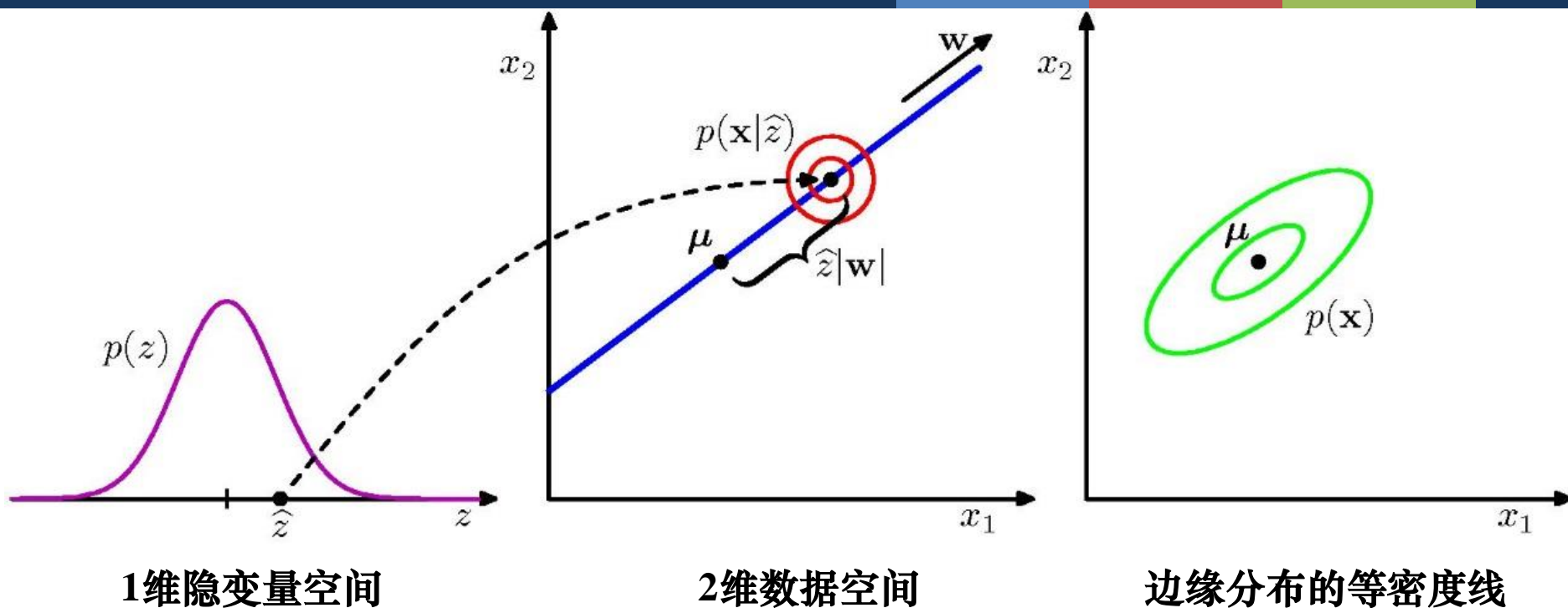
x 以 z 为条件的分布也满足高斯

$$p(x|z) = \mathcal{N}(x|Wz + \mu, \sigma^2 I)$$

以有向图表示



概率主成分分析



从隐空间到数据空间的映射，与PCA的传统视角相反
从数据空间到隐空间的映射，可以由贝叶斯定理得到

概率主成分分析

- PCA的概率表示

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I})$$

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \mu, \sigma^2\mathbf{I})$$



$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z} = \mathcal{N}(\mathbf{x}|\mu, \mathbf{C})$$

$$\mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I}$$

$$\mathbb{E}[\mathbf{x}] = \mathbb{E}[\mathbf{W}\mathbf{z} + \mu + \epsilon] = \mu$$

$$\text{cov}[\mathbf{x}] = \mathbb{E}[(\mathbf{W}\mathbf{z} + \epsilon)(\mathbf{W}\mathbf{z} + \epsilon)^T]$$

$$= \mathbb{E}[\mathbf{W}\mathbf{z}\mathbf{z}^T\mathbf{W}^T] + \mathbb{E}[\epsilon\epsilon^T] = \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I}$$

概率主成分分析

● PCA的概率表示


最大似然估计求解

给定 $\mathbf{X} = \{\mathbf{x}_n\}$ ，求其对数似然函数

$$\begin{aligned}\ln p(\mathbf{X}|\mu, \mathbf{W}, \sigma^2) &= \sum_{n=1}^N \ln p(\mathbf{x}_n|\mu, \mathbf{W}, \sigma^2) \\ &= -\frac{ND}{2} \ln(2\pi) - \frac{N}{2} \ln|\mathbf{C}| - \frac{1}{2} \sum_{n=1}^N (\mathbf{x}_n - \mu)^T \mathbf{C}^{-1} (\mathbf{x}_n - \mu)\end{aligned}$$

对 μ 求导置零并代回 

$$= -\frac{N}{2} \{ D \ln(2\pi) + \ln|\mathbf{C}| + \text{Tr}(\mathbf{C}^{-1} \mathbf{S}) \}$$

 相关矩阵

 $\mathbf{W}_{\text{ML}} = \mathbf{U}_M (\mathbf{L}_M - \sigma^2 \mathbf{I})^{1/2} \mathbf{R}$

