统计学习第四章

贾金柱

Nov 9, 2017

讲课老师

- 主讲人： 贾金柱
  - Email: jzjia@math.pku.edu.cn
- 助教：肖一君
  - Email: xiaoyijun1994@126.com

第四章 Dimension Reduction

- 课程目标
  - PCA and Kernel PCA
  - MDS
  - Manifold Learning

PCA and Kernel PCA

· PCA

对于观测数据 $X \in \mathrm{R}^{n \times p}$, 将之中心化，得到 X_c.
PCA 的基本步骤是： 1. 计算投影方向 v

$$X_c = UDV^T, or\ X_c^T X_c = VD^2 V^T;$$

2. 得到各个主成分：

$$X_c V = UD.$$

性质： 各个主成分正交

$$\langle X_c V_1, X_c V_2 \rangle = V_1^T X_c^T X_c V_2 = 0.$$

PCA review

Statement: **PCA 完全有任意两个点的内积决定！**

常规的PCA由欧几里德内积决定的。

定义内积 $\langle x_i, x_j \rangle = \sum_k x_{ik} x_{jk}$.

我们需要说明，任意一点 x, 朝主成分方向的投影的值（新向量） $\langle x, V_j \rangle$ 由内积决定，则表明PCA完全由内积决定。

PCA review

注意到

$$\lambda_j V_j = SV_j = X_c^T[X_c V_j]$$

这表明， 存在向量α_j, 使得 $V_j = X_c^T \alpha_j$， 于是

$\langle x, V_j \rangle = \langle x, \alpha_j X_c^T \rangle = \sum_i \alpha_{ji} \langle x, x_c[i, :] \rangle$

现在需要说明 α_j 可以完全由内积来决定。 注意到

$$SV_j = \lambda_j V_j,$$

即

$$[X_c^T X_c] X_c^T \alpha_j = \lambda_j X_c^T \alpha_j.$$

PCA review

易知，满足方程

$$X_c X_c^T \alpha_j = \lambda_j \alpha_j$$

的α_j一定满足上式。

这表明α_j由内积决定。

由 $\alpha_j = \frac{1}{\lambda_j} X_c V_j$, 知 $\|\alpha_j\|_2^2 = \frac{1}{\lambda_j^2} V_j^T X_c^T X_c V_j = \frac{1}{\lambda_j}.$

PCA 步骤

Given： n samples $x_i \in \mathrm{R}^p, i = 1, 2, \ldots, n$

1. 计算（中心化后的）内积矩阵 K with $K_{ij} = \langle x_i - \mu, x_j - \mu \rangle$
2. 计算 K 的特征值λ_i 和它对应的特征向量 α_i
3. 选取第i个你要投影的方向α_i
4. 正则化该方向, 使它的长度是 $\frac{1}{\sqrt{\lambda_i}}$
5. 对于一个（中心化后的）观测点，向该方向投影，得到一个点：
   $\langle x, \alpha_i \rangle = \langle x, \frac{1}{\sqrt{\lambda_i}} \alpha_i X_c^T \rangle = \frac{1}{\sqrt{\lambda_i}} \sum_k \alpha_{ik} \langle x, x_c[k, :] \rangle$

5A. 如果观测点是training data 中的一点，则其投影为

$$\frac{1}{\sqrt{\lambda_i}} \sum_k \alpha_{ik} \langle x_c[t, :], x_c[k, :] \rangle = \frac{1}{\sqrt{\lambda_i}} e_t^T X_c^T X_c \alpha_i = \sqrt{\lambda_i} \alpha_{it}.$$

Kernel PCA

- 以上步骤表明， PCA的计算只和内积有关。
- 因此，可以将PCA 扩展到更广阔的内积空间（RKHS）
- Nonlinear PCA

RKHS 简介

- 它是一个（函数）泛函空间
- 它有内积，该内积定义在两个函数上 $\langle f, g \rangle$
- 它存在一个特殊的函数记作 $k_x(\cdot)$ indexed by x , 它满足 $\langle f, k_x(\cdot) \rangle = f(x), \forall f \in \mathrm{H}$
- 这个 k 称为 再生核。

