

## 摘 要

随着大数据时代的到来，科学发展正在迈入一个新阶段，科研的方法也从之前的实验型科研（Experimental Science）、理论型科研（Theoretical Science）、计算型科研（Computational Science）推进到第四范式——数据密集型科研（Data-intensive Science）。

现在面临的是大数据时代，数据分析被认为是继实验、理论、仿真之后的第四种科学的研究范式。本书介绍流形学习和稀疏表示的基本方法，侧重方法和实际应用，不侧重理论。

**关键词：**流形学习、稀疏表示



# **Manifold Learning and Sparse Representation**

Zhouchen Lin (Computer Application)

Directed by

Blah, Blah, Blah, ...

**关键词：** Manifold Learning, Sparse Representation



# 第一章 引言

## 1.1 流形学习与稀疏表示简介

## 1.2 本文的组织结构

本文总共由五章组成。

第二章介绍线性代数补充知识。

第三章介绍数据几何。

第四章介绍线性降维技术，包括主成分分析、多维标度法、随机投影法。

第五章介绍非线性降维技术，包括ISOMAP、LLE、LTSA、LE、Dmaps、MVU。

第六章介绍一阶稀疏表示。



## 第二章 线性代数补充知识

(本章内容主要取自[4])

统计学是建立在统计数据之上的。多变量统计分析是建立在用矩阵表示样本的基础上的。因此，矩阵也就成为多变量统计学的基础。本章仅对已有矩阵一般知识的读者补充某些必要的内容。本书仅考虑实矩阵。

### 2.1 方阵的迹

设  $\mathbf{A} = (A_{ij})$  为  $n$  阶方阵，其迹定义为：

$$\text{tr}(\mathbf{A}) \triangleq \sum_{i=1}^n A_{ii}. \quad (2.1)$$

迹的基本性质有：

$$(1) \text{ tr}(a\mathbf{A} + b\mathbf{B}) = a\text{tr}(\mathbf{A}) + b\text{tr}(\mathbf{B}).$$

$$(2) \text{ tr}(\mathbf{AB}) = \text{tr}(\mathbf{BA}), \text{ 有时写成内积形式 } \langle \mathbf{A}, \mathbf{B}^T \rangle. \text{ Proof: } \text{tr}(\mathbf{AB}) = \text{tr}(\mathbf{BA}) = \sum_{ij} A_{ij} B_{ji}.$$

$$(3) \text{ 迹在相似变换下不变：对任何可逆矩阵 } \mathbf{X}, \text{ 恒有 } \text{tr}(\mathbf{XAX}^{-1}) = \text{tr}(\mathbf{A}).$$

$$(4) \|\mathbf{A}\|_F = (\text{tr}(\mathbf{A}^T \mathbf{A}))^{1/2}. \text{ Proof: 用(2).}$$

**练习1.** 求矩阵

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

的迹。

### 2.2 对称矩阵的特征值分解

若  $\mathbf{A}$  为  $n$  阶对称矩阵，则存在正交矩阵  $\mathbf{U} = (\mathbf{u}_1, \dots, \mathbf{u}_n)$  及对角矩阵  $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$ ，使得

$$\mathbf{A} = \mathbf{U}\Lambda\mathbf{U}^T, \quad (2.2)$$

其中 $\lambda_i$ 称为 $\mathbf{A}$ 的特征值， $\mathbf{u}_i$ 称为 $\lambda_i$ 的特征向量。当 $\mathbf{A}$ 是非负定矩阵（记作 $\mathbf{A} \succcurlyeq \mathbf{0}$ ）时， $\lambda_i$ 均非负；当 $\mathbf{A}$ 是正定矩阵（记作 $\mathbf{A} \succ \mathbf{0}$ ）时， $\lambda_i$ 均为正数。一般把特征值从大到小排列。对于非负定矩阵 $\mathbf{A}$ ，可以定义

$$\mathbf{A}^\alpha \triangleq \mathbf{U} \text{diag}(\lambda_1^\alpha, \dots, \lambda_n^\alpha) \mathbf{U}^T, \quad (2.3)$$

其中 $\alpha$ 是任意正实数。对于正定矩阵 $\mathbf{A}$ ， $\alpha$ 可以是任意实数。最重要的 $\alpha$ 值是 $1/2$ ，即给非负定矩阵开根号。

**练习2.** 求矩阵

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 5 & 6 \\ 3 & 6 & 9 \end{pmatrix}$$