$$\sigma_{ML}^2 = \frac{1}{D-M} \sum_{i=M+1}^D \lambda_i$$

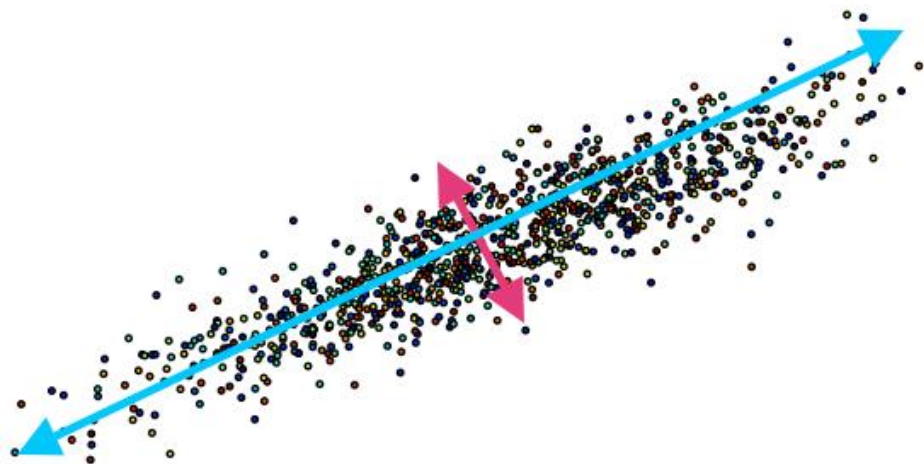
讨论

● PCA的优点

具有很**高普适性**，**最大程度地保持**了原有数据的**信息**；

可**对主元的重要性**进行**排序**，并**根据需要略去部分维数**，达到降维从而简化模型或对数据进行压缩的效果；

完全**无参数限制**，在计算过程中不需要人为设定参数或是根据任何经验模型对计算进行干预，最终结果只与数据相关。



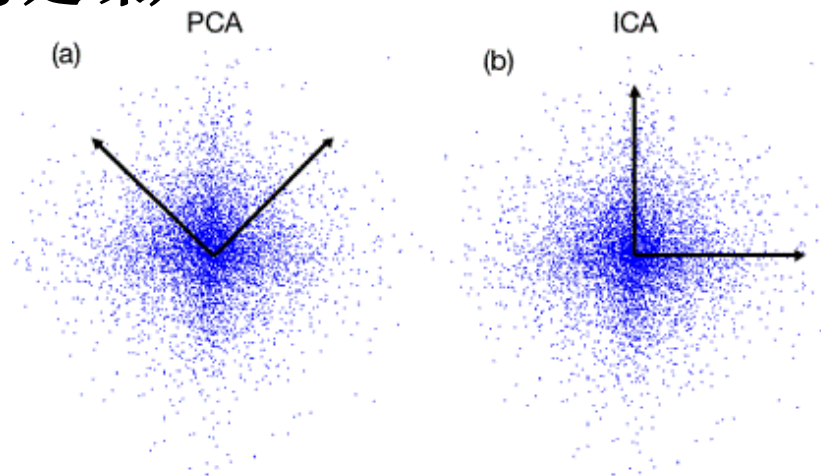
讨论

● PCA的局限性

假设模型是**线性的**，也就决定了它能进行的主元分析之间的关系也是线性的；

假设概率分布模型是**指数型**；

假设数据具有**较高信噪比**，具有最高方差的一维向量被看作是主元，而方差较小的变化被认为是噪声。

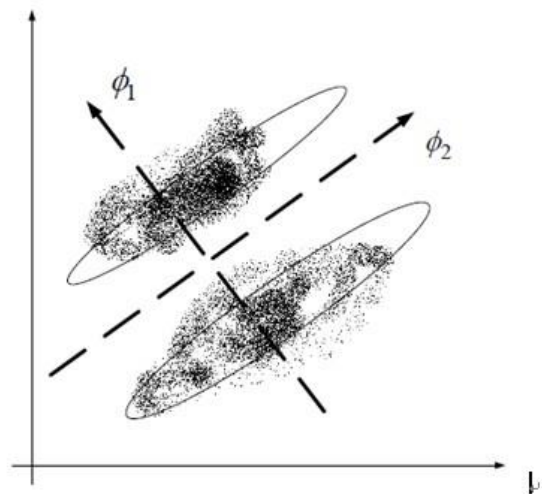


讨论

● PCA vs. LDA

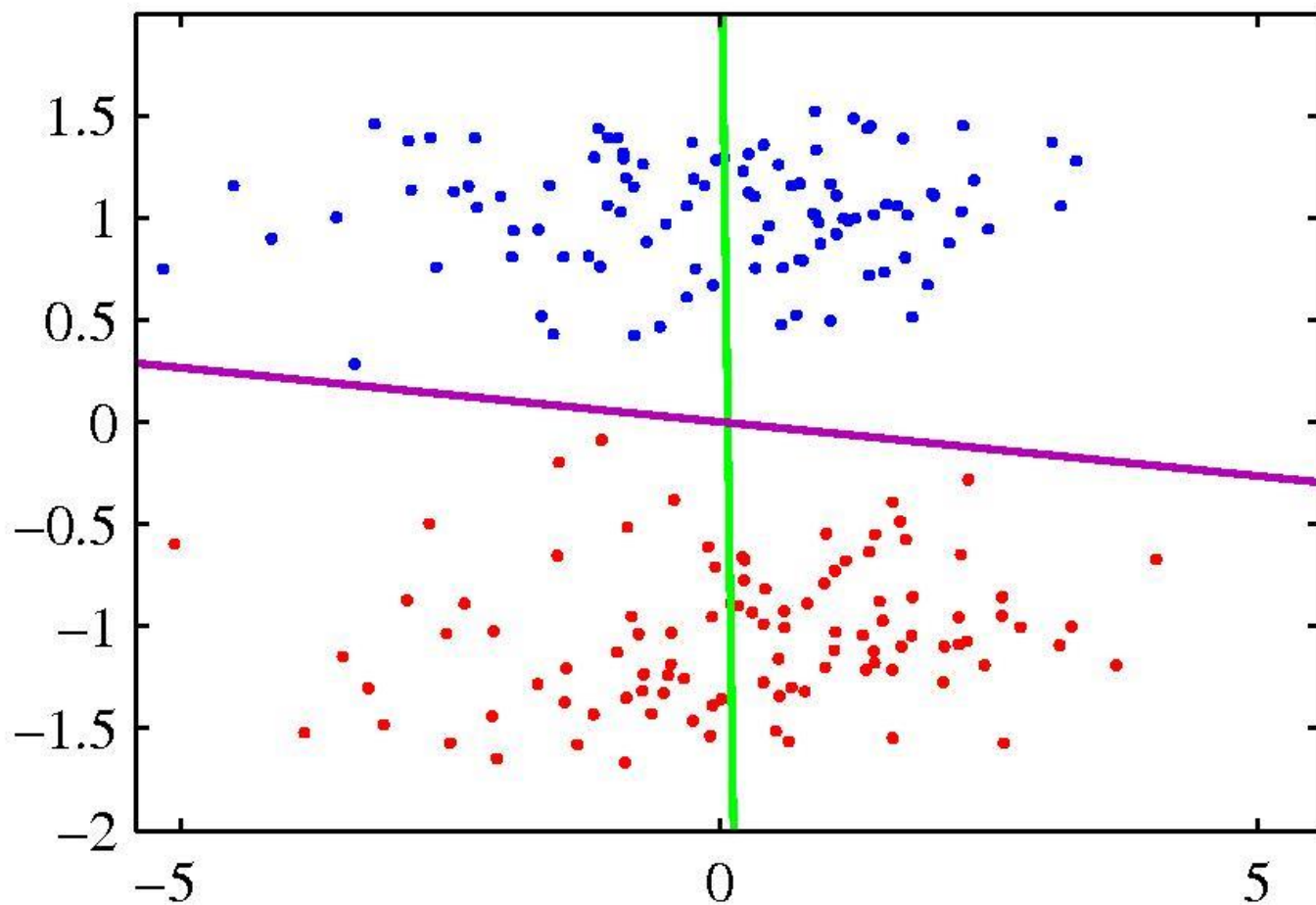
PCA追求降维后能够最大化保持数据内在信息，并通过衡量在投影方向上的数据方差来判断其重要性。但这对数据的**区分作用并不大**，反而可能使得数据点混杂在一起。