定义 $k_{x_1}(x_2) := k(x_1, x_2)$, 则

$$\langle k_{x_1}, k_{x_2} \rangle = k(x_1, x_2)$$

结论： 再生核空间的点（函数）的内积，由再生核唯一决定！

Kernel PCA

将数据点 x_i 映射到再生核空间：

$$x_i \rightarrow \Phi(x) := k_x(\cdot) \in \mathrm{H}.$$

然后利用PCA 降维的方法称为Kernel PCA。

Kernel PCA

Given： n samples $x_i \in \mathrm{R}^p, i = 1, 2, \ldots, n$

Compute the inner product matrix K with $K_{ij} = K(x_i, x_j)$

中心化

计算 K 的特征值λ_i 和它对应的特征向量 α_i

选取第i个你要投影的方向α_i

正则化该方向, 使它的长度是 $\frac{1}{\sqrt{\lambda_i}}$

对于一个（中心化后的）观测点，向该方向投影，得到一个点： $\frac{1}{\sqrt{\lambda_i}} \sum_k \alpha_{ik} K(x, x_c[k, :])$

Kernel PCA

中心化的计算：

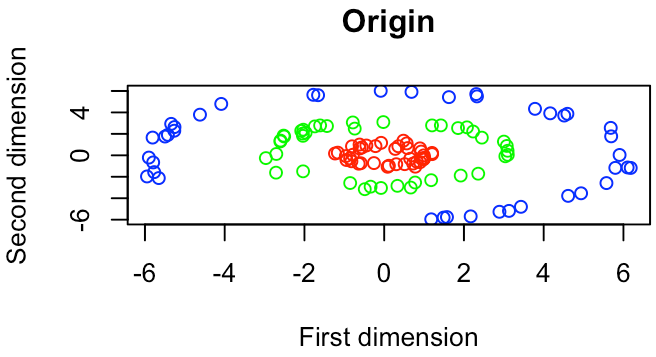
$$
\begin{aligned}
K_{ij}^{c} &= \langle \Phi_i^{c}, \Phi_j^{c} \rangle \\
&= \langle \Phi_i - \frac{1}{n}\sum_k \Phi_k, \Phi_j - \frac{1}{n}\sum_\ell \Phi_\ell \rangle \\
&= K_{ij} - \frac{1}{n}\sum_k K_{kj} - \frac{1}{n}\sum_\ell K(i,l) + \sum_{k\ell} \frac{1}{n^2} K_{k\ell}
\end{aligned}
$$

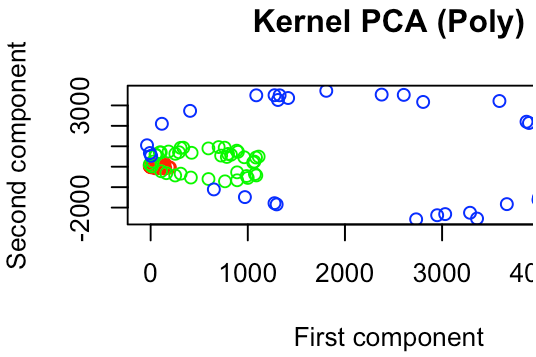
So,

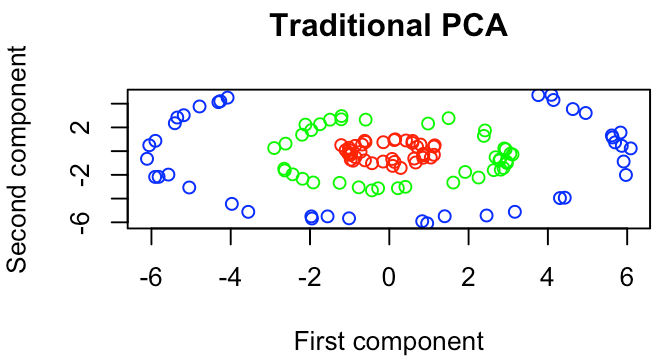
$$
K^{c} = K - 1_n K - K 1_n + 1_n K 1_n,
$$