的特征值分解。

### 2.2.1 特征值的性质

设 $\mathbf{A}$ 为 $n$ 阶对称矩阵，其特征值为 $\lambda_1 \geq \dots \geq \lambda_n$ ，则

$$\lambda_1 = \sup_{\mathbf{x} \neq \mathbf{0}} \frac{\mathbf{x}^T \mathbf{A} \mathbf{x}}{\mathbf{x}^T \mathbf{x}} = \sup_{\|\mathbf{x}\|=1} \mathbf{x}^T \mathbf{A} \mathbf{x}, \quad (2.4)$$

$$\lambda_n = \inf_{\mathbf{x} \neq \mathbf{0}} \frac{\mathbf{x}^T \mathbf{A} \mathbf{x}}{\mathbf{x}^T \mathbf{x}} = \inf_{\|\mathbf{x}\|=1} \mathbf{x}^T \mathbf{A} \mathbf{x}. \quad (2.5)$$

Proof: 使用Lagrange乘子法。

$\mathbf{AB}$ 和 $\mathbf{BA}$ 的非零特征值完全相同。

## 2.3 矩阵的奇异值分解

对于任意 $m \times n$ 矩阵 $\mathbf{A}$ （除非另外指出，都假定 $m \geq n$ ），都存在正交矩阵 $\mathbf{U} = (\mathbf{u}_1, \dots, \mathbf{u}_m)$ 、 $\mathbf{V} = (\mathbf{v}_1, \dots, \mathbf{v}_n)$  和对角矩阵 $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_n)$ ，使得

$$\mathbf{A} = \mathbf{U} \begin{pmatrix} \Sigma \\ \mathbf{0} \end{pmatrix} \mathbf{V}^T, \quad (2.6)$$

其中 $\sigma_i$ 均非负，称为 $\mathbf{A}$ 的奇异值， $\mathbf{u}_i$ 称为 $\sigma_i$ 的左奇异向量， $\mathbf{v}_i$ 称为 $\sigma_i$ 的右奇异向量， $i = 1, \dots, n$ 。一般把奇异值从大到小排列。奇异值分解分完全奇异值分解和瘦型奇异值分解两种。

矩阵的奇异值分解和特征值分解之间的关系：

$$\sigma_i(\mathbf{A}) = \sqrt{\lambda_i(\mathbf{A} \mathbf{A}^T)} = \sqrt{\lambda_i(\mathbf{A}^T \mathbf{A})},$$

$\mathbf{A}$ 的左奇异向量为 $\mathbf{AA}^T$ 的特征向量， $\mathbf{A}$ 的右奇异向量为 $\mathbf{A}^T \mathbf{A}$ 特征向量。

对于对称矩阵， $\sigma_i = |\lambda_i|$ ， $i = 1, \dots, n$ 。

练习3. 求矩阵

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{pmatrix}$$

的完全奇异值分解和瘦型奇异值分解。

### 2.3.1 奇异值的极值性质

设 $\mathbf{A}$ 为 $m \times n$ 矩阵，其奇异值为 $\sigma_1 \geq \dots \geq \sigma_n$ ，则

$$\sigma_1 = \sup_{\mathbf{x} \neq \mathbf{0}} \frac{\|\mathbf{Ax}\|}{\|\mathbf{x}\|} = \sup_{\|\mathbf{x}\|=1} \|\mathbf{Ax}\|, \quad (2.7)$$

$$\sigma_n = \inf_{\mathbf{x} \neq \mathbf{0}} \frac{\|\mathbf{Ax}\|}{\|\mathbf{x}\|} = \inf_{\|\mathbf{x}\|=1} \|\mathbf{Ax}\|. \quad (2.8)$$

## 2.4 广义特征值和广义特征向量

设 $\mathbf{A}$ 和 $\mathbf{B}$ 是 $n$ 阶对称矩阵， $\mathbf{B} \succ \mathbf{0}$ ，则方程

$$|\mathbf{A} - \lambda \mathbf{B}| = 0 \quad (2.9)$$

的根称为 $\mathbf{A}$ 相对于 $\mathbf{B}$ 的特征值，或称为广义特征值。满足 $\mathbf{A}\alpha = \lambda \mathbf{B}\alpha$ 的向量称为 $\mathbf{A}$ 相对于 $\mathbf{B}$ 的特征向量，或广义特征向量。容易证明 $\lambda$ 和 $\mathbf{B}^{1/2}\alpha$ 是 $\mathbf{B}^{-1/2}\mathbf{AB}^{-1/2}$ 的特征值和特征向量。