LDA所追求的目标与PCA不同，不是希望保持数据最多的信息，而是希望数据在降维后能够很**容易地被区分开**。



讨论

● PCA vs. LDA



核主成分分析

● Kernel PCA

将主成分分析的线性假设一般化使之适应非线性数据

传统PCA: D 维样本 $\{\mathbf{x}_n\}, n = 1, 2, \dots, N, \sum_n \mathbf{x}_n = \mathbf{0}$

$$\mathbf{S}\mathbf{u}_i = \lambda_i \mathbf{u}_i \quad \mathbf{S} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n^T \quad \mathbf{u}_i^T \mathbf{u}_i = 1$$

核PCA: 非线性映射 $\phi(\mathbf{x}), \mathbf{x}_n \mapsto \phi(\mathbf{x}_n), \sum_n \phi(\mathbf{x}_n) = \mathbf{0}$

$$\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i \quad \mathbf{C} = \frac{1}{N} \sum_{n=1}^N \phi(\mathbf{x}_n) \phi(\mathbf{x}_n)^T$$

$$\longrightarrow \frac{1}{N} \sum_{n=1}^N \phi(\mathbf{x}_n) \{\phi(\mathbf{x}_n)^T \mathbf{v}_i\} = \lambda_i \mathbf{v}_i$$

$$\longrightarrow \mathbf{v}_i = \sum_{n=1}^N a_{in} \phi(\mathbf{x}_n)$$

核主成分分析

● Kernel PCA

将主成分分析的线性假设一般化使之适应非线性数据

$$\text{核PCA: } \frac{1}{N} \sum_{n=1}^N \phi(\mathbf{x}_n) \phi(\mathbf{x}_n)^T \sum_{m=1}^N a_{im} \phi(\mathbf{x}_m) = \lambda_i \sum_{n=1}^N a_{in} \phi(\mathbf{x}_n)$$

$$k(\mathbf{x}_n, \mathbf{x}_m) = \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_m)$$

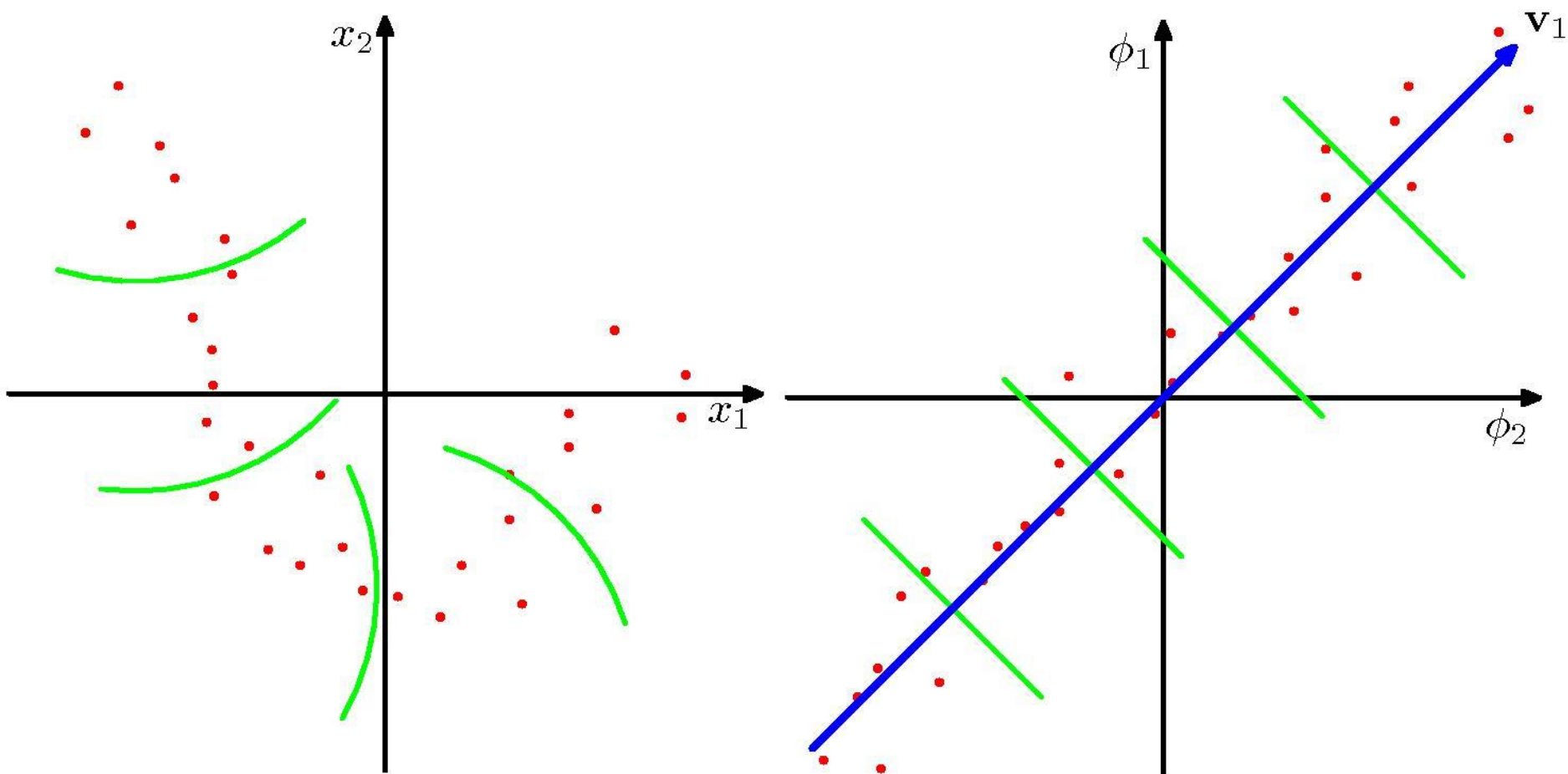
$$\longrightarrow \frac{1}{N} \sum_{n=1}^N k(\mathbf{x}_l, \mathbf{x}_n) \sum_{m=1}^N a_{im} k(\mathbf{x}_n, \mathbf{x}_m) = \lambda_i \sum_{n=1}^N a_{in} k(\mathbf{x}_l, \mathbf{x}_n)$$