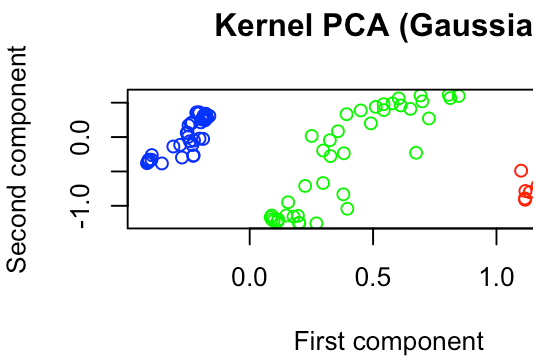
其中 $1_n \in \mathrm{R}^{n \times 1}$ and $1_n(i,j) = \frac{1}{n}$.

Kernel PCA


Origin
Second dimension
-6
0
4
-6
-4
-2
0
2
4
6
First dimension


Kernel PCA (Poly)
Second component
-2000
3000
0
1000
2000
3000
First component


Traditional PCA
Second component
-6
-2
2
-6
-4
-2
0
2
4
6
First component


Kernel PCA (Gaussia
Second component
-1.0
0.0
0.0
0.5
1.0
First component

MDS

MDS 首先定义任意两个数据点之间的相似度（不相似度，dissimilarity）。

然后寻找一个map，将数据点映射到低维空间

达到数据可视化或者降维的效果

MDS (一种数学描述方法)

假设要分析的数据有 n 个物体，现在定义一个距离矩阵

$$\Delta := \begin{pmatrix} \delta_{11} & \delta_{12} & \cdots & \delta_{1n} \\ \delta_{21} & \delta_{22} & \cdots & \delta_{2n} \\ \delta_{n1} & \delta_{n2} & \cdots & \delta_{nn} \end{pmatrix},$$

其中 δ_{ij} 表示第i个物体和第j个物体之间的距离。

MDS 的目的是根据这个距离矩阵，寻找 n 个物体合适的坐标。即，寻找n 个向量 $x_1, x_2, \ldots x_n \in \mathrm{R}^k$, 使得 $\|x_i - x_j\| \approx \delta_{ij}$.

MDS (一种数学描述方法)

classical MDS 可以看成是一个最优化问题：

$$\min_{x_1, x_2, \ldots, x_n} \sum_{i<j} \left(\|x_i - x_j\|^2 - \delta_{ij}^2 \right)^2.$$

如何求解?

MDS

另一种思路

定义距离矩阵 A with $A_{ij} = \|x_i - x_j\|^2$, 这里 $\|x_i - x_j\|$ 代表欧式距离。

可以将距离矩阵转化为内积矩阵.

定义 $P := I - \frac{1}{n} ee^T$, 其中 I 是 $n \times n$ 单位矩阵，e 代表各个元素全为1的向量。

注意： $Pe = e - \frac{1}{n} ee^T e = 0$

任意的$X \in \mathrm{R}^{n \times d}$, $PX = X - \frac{1}{n} ee^T X$ 代表中心化。

MDS

定义矩阵 [C(i,j)] 代表一个矩阵，它的第(i,j)元素为C_{ij}

性质：

$$PAP = -2PX(PX)^T = -2[\langle x_i - \mu, x_j - \mu \rangle]$$

即： 距离矩阵经过变换后，成为内积矩阵。

定理 考虑 n 维对称矩阵 $A \in S_n$, 且 $A_{ij} > 0, A_{ii} = 0.$ 定义 $\bar{A} := -PAP.$ 结论：

（1） $\bar{A}$ 非负定当且仅当 A 是一个距离矩阵。

（2） 如果A 是一个距离矩阵，设 $\bar{A}$ 的秩为k, 则存在 $y_1, y_2, \ldots, y_n \in \mathrm{R}^k$ 使得 $A_{ij} = \|y_i - y_j\|^2.$

MDS

- 上述定理告诉我们，给定了一个 $n \times n$ 的距离矩阵 A，可以找到 n 个 k 维 点（坐标），使得这些点的两两距离的平方，恰好是 A_{ij}.