## 2.5 投影矩阵

对称幂等矩阵称为投影矩阵，即

$$\mathbf{A}^T = \mathbf{A}, \quad \mathbf{A}^2 = \mathbf{A}. \quad (2.10)$$

投影矩阵具有如下性质：

- (1) 若 $\mathbf{A}$ 是秩为 $r$ 的投影矩阵，则 $\mathbf{A}$ 必有 $r$ 个特征值为1，而其他的特征值全是0。因此，满秩的投影矩阵必是单位矩阵 $\mathbf{I}$ 。
- (2) 若 $\mathbf{A}$ 是投影矩阵，则 $\mathbf{I} - \mathbf{A}$ 也是投影矩阵。
- (3) 若 $\mathbf{A}$ 是投影矩阵，则 $\text{tr}(\mathbf{A}) = \text{rank}(\mathbf{A})$ 。
- (4) 若 $\mathbf{A}$ 与 $\mathbf{B}$ 是投影矩阵，且 $\mathbf{A} + \mathbf{B} = \mathbf{I}$ ，则 $\mathbf{AB} = \mathbf{BA} = \mathbf{0}$ 。

- (5) 若 $\mathbf{X}$ 是 $n \times p$ 矩阵,  $n \geq p$ ,  $\text{rank}(\mathbf{X}) = p$ , 则 $\mathbf{P}_X = \mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T$ 是投影矩阵。特别地,  $\mathbf{P} \triangleq \mathbf{I}_{n \times n} - \frac{1}{n} \mathbf{J}_{n \times n}$ 是投影矩阵, 其中 $\mathbf{J}_{n \times n}$ 是 $n$ 阶全1矩阵。 $\mathbf{P}$ 也称为中心化算子矩阵。

注:  $\mathbf{J} = \mathbf{1}\mathbf{1}^T$ 。中心化时,  $\mathbf{Y} \rightarrow \mathbf{Y}\mathbf{P} = \mathbf{Y} - \left(\frac{1}{n}\mathbf{Y}\mathbf{1}\right)\mathbf{1}^T$ .

**练习4.** 给定

$$\mathbf{X} = \begin{pmatrix} 1 & 4 & 6 \\ 7 & 2 & 5 \\ 9 & 8 & 3 \\ 10 & 11 & 12 \end{pmatrix},$$

求相应的投影矩阵 $\mathbf{P}_X$ 。

## 2.6 矩阵的拉直

所谓矩阵的拉直, 就是把矩阵拉成一个长的列向量。本书采用按列拉直, 即把矩阵的列依次拼在一起得到一个列向量。设 $\mathbf{A} = (\mathbf{a}_1, \dots, \mathbf{a}_n)$ , 则

$$\text{vec}(\mathbf{A}) \triangleq \begin{pmatrix} \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_n \end{pmatrix}. \quad (2.11)$$

矩阵的拉直有如下性质:

$$(1) \text{ vec}(a\mathbf{A} + b\mathbf{B}) = a\text{vec}(\mathbf{A}) + b\text{vec}(\mathbf{B}).$$

$$(2) \text{ tr}(\mathbf{A}^T \mathbf{B}) = (\text{vec}(\mathbf{A}))^T \text{vec}(\mathbf{B}) = \sum_{ij} A_{ij} B_{ij}。由此可以得到矩阵的Cauchy-Schwarz不等式: (\text{tr}(\mathbf{A}^T \mathbf{B}))^2 \leq \text{tr}(\mathbf{A}^T \mathbf{A})\text{tr}(\mathbf{B}^T \mathbf{B})。$$

## 2.7 Kronecker乘积

设 $\mathbf{A}$ 为 $m \times p$ 矩阵,  $\mathbf{B}$ 为 $n \times q$ 矩阵, 它们的Kronecker乘积定义为:

$$\mathbf{A} \otimes \mathbf{B} \triangleq (A_{ij}\mathbf{B}) = \begin{pmatrix} A_{11}\mathbf{B} & \cdots & A_{1p}\mathbf{B} \\ \vdots & & \vdots \\ A_{m1}\mathbf{B} & \cdots & A_{mp}\mathbf{B} \end{pmatrix}. \quad (2.12)$$