$$\longrightarrow \mathbf{K}^2 \mathbf{a}_i = \lambda_i N \mathbf{K} \mathbf{a}_i$$

$$\longrightarrow \mathbf{K} \mathbf{a}_i = \lambda_i N \mathbf{a}_i$$

核主成分分析

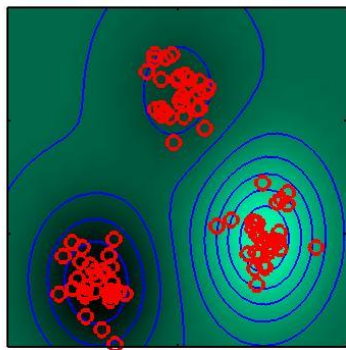
● Kernel PCA



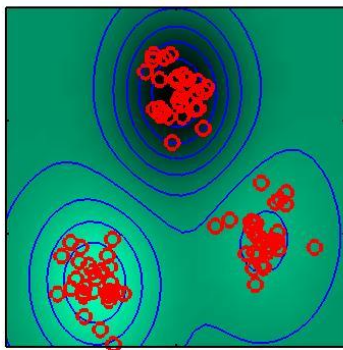
核主成分分析

● Kernel PCA

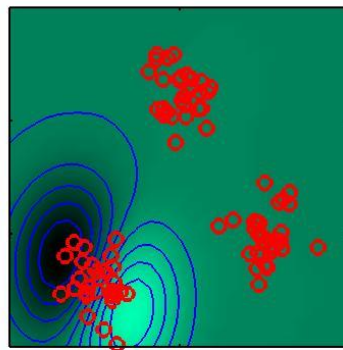
Eigenvalue=21.72



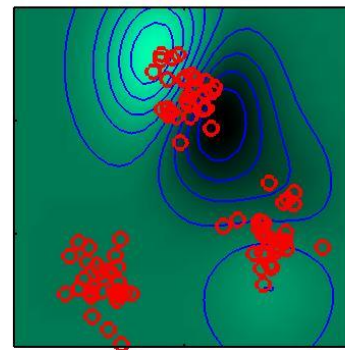
Eigenvalue=21.65



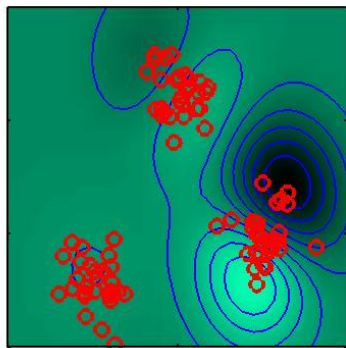
Eigenvalue=4.11



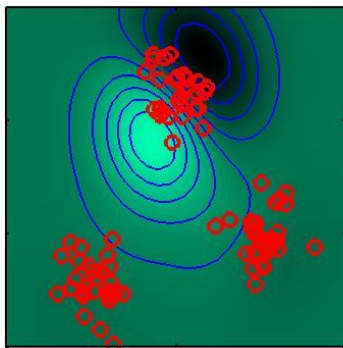
Eigenvalue=3.93



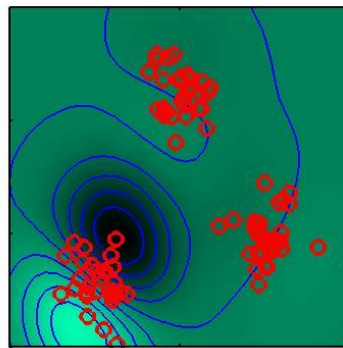
Eigenvalue=3.66



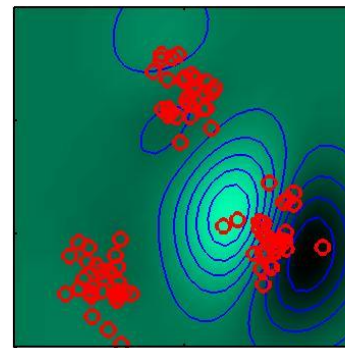
Eigenvalue=3.09



Eigenvalue=2.60



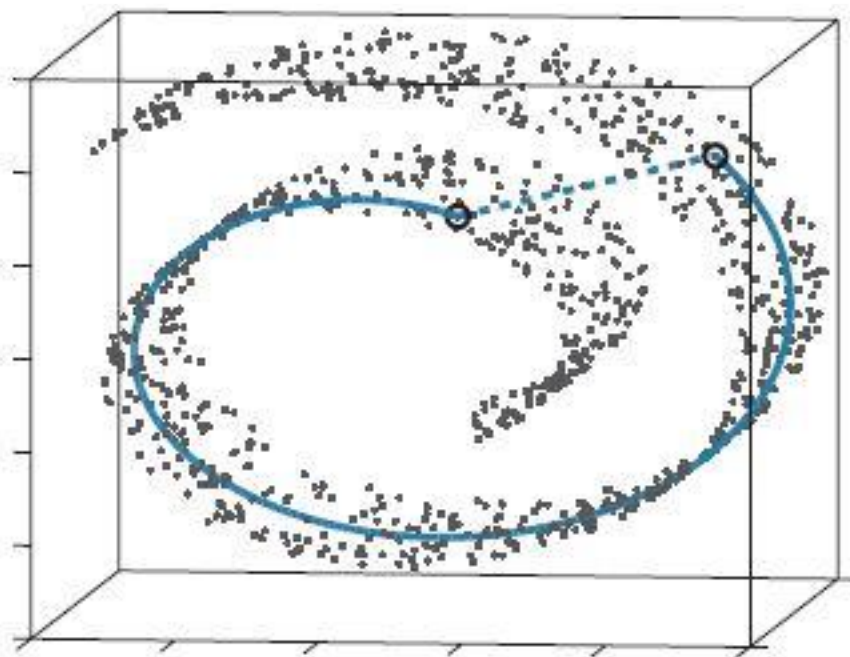
Eigenvalue=2.53



等距映射

● ISO-Metric Mapping

保持数据点内在几何性质(测地距离)



A Global Geometric Framework for Nonlinear Dimensionality Reduction

Joshua B. Tenenbaum,^{1*} Vin de Silva,² John C. Langford³

SCIENCE VOL 290 22 DECEMBER 2000

等距映射

● ISO-Metric Mapping (ISOMAP)

保持数据点内在几何性质(测地距离)

