

## Lecture 19: 流形学习

2024.6.3

Lecturer: 丁虎

Scribe: 王运韬, 莫官霖

## 1 动机与基本概念

多维尺度分析 (MDS) 是一种非线性降维技术, 旨在将高维数据投影到低维空间, 同时尽可能保留数据点之间的相似性或距离关系。

**核心思想:** 给定一组对象间的相异度 (距离) 矩阵  $D \in \mathbb{R}^{n \times n}$ , 其中  $D_{ij} = \text{dist}(x_i, x_j)$ , MDS 寻找低维表示  $\{y_1, \dots, y_n\} \in \mathbb{R}^{n \times k}$  (通常  $k = 2$  或  $3$ ), 使得这些点之间的欧氏距离尽可能接近原始距离。

## 2 经典 MDS 算法

### 2.1 输入与输出

- **输入:** 距离矩阵  $D \in \mathbb{R}^{n \times n}$ , 其中  $D_{ij} = \text{dist}(x_i, x_j)$
- **输出:** 低维表示  $Y = \{y_1, \dots, y_n\} \in \mathbb{R}^{n \times k}$

### 2.2 算法步骤

1. 计算平方距离矩阵:

$$D^{(2)} = D \odot D$$

2. 构造双中心化矩阵:

$$B = -\frac{1}{2}JD^{(2)}J$$

其中  $J = I_n - \frac{1}{n}11^T$  为中心化矩阵,  $1$  为全 1 向量

3. 特征分解:

$$B = V\Lambda V^T$$

其中  $\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$ ,  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$

#### 4. 选择前 $k$ 个特征值及对应特征向量:

$$Y = V_k \Lambda_k^{1/2}$$

其中  $V_k$  为前  $k$  个特征向量组成的矩阵,  $\Lambda_k = \text{diag}(\lambda_1, \dots, \lambda_k)$

可以看到, MDS 算法实质上就是 PCA 的一种等价形式.

从距离矩阵  $D$  重建内积矩阵  $B$ :

$$B_{ij} = \frac{1}{2}(\|x_i\|^2 + \|x_j\|^2 - D_{ij}^2)$$

### 3 ISOMAP 算法

ISOMAP (等距映射) 是 MDS 的扩展, 适用于非线性流形数据。

#### 3.1 算法原理

##### 1. 构建邻域图:

- 对每个点  $x_i$ , 找到其  $k$  近邻或  $\epsilon$ -邻域
- 构建邻接图  $G$ , 边权重为欧氏距离.

##### 2. 计算最短路径距离: 使用 Dijkstra 或 Floyd-Warshall 算法计算图上所有点对之间的最

短路径距离  $\tilde{D}$ , 从而近似测地距离, 如图 1.  $\tilde{D}_{ij} = \begin{cases} D_{ij} & \text{如果 } j \in \mathcal{N}(i) \\ \text{最短路径距离} & \text{否则.} \end{cases}$

##### 3. 应用经典 MDS: 对 $\tilde{D}$ 应用 MDS 得到低维嵌入.

缺点: 复杂度太高.

### 4 局部线性嵌入 (Locally Linear Embedding, LLE)

LLE 是另一种非线性降维方法, 强调局部线性结构的保持。它有广泛的应用 [2].