Kronecker乘积有如下性质:

$$(1) (a\mathbf{A}) \otimes (b\mathbf{B}) = ab(\mathbf{A} \otimes \mathbf{B}).$$

$$(2) \mathbf{A} \otimes (\mathbf{B} + \mathbf{C}) = \mathbf{A} \otimes \mathbf{B} + \mathbf{A} \otimes \mathbf{C}, \quad (\mathbf{B} + \mathbf{C}) \otimes \mathbf{A} = \mathbf{B} \otimes \mathbf{A} + \mathbf{C} \otimes \mathbf{A}.$$

$$(3) \quad \mathbf{A} \otimes (\mathbf{B} \otimes \mathbf{C}) = (\mathbf{A} \otimes \mathbf{B}) \otimes \mathbf{C}.$$

$$(4) \quad (\mathbf{A} \otimes \mathbf{B})^T = \mathbf{A}^T \otimes \mathbf{B}^T.$$

$$(5) \quad (\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = (\mathbf{AC}) \otimes (\mathbf{BD}).$$

$$(6) \text{ 若 } \mathbf{A} \text{ 与 } \mathbf{B} \text{ 为非退化方阵, 则 } (\mathbf{A} \otimes \mathbf{B})^{-1} = \mathbf{A}^{-1} \otimes \mathbf{B}^{-1}.$$

$$(7) \quad \text{vec}(\mathbf{AXB}) = (\mathbf{B}^T \otimes \mathbf{A})\text{vec}(\mathbf{X}).$$

$$(8) \quad \text{tr}(\mathbf{A} \otimes \mathbf{B}) = \text{tr}(\mathbf{A})\text{tr}(\mathbf{B}).$$

$$(9) \text{ 若 } \mathbf{x} \text{ 与 } \mathbf{y} \text{ 为向量, 则 } \mathbf{xy}^T = \mathbf{x} \otimes \mathbf{y}^T = \mathbf{y}^T \otimes \mathbf{x}.$$

(10) 若  $\mathbf{A}$  为  $m \times m$  矩阵, 其特征值和特征向量为  $\{(\lambda_i, \alpha_i), i = 1, \dots, m\}$ ,  $\mathbf{B}$  为  $n \times n$  矩阵, 其特征值和特征向量为  $\{(\mu_i, \beta_i), i = 1, \dots, n\}$ , 则  $\mathbf{A} \otimes \mathbf{B}$  的特征值和特征向量为  $\{(\lambda_i \mu_j, \alpha_i \otimes \beta_j), i = 1, \dots, m, j = 1, \dots, n\}$ .

(11) 若  $\mathbf{A}$  为  $m \times m$  矩阵,  $\mathbf{B}$  为  $n \times n$  矩阵, 则  $|\mathbf{A} \otimes \mathbf{B}| = |\mathbf{A}|^m |\mathbf{B}|^n$ .

Proof: (5)  $\Rightarrow$  (6)、(5)  $\Rightarrow$  (10)、(10)  $\Rightarrow$  (11)。

(4):

$$(\mathbf{A} \otimes \mathbf{B})^T = \begin{pmatrix} A_{11}\mathbf{B} & A_{12}\mathbf{B} & \cdots & A_{1p}\mathbf{B} \\ A_{21}\mathbf{B} & A_{22}\mathbf{B} & \cdots & A_{2p}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m1}\mathbf{B} & A_{m2}\mathbf{B} & \cdots & A_{mp}\mathbf{B} \end{pmatrix}^T = \begin{pmatrix} A_{11}\mathbf{B}^T & A_{21}\mathbf{B}^T & \cdots & A_{m1}\mathbf{B}^T \\ A_{12}\mathbf{B}^T & A_{22}\mathbf{B}^T & \cdots & A_{m2}\mathbf{B}^T \\ \vdots & \vdots & \ddots & \vdots \\ A_{1p}\mathbf{B}^T & A_{2p}\mathbf{B}^T & \cdots & A_{mp}\mathbf{B}^T \end{pmatrix} = \mathbf{A}^T \otimes \mathbf{B}^T.$$