对于给定数据 $X = \{x_1, x_2, \dots, x_N\}$ ，构造图 $G = \{V, E\}$

V 是顶点集合， E 是边的集合

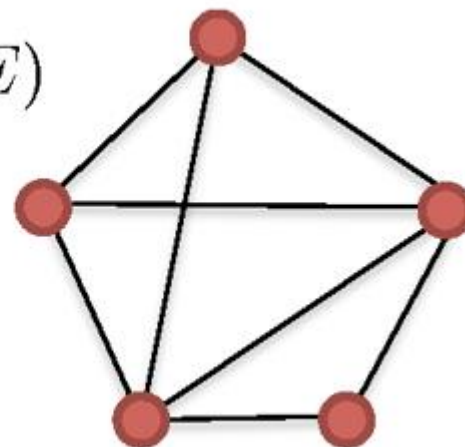
若 $d(i, j) = \text{dist}(x_i, x_j)$

$G = (V, E)$

小于某个值 $\in (\epsilon - \text{ISOMAP})$

或 j 是 i 的 K 近邻($K - \text{ISOMAP}$)

则顶点 i 与 j 的边权值设为 $d(i, j)$ ，否则为0。



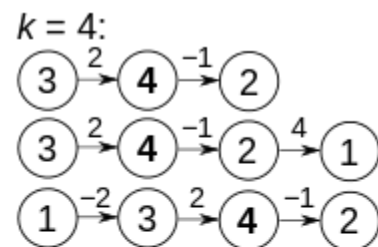
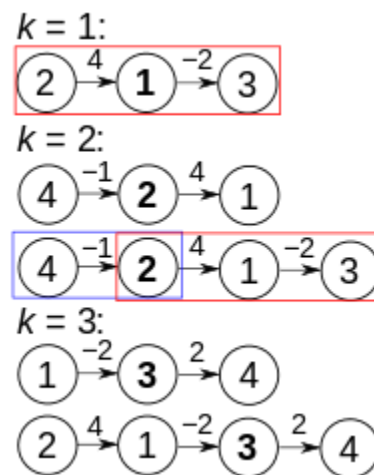
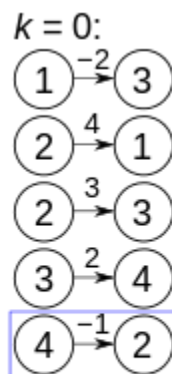
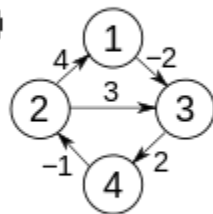
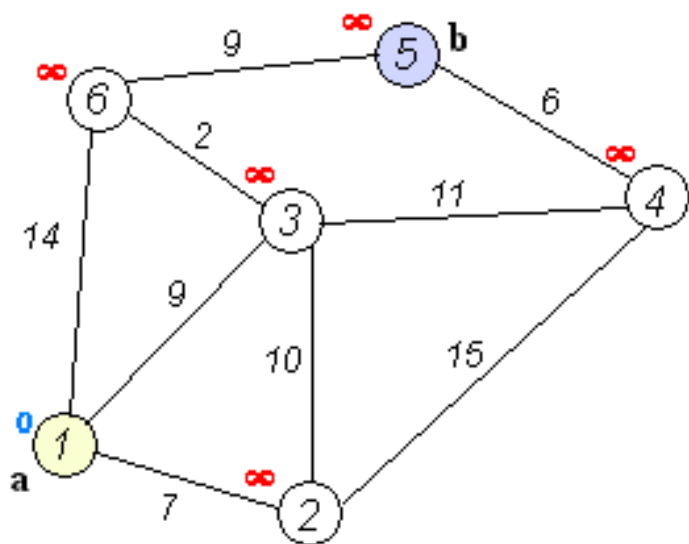
等距映射

● ISO-Metric Mapping (ISOMAP)

保持数据点内在几何性质(测地距离)

计算图 $G = \{V, E\}$ 中任意两点间的最短距离，得到矩阵 $D_G(i, j)$

- **Dijkstra最短路径算法**
- **Floyd–Warshall算法**



等距映射

● ISO-Metric Mapping (ISOMAP)

保持数据点内在几何性质(测地距离)

令 $H = I_n - \frac{1}{N}\mathbf{1}\mathbf{1}^T$ (中心矩阵, Centering Matrix)