- 定理的证明

（1）" $\Rightarrow$." 因为 $\bar{A}$ 非负定，设它的秩为k, 则存在 $X \in \mathrm{R}^{n \times k}$, 使得 $\bar{A} = XX^T$. 记 X 的行向量为 $x_1, x_2, \ldots, x_n$, 则 $\bar{A}_{ij} = \langle x_i, x_j \rangle = -A_{ij} + A_{ij}^R + A_{ij}^C - A_{ij}^{RC}$. 定义 $y_i = \frac{1}{\sqrt{2}} x_i$. 则

$$\begin{aligned} 2(y_i - y_j)^2 &= (x_i - x_j)^2 \\ &= \langle x_i, x_i \rangle + \langle x_j, x_j \rangle - 2\langle x_i, x_j \rangle \\ &= 2A_{ij} \end{aligned}$$

定理的证明

（1）" $\Leftarrow$." 现在假定 A 是一个距离矩阵。即 存在 $y_1, y_2, \ldots, y_n$,使得 $A_{ij} = (y_i - y_j)^2$, 则 $\bar{A} = (PY)(PY)^T$.

从（1）的证明看出，结论（2）是显然的。只要找到 $x_1, x_2, \ldots, x_n$, y_i 可以选取 $\frac{1}{\sqrt{2}} x_i$.

寻找坐标

- 目标： 给定一个距离矩阵 $A \in \mathrm{R}^{n\times n}$, 寻找坐标 $y_1, y_2, \ldots, y_n \in \mathrm{r}^k$, 使得 $\|y_i - y_j\|_2^2 \approx A_{ij}$
- 算法：
  - 计算 $\bar{A} = -PAP$
  - 计算 $\bar{A}$ 的特征值分解， $\bar{A} = UDU^T$
  - 得到 Gram vectors $\bar{A} = [UD^{1/2}][UD^{1/2}]^T := XX^T$
  - 令 $y_i = \frac{1}{\sqrt{2}} x_i$, 其中 x_i 代表 X 的第i行。
- 注1： 如果$\bar{A}$ 的秩为k, 则 $D \in \mathrm{R}^{k\times k}$, 于是 $X = UD^{1/2} \in \mathrm{R}^{n\times k}$
- 注2: 如果$\bar{A}$ 的秩为n, 则$D \in \mathrm{R}^{n\times n}$, 于是 $X = UD^{1/2} \in \mathrm{R}^{n\times n}$。此时可以截断 D得到一个（类似于PCA）的近似。

Manifold Learning

- MDS 给出了数据的低维表示
- 但是它并没有对数据是怎样产生的，进行显示建模
- 对数据所在的低维流形直接建模的方法，常用的方法是 Isomap 和 LLE （locally Linear Embedding）.

Isomap

- Reference. J. B. Tenenbaum, V. de Silva, J. C. Langford, A Global Geometric Framework for Nonlinear Dimensionality Reduction, Science 290, (2000), 2319–2323.
- 主体思想： 首先构造一个距离矩阵，但是这个距离矩阵不是欧氏距离，它使用流行上的测地距离。使用图来近似流形，然后使用图上的最小路径逼近测地距离。

LLE

- Reference. S. T. Roweis and L. K. Saul, Nonlinear Dimensionality Reduction by Locally Linear Embedding, Science Vol 290, 22 December 2000, 2323–2326.