(5):

$$\begin{aligned}
 (\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) &= \begin{pmatrix} A_{11}\mathbf{B} & A_{12}\mathbf{B} & \cdots & A_{1p}\mathbf{B} \\ A_{21}\mathbf{B} & A_{22}\mathbf{B} & \cdots & A_{2p}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m1}\mathbf{B} & A_{m2}\mathbf{B} & \cdots & A_{mp}\mathbf{B} \end{pmatrix} \begin{pmatrix} C_{11}\mathbf{D} & C_{12}\mathbf{D} & \cdots & C_{1q}\mathbf{D} \\ C_{21}\mathbf{D} & C_{22}\mathbf{D} & \cdots & C_{2q}\mathbf{D} \\ \vdots & \vdots & \ddots & \vdots \\ C_{p1}\mathbf{D} & C_{p2}\mathbf{D} & \cdots & C_{pq}\mathbf{D} \end{pmatrix} \\
 &= \begin{pmatrix} \left(\sum_k A_{1k}C_{k1}\right)\mathbf{BD} & \left(\sum_k A_{1k}C_{k2}\right)\mathbf{BD} & \cdots & \left(\sum_k A_{1k}C_{kq}\right)\mathbf{BD} \\ \left(\sum_k A_{2k}C_{k1}\right)\mathbf{BD} & \left(\sum_k A_{2k}C_{k2}\right)\mathbf{BD} & \cdots & \left(\sum_k A_{2k}C_{kq}\right)\mathbf{BD} \\ \vdots & \vdots & \ddots & \vdots \\ \left(\sum_k A_{mk}C_{k1}\right)\mathbf{BD} & \left(\sum_k A_{mk}C_{k2}\right)\mathbf{BD} & \cdots & \left(\sum_k A_{mk}C_{kq}\right)\mathbf{BD} \end{pmatrix} \\
 &= \begin{pmatrix} (\mathbf{AC})_{11}\mathbf{BD} & (\mathbf{AC})_{12}\mathbf{BD} & \cdots & (\mathbf{AC})_{1q}\mathbf{BD} \\ (\mathbf{AC})_{21}\mathbf{BD} & (\mathbf{AC})_{22}\mathbf{BD} & \cdots & (\mathbf{AC})_{2q}\mathbf{BD} \\ \vdots & \vdots & \ddots & \vdots \\ (\mathbf{AC})_{m1}\mathbf{BD} & (\mathbf{AC})_{m2}\mathbf{BD} & \cdots & (\mathbf{AC})_{mq}\mathbf{BD} \end{pmatrix} = (\mathbf{AC}) \otimes (\mathbf{BD}).
 \end{aligned}$$

(8):

$$\text{tr}(\mathbf{a} \otimes \mathbf{B}) = \text{tr} \begin{pmatrix} A_{11}\mathbf{B} & A_{12}\mathbf{B} & \cdots & A_{1p}\mathbf{B} \\ A_{21}\mathbf{B} & A_{22}\mathbf{B} & \cdots & A_{2p}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m1}\mathbf{B} & A_{m2}\mathbf{B} & \cdots & A_{mp}\mathbf{B} \end{pmatrix} = \sum_{i=1}^n \text{tr}(A_{ii}\mathbf{B}) = \sum_{i=1}^n A_{ii} \text{tr}(\mathbf{B}) = \text{tr}(\mathbf{A}) \text{tr}(\mathbf{B}).$$

(10): 直接按特征值/特征向量定义验证。

**练习5.** 证明(7)。

## 2.8 Sherman-Morrison-Woodbury Formula

**命题6.**

$$(\mathbf{A} + \mathbf{U}\mathbf{C}\mathbf{V}^T)^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{C}^{-1} + \mathbf{V}^T\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}^T\mathbf{A}^{-1},$$

where  $\mathbf{A}$  and  $\mathbf{C}$  are invertible and the sizes of  $\mathbf{U}$ ,  $\mathbf{C}$ , and  $\mathbf{V}$  are compatible.

**推论7.**

$$(\mathbf{A} + \mathbf{u}\mathbf{v}^T)^{-1} = \mathbf{A}^{-1} - \frac{(\mathbf{A}^{-1}\mathbf{u})(\mathbf{v}^T\mathbf{A}^{-1})}{1 + \mathbf{v}^T\mathbf{A}^{-1}\mathbf{u}}.$$

It is useful for online update.

**练习8.** 证明Sherman-Morrison-Woodbury公式。

## 2.9 向量与矩阵的范数

**定义9.** A function  $\|\cdot\|$  on  $\mathbb{R}^n$  is called a norm if it satisfies the following conditions:

1.  $\|\mathbf{x}\| \geq 0, \forall \mathbf{x} \in \mathbb{R}^n$ , and  $\|\mathbf{x}\| = 0$  iff  $\mathbf{x} = 0$ ;
2.  $\|\alpha \mathbf{x}\| = |\alpha| \|\mathbf{x}\|, \forall \alpha \in \mathbb{R}$  and  $\mathbf{x} \in \mathbb{R}^n$ ;
3.  $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|, \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ .