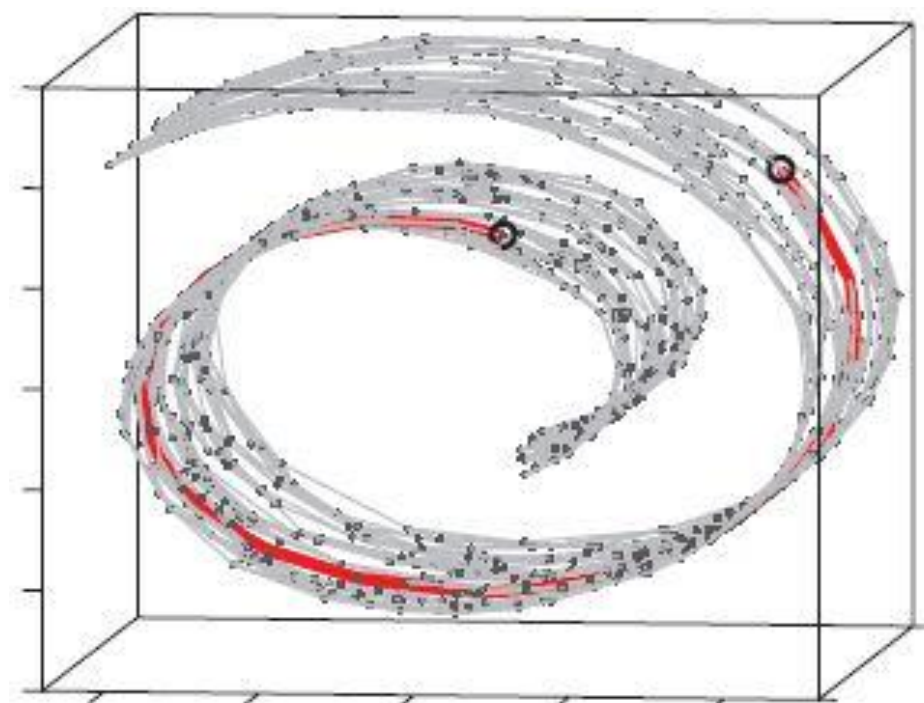
并定义平方距离矩阵 $S(i, j) = D_G^2(i, j)$

求矩阵 $L = -\frac{1}{2}HSH$ 的特征值与特征向量(按特征值降序排列), λ_p 为第 p 个特征值, v_p 为对应的特征向量。

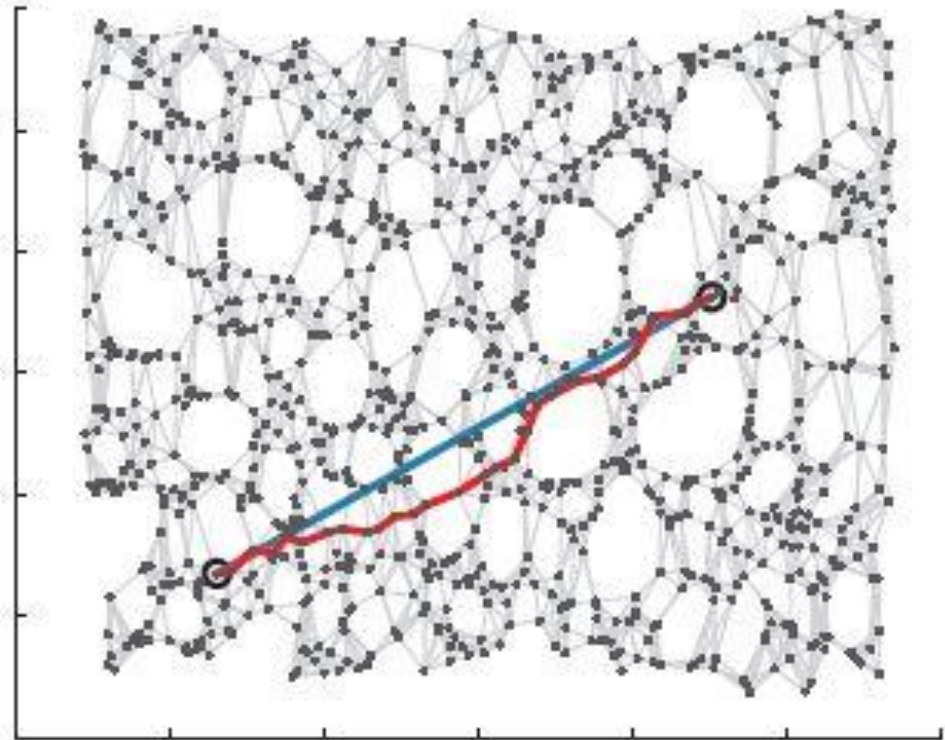
降维矩阵为 $[\sqrt{\lambda_1}v_1, \dots, \sqrt{\lambda_d}v_d]_{N \times d}$

等距映射

● ISO-Metric Mapping (ISOMAP)



$K=7, N=1000$



等距映射

● 计算步骤

①构造临近关系图

对每一个点，将它与指定半径邻域内所有点相连(或与指定个数最近邻相连)

②计算最短路径

计算临近关系图所有点对之间的最短路径，得到距离矩阵

③多尺度分析

将高维空间中的数据点投影到低维空间，使投影前后的距离矩阵相似度最大

等距映射

● ISOMAP优点和缺点

非线性

非迭代

全局最优

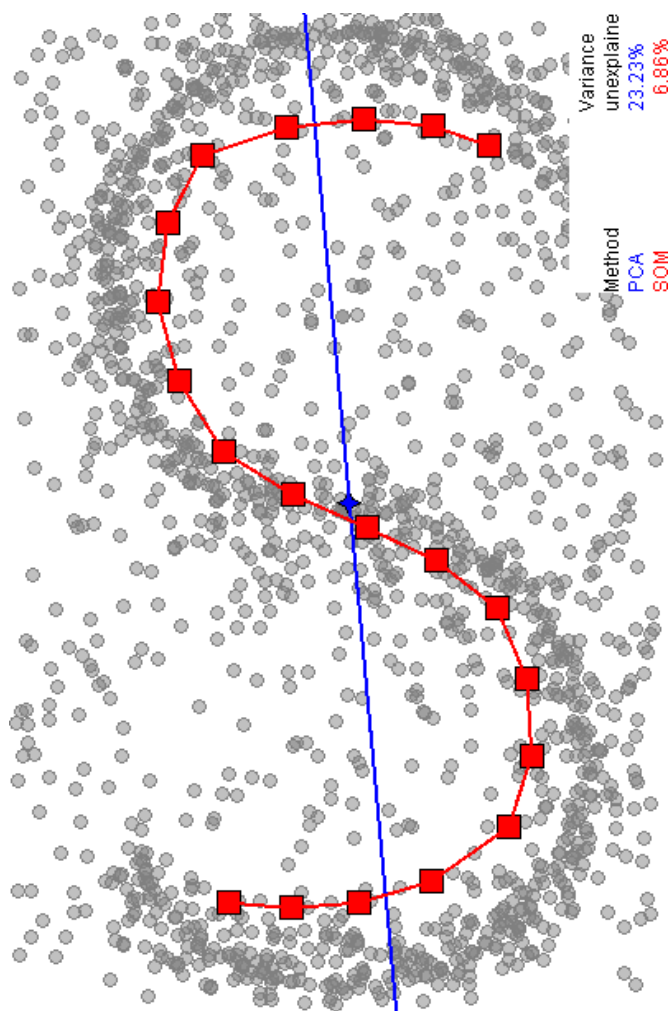
参数可调节

容易受噪声干扰

在大曲率区域存在短路现象

不适用于非凸参数空间

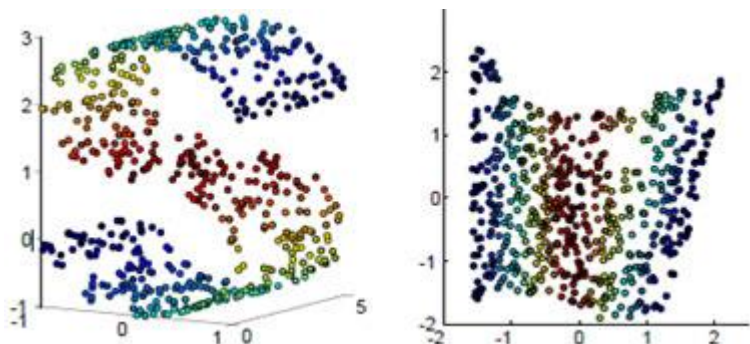
大样本训练速度慢



局部线性嵌入

● Local Linear Embedding (LLE)

保持数据点的原有流形结构



Nonlinear Dimensionality Reduction by Locally Linear Embedding

Sam T. Roweis¹ and Lawrence K. Saul²

SCIENCE VOL 290 22 DECEMBER 2000

前提假设：采样数据所在的低维流形在局部是线性的，每个采样点可以用它的近邻点线性表示。

学习目标：在低维空间中保持每个邻域中的权值不变，即假设嵌入映射在局部是线性的条件下，最小化重构误差。

局部线性嵌入

● Local Linear Embedding (LLE)