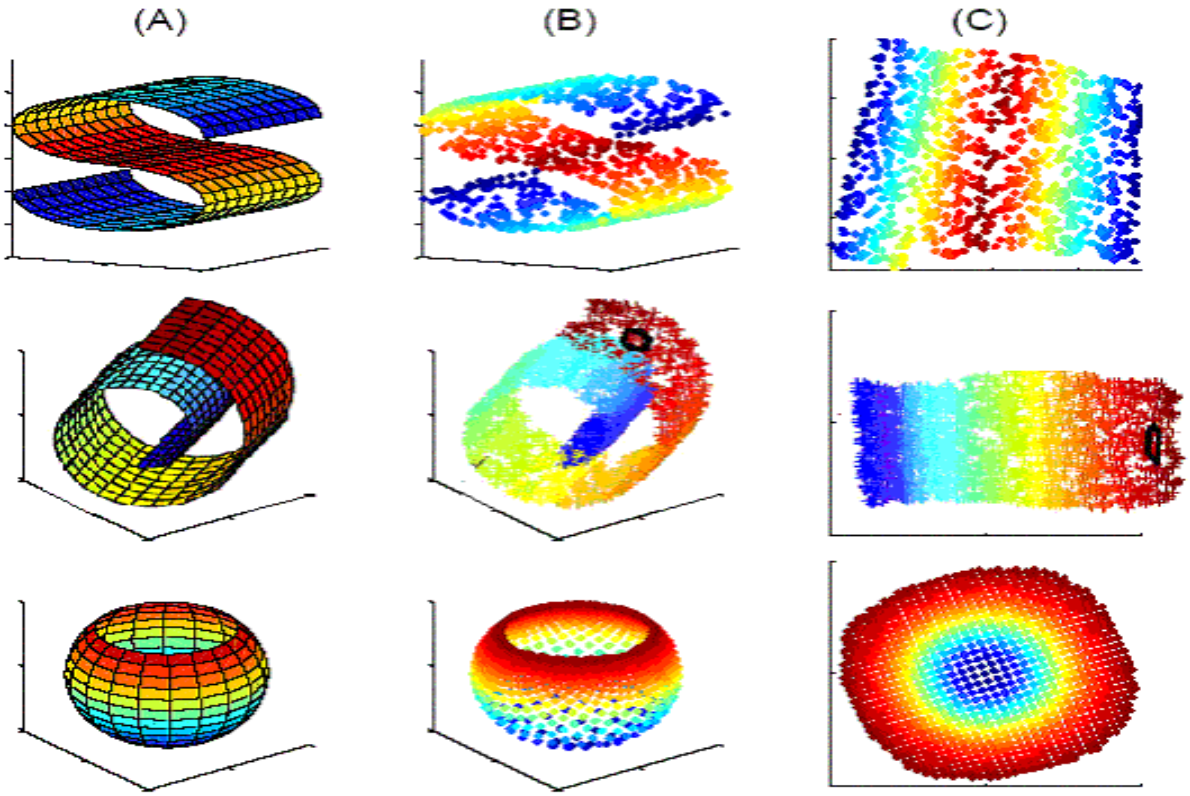


图: LLE 效果图

LLE 算法

该算法归结为三步。（1）寻找每个样本点的k个近邻点；（2）由每个样本点的近邻点计算出该样本点的局部重建权值矩阵；（3）由该样本点的局部重建权值矩阵和其近邻点计算出该样本点的输出值。

· 第一步：

计算每一个样本点的 k 个近邻点。k 是一个预先设定的值。可以选用欧式距离，也可以使用类似于 Isomap 的测地距离。

· 第二步：计算局部重建权值矩阵。

$$\min_{w_{ij}} (x_i - \sum_j w_{ij} x_{ij})^2,$$

such that $\sum_j w_{ij} = 1$.