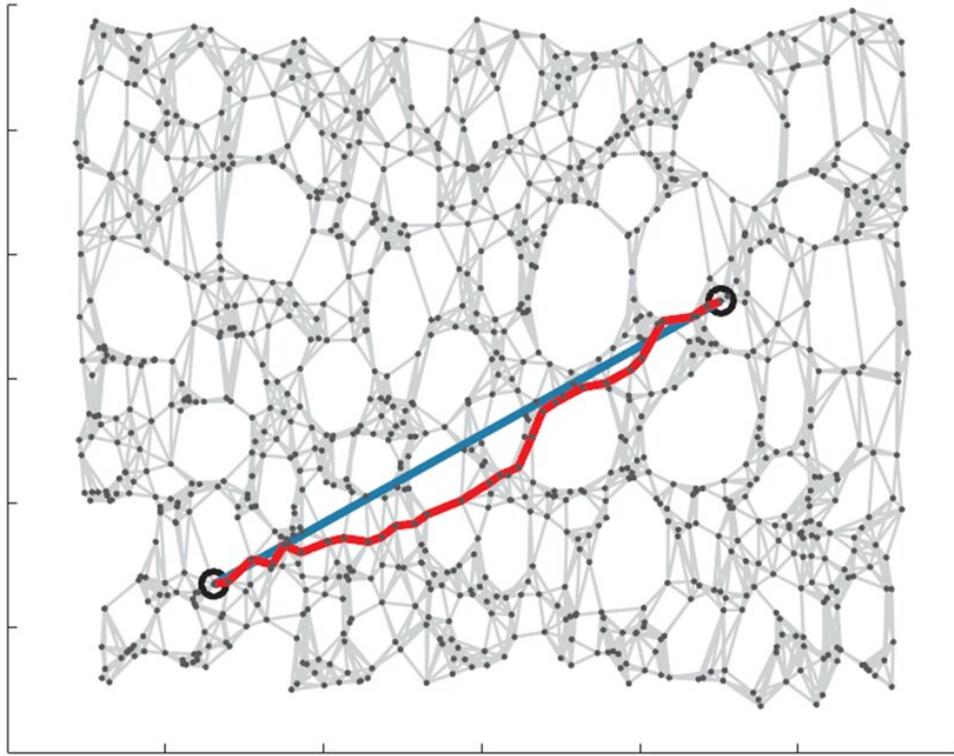


Figure 1: Isomap 算法中的距离为红线长度, 相比蓝线的欧氏距离更能体现内蕴性质.

## 4.1 算法步骤

1. **选择邻域**: 对每个点  $x_i \in \mathbb{R}^n$ , 找到  $k$  近邻  $\mathcal{N}(i)$
2. **计算局部重建权重**, 其中  $W_{ij}$  为希望找到的权重:

$$\begin{aligned} \min_W \sum_{i=1}^n \|x_i - \sum_{j \in \mathcal{N}(i)} W_{ij} x_j\|^2 \\ \text{s.t. } \sum_{j \in \mathcal{N}(i)} W_{ij} = 1 \end{aligned}$$

这里的约束项, 从几何上看, 保证了在变换  $x \mapsto x + c$  下目标函数的不变性.

3. **计算低维嵌入**. 求降维后的数据点矩阵  $Y \in \mathbb{R}^{n \times k}$ , 使用一个简单的凸优化:

$$\begin{aligned} \min_Y \sum_{i=1}^n \|y_i - \sum_{j \in \mathcal{N}(i)} W_{ij} y_j\|^2 \\ \text{s.t. } \frac{1}{n} Y^T Y = I, \quad \sum_{i=1}^n y_i = 0 \end{aligned}$$

*Remark 4.1.* 权重矩阵  $W$  的求解:

$$W_i = \frac{C^{-1} \mathbf{1}}{\mathbf{1}^T C^{-1} \mathbf{1}}$$

其中  $C_{jk} = (x_i - x_j)^T (x_i - x_k)$  为局部协方差矩阵.

## 5 Doubling Dimension and R-Net

在各种计算几何领域的算法中, 我们常常要反复搜索一个点的邻居。在先前的课程中我们提到了可以用 LSH 来加速这一过程。进一步地, 当数据处于一个流形上, 我们是否可以使用其他方法来加速这一近邻搜索的过程?

### 5.1 Doubling Dimension

倍增维数 (Doubling Dimension) 是数据内蕴维度的一种。

若对于任意的  $p \in P$ ,  $r > 0$ , 有

$$\frac{|Ball(p, 2r)|}{|Ball(p, r)|} \leq 2^\lambda$$

则  $\lambda$  称为  $P$  的倍增维数. 一般地,  $\mathbb{R}^d$  的倍增维数为  $\Theta(d)$ , 但如果数据集  $P$  分布在一个低维流形上, 那么  $P$  的 Doubling Dimension 为  $\Theta(1)$ . 对于这样的  $P$ , 我们可以构造一个  $r$ -net  $Q$  来近似.

**Definition 5.1** ( $r$ -net). 给定度量空间  $(X, d)$  和半径  $r > 0$ , 子集  $N \subseteq X$  称为  $r$ -net 如果满足:

1. (覆盖性) 对所有  $x \in X$ , 存在  $y \in N$  使得  $d(x, y) \leq r$
2. (分离性) 对所有  $y, z \in N$ ,  $d(y, z) > r$

由定义和三角不等式, 有如下结论:

**Lemma 5.2.** 对于集合  $P$  和它的  $r$ -net  $Q$

$$Ball(P, r) \subseteq \bigcup_{\substack{q' \in Ball(q, 3r) \\ q' \in Q}} Ball(q', r)$$

根据上述的结论, 我们可以将近邻的搜索范围降低到  $Q \cap Ball(q', r)$ .