保持数据点的原有流形结构

寻找每个样本点的 K 近邻 $x_{ij} (j = 1, \dots, k)$

对每个点用 K 个近邻进行重建，即求一组权值 w_{ij} , $\sum_j w_{ij} = 1$

使 $\min \sum_i |x_i - \sum_j w_{ij} x_{ij}|^2$

求低维空间中的点集 y_i

使 $\min \sum_i |y_i - \sum_j w_{ij} y_{ij}|^2$

局部线性嵌入

● Local Linear Embedding (LLE)

保持数据点的原有流形结构

计算权值，首先构造局部协方差矩阵 C^i

$$C_{jk}^i = (x_i - x_j) \cdot (x_i - x_k)$$

然后，最小化 $\min \sum_i |x_i - \sum_j w_{ij} x_{ij}|^2$

$$s.t. \quad \sum_j w_{ij} = 1$$

最后可求得 $w_{ij} = \frac{\sum_k (C_{jk}^i)^{-1}}{\sum_{jk} (C_{jk}^i)^{-1}}$

局部线性嵌入

● Local Linear Embedding (LLE)

保持数据点的原有流形结构

计算低维数据，最小化 $\min \sum_i |y_i - \sum_j w_{ij} y_{ij}|^2$

可转化为最小化 $\min \sum_{ij} M_{ij} (y_i \cdot y_j)$