LLE 算法

- 第三步： 将样本点映射到低维空间。即寻找 n 个低维坐标 $y_1, y_2, \ldots, y_n \in \mathrm{R}^k$, 使得他们保持 $x_1, x_2, \ldots x_n$ 之间的关系。

$$\min \sum_{i=1}^{n} (y_i - \sum_{j=1}^{k} w_{ij} y_{ij})^2$$

这个解不唯一，我们加进一个约束

$$Y^T Y = I.$$

谱聚类算法（spectral clustering)

- 目标： 给定一个距离矩阵（相似矩阵，不相似度矩阵），将数据点分类
- 谱聚类算法，首先构造一个图。图上的每一个点代表一个数据点。两个点之间给一个权重，如果两个点离得很远，权重接近于0.

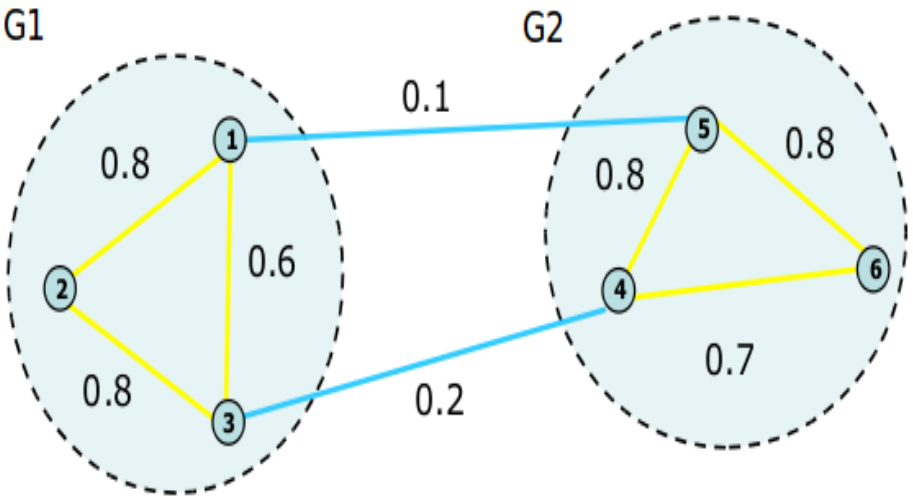


图: 谱聚类示意图

谱聚类算法（spectral clustering)

考虑最优图分割方法。将图cut为G_1 和 G_2 两个部分，最小化如下的损失函数：

$$cut(G_1, G_2) = \sum_{i \in G_1, j \in G_2} e_{ij}.$$

用 $[q_1, q_2, \ldots, q_n]$ 代表每个点的label $\in \{-1, 1\}$. 则损失函数可以表示为

$$cut(G_1, G_2) = \sum_{i \in G_1, j \in G_2} e_{ij} \propto \sum_{i=1}^{n} \sum_{j=1}^{n} e_{ij}(q_i - q_j)^2$$

谱聚类算法（spectral clustering)

注意

$$\begin{aligned}\sum_{i=1}^{n}\sum_{j=1}^{n} e_{ij}(q_i - q_j)^2 &= \sum_{i=1}^{n}\sum_{j=1}^{n} e_{ij}(q_i^2 - 2q_iq_j + q_j^2) \\ &= 2\sum_{i=1}^{n} q_i^2(\sum_{j=1}^{n} e_{ij}) - \sum_{i=1}^{n}\sum_{j=1}^{n} e_{ij}q_iq_j \\ &= 2q^T[D - E]q,\end{aligned}$$

其中 $E = [e_{ij}]$ 是边的权重矩阵， D是一个对角矩阵，$D_{ii} = \sum_{j=1}^{n} e_{ij}$

定义 $L = D - E$. 则 q minimize q^TLq with $q^Tq = n$.

· 特征值!

Normalized cut 方法

- normalized cut

$$\tilde{L} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} E D^{-1/2}$$

作业

due： Nov 30.

补充完整 LLE 算法。