我们有重要的推论:

**Lemma 5.3.** 设度量空间  $M = (X, dist)$ ,  $S \subset X$ , 其中  $(S, dist)$  仍然是一个度量空间. 令  $N$  为  $S$  的一个  $r$ -net, 并设  $S$  的直径为  $D$ , 则:

$$|N| \leq \left(\frac{2D}{r}\right)^{ddim(M)}$$

*Proof.* 由于  $S \subset X$ , 我们至少需要  $\lambda(M)$  个直径为  $D/2$  的集合覆盖  $S$ . 每个这样的子集  $S_i \subset S$  仍然是  $M$  的子集, 因此根据相同的定义, 我们可以用  $\lambda(M)$  个直径为  $D/4$  的集合覆盖每个  $S_i$ . 由于有  $\lambda(M)$  个  $S_i$ , 总共需要  $\lambda(M)^2$  个直径为  $D/4$  的集合覆盖  $S$ .

重复这一过程, 我们可以覆盖  $S$  使用:

- $\lambda(M)$  个直径为  $D/2$  的集合;
- $\lambda(M)^2$  个直径为  $D/4$  的集合;
- $\lambda(M)^3$  个直径为  $D/8$  的集合;
- ...
- $\lambda(M)^{\log_2(\frac{2D}{r})}$  个直径为  $D/2^{\log_2(\frac{2D}{r})} = \frac{r}{2}$ .

由于  $\frac{r}{2}$  直径的集合最多只能包含一个  $r$ -net 中的点，因此  $N$  的大小受到以下限制：

$$|N| \leq \lambda(M)^{\log_2(\frac{2D}{r})} = \left(\frac{2D}{r}\right)^{\text{ddim}(M)}.$$

□

$r$ -net 的建立方法: Gonzalez 算法.

1. 从  $P$  中任取一个点  $p_1, S = \{p_1\}$
2. Do  $p_j = \arg \max d(p_j, S), S = S \cup \{p_j\}$ .  
While  $d(p_j, S) > r$
3. Return  $S$

根据引理 5.3,  $|S| < (\frac{D}{r})^\lambda$ . 时间复杂度为  $\Theta((\frac{D}{r})^\lambda nd)$ .

**问题：如何降低建立  $r$ -net 的复杂度？**

Friend List 方法 [1].

**核心思想：**只在“局部”更新距离，而不是全局扫描。每次加入一个新的 center 时，那些“距离当前 center 集合的距离”会产生变化的数据点，只会存在于一些“可能受影响的簇”内。我们用 friend list 把“可能受影响的簇”限定在常数个（与度量的倍增维度  $\lambda$  有关），从而把一次迭代的代价降到  $\lambda^{O(1)}$ ，而非朴素 Gonzalez 中的  $O(nd)$ 。

**数据结构：**

1. Max heap  $H_\alpha$ ：维护每个数据点  $q$  到类中心集合  $S$  的距离  $\alpha_q$ 。（大小为  $n$ ）
2.  $C(p_i)$ ：维护每个类中心  $p_i$  负责的点集。（总大小为  $n$ ）
3. Friend list  $F(p_i)$ ：与类中心  $p_i$  距离  $\leq 4r_k$  的其他类中心。

**单次迭代流程：**

1. 从  $H_\alpha$  弹出距离当前类中心集合  $S$  距离最远的点  $p_j$ ，设其原本所属的类中心是  $c$ 。
2. 建立新的类中心  $p_j$ 。

3. 局部更新：只扫描两类簇中的所有数据点： $c$  负责的所有点、和  $c$  的 **friend list** 中的那些类中心负责的所有点。对每个需要更新的点  $q$ ，我们重新计算  $\alpha_q$ ，并在需要时把  $q$  迁入新的簇  $C(p_j)$ 。

4. 生成  $p_j$  的 friend list。

#### 时间复杂度改进：

由于每个类中心的 friend list 规模最多不超过  $O(c'^{\lambda})(c'$  为常数)，总的时间复杂度不超过  $O(c'^{\lambda}n \log n \log \frac{D}{r})$ ，进一步地，可以将这个复杂度降低至  $O(c'^{\lambda}n \log n)$ 。

## References

- [1] S. Har-Peled and M. Mendel. Fast construction of nets in low dimensional metrics, and their applications. In *Proceedings of the twenty-first annual symposium on Computational geometry*, pages 150–158, 2005.
- [2] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.