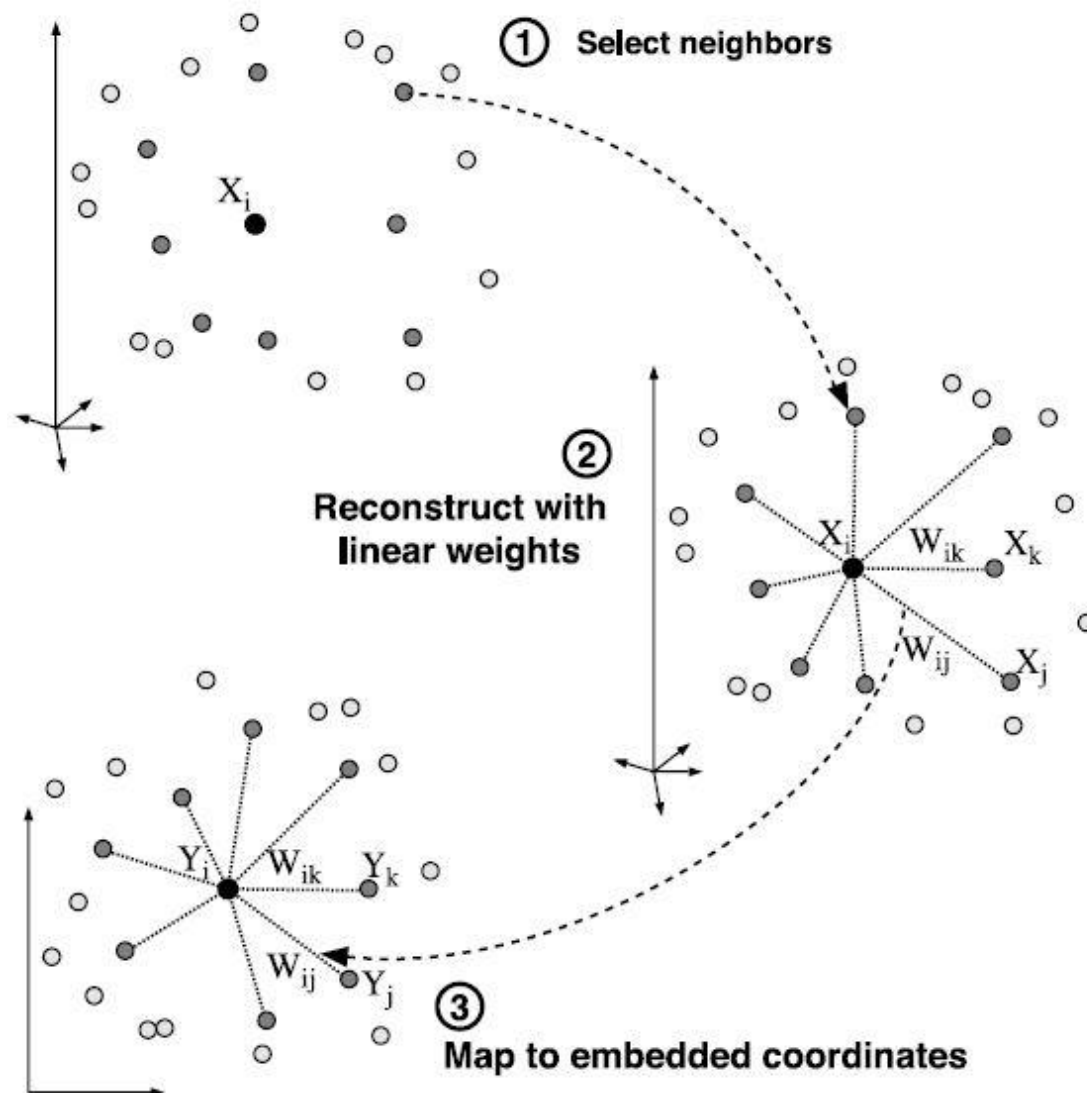
其中， $M = (I - W)^T (I - W)$

求解 $MY = \lambda Y$

取 Y 为 M 的最小 d 个非零特征值所对应的特征向量，最终的输出结果即为 $N \times d$ 大小的矩阵

局部线性嵌入

● 计算步骤



局部线性嵌入

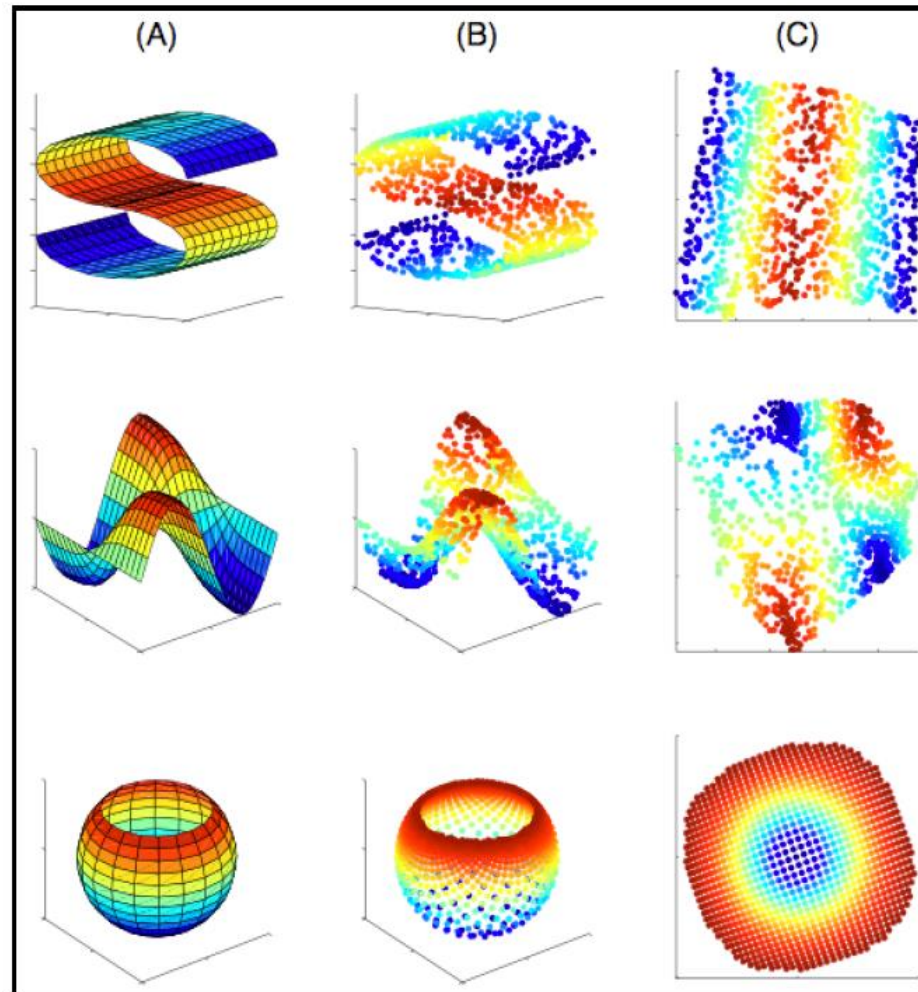
● Local Linear Embedding (LLE)

Surfaces

$N=1000$
inputs

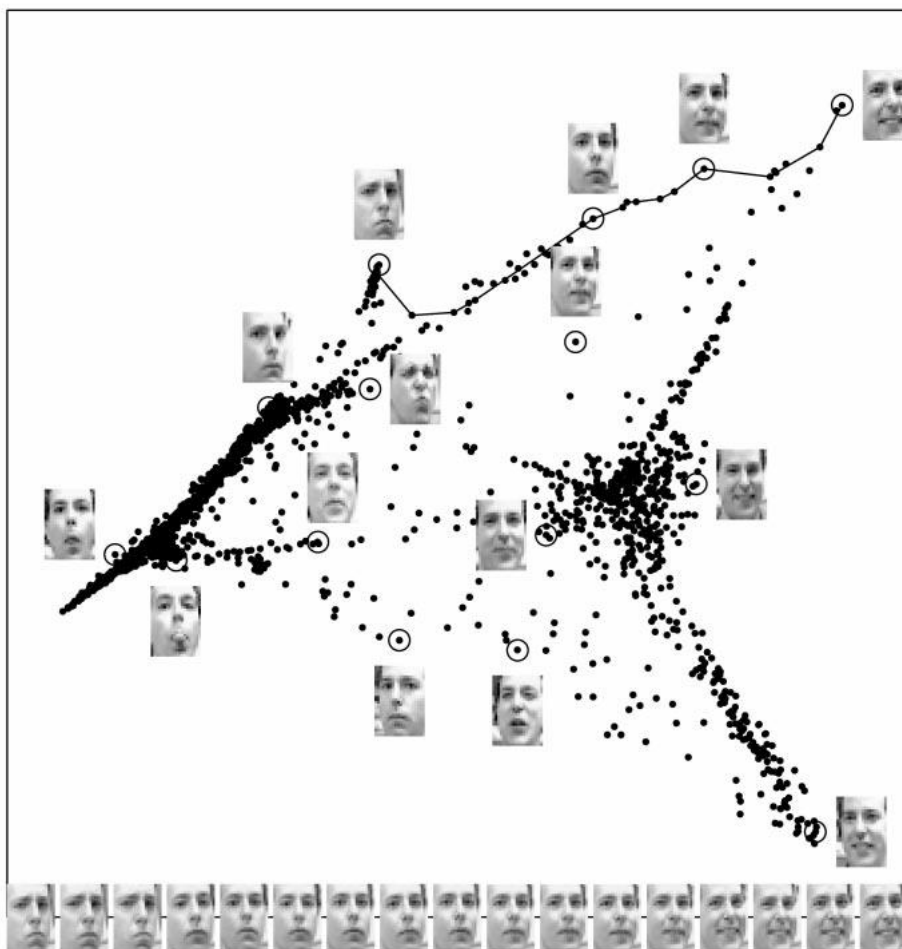
$k=8$
nearest
neighbors

$D=3$
 $d=2$
dimensions



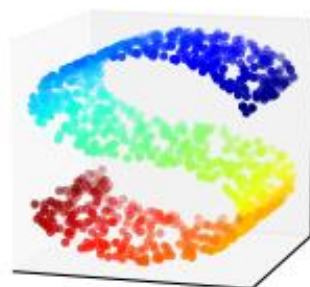
局部线性嵌入

- Local Linear Embedding (LLE)

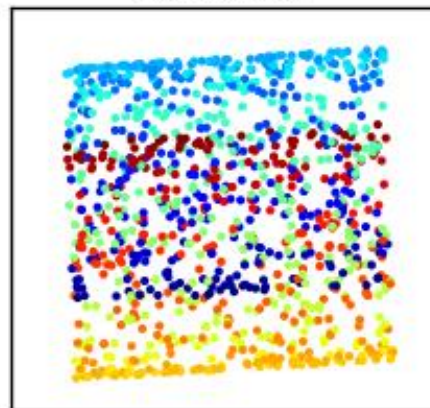


局部线性嵌入

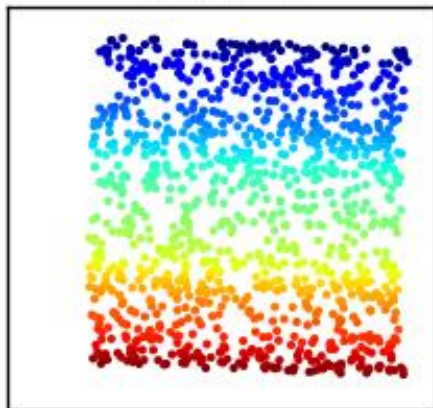
- Local Linear Embedding (LLE)



PCA projection



LLE projection



IsoMap projection