Common vector norms include:

1.  $\|\mathbf{x}\|_1 = \sum_i |x_i|$ ;
2.  $\|\mathbf{x}\|_\infty = \max_i |x_i|$ ;
3.  $\|\mathbf{x}\|_2 = \sqrt{\sum_i x_i^2}$ ;
4.  $\|\mathbf{x}\|_p = \left( \sum_i |x_i|^p \right)^{1/p}, p \geq 1$ .

**定义10.** A matrix norm  $\|\cdot\|$  is a vector norm that further satisfies

$$\|\mathbf{AB}\| \leq \|\mathbf{A}\| \|\mathbf{B}\|.$$

**定义11.** Suppose  $\|\cdot\|$  is a vector norm. The matrix norm  $\|\cdot\|$  induced by the vector norm is defined as

$$\|\mathbf{A}\| = \sup_{\|\mathbf{x}\|=1} \|\mathbf{Ax}\|.$$

Common vector norm induced matrix norms include:

1.  $\|\mathbf{A}\|_1 = \max_j \sum_i |A_{ij}|$ , induced by vector 1-norm;
2.  $\|\mathbf{A}\|_\infty = \max_i \sum_j |A_{ij}|$ , induced by vector  $\infty$ -norm;
3.  $\|\mathbf{A}\|_2 = \sigma_1(\mathbf{A})$ , where  $\sigma_1(\mathbf{A})$  denotes the largest singular value of  $\mathbf{A}$ . It is induced by vector 2-norm.

**备注12.** Not all matrix norms are induced norms. According to Theorem 5.6.32 of Matrix Analysis by Horn & Johnson (2nd Ed.), a matrix norm  $\|\cdot\|$  is induced iff it is a minimal matrix norm, i.e., the only matrix norm  $N(\cdot)$  that satisfies  $N(\mathbf{X}) \leq \|\mathbf{X}\|, \forall \mathbf{X}$ , is  $N(\cdot) = \|\cdot\|$ . By this theorem, the Frobenius norm cannot be induced.

**定理13.** Any norm for matrices can be scaled appropriately so that it becomes a matrix norm.

Define

$$a = \sup_{\|\mathbf{A}\|=1, \|\mathbf{B}\|=1} \|\mathbf{AB}\|.$$

Then  $\|\mathbf{X}\|_m = a\|\mathbf{X}\|$  is a matrix norm:

$$\begin{aligned} \|\mathbf{AB}\|_m &= a\|(\mathbf{A}/\|\mathbf{A}\|)(\mathbf{B}/\|\mathbf{B}\|)\| \|\mathbf{A}\| \|\mathbf{B}\| \\ &= a^{-1}\|(\mathbf{A}/\|\mathbf{A}\|)(\mathbf{B}/\|\mathbf{B}\|)\| \|\mathbf{A}\|_m \|\mathbf{B}\|_m \\ &\leq \|\mathbf{A}\|_m \|\mathbf{B}\|_m. \end{aligned}$$

### 2.9.1 Nuclear Norm

**定义14.** The nuclear norm  $\|\mathbf{A}\|_*$  of a matrix  $\mathbf{A}$  is defined as the sum of all the singular values of  $\mathbf{A}$ .

It is usually used for approximating the rank of a matrix, because:

**定理15.** The nuclear norm is the convex envelope of the rank function on the unit ball  $B_1 = \{\mathbf{A} \mid \|\mathbf{A}\|_2 \leq 1\}$  of matrix 2-norm.

**定义16.** Convex envelope is the largest convex function upper bounded by the give function

**思考题17.** Is nuclear norm  $\|\cdot\|_*$  a matrix norm?

**命题18.** The matrix norms have the following relationship:

$$1. \quad \|\mathbf{A}\|_2 \leq \|\mathbf{A}\|_F \leq \|\mathbf{A}\|_*. \tag{2.13}$$

$$2. \quad \|\mathbf{A}\|_2 \leq \sqrt{\|\mathbf{A}\|_1 \|\mathbf{A}\|_\infty}. \tag{2.14}$$

### 2.9.2 $(p, q)$ -Norm

The  $(p, q)$ -norm of matrices are also widely used in sparse representation.

**定义19.**

$$\|\mathbf{A}\|_{p,q} = \left( \sum_{i=1}^n \|\mathbf{A}_i\|_p^q \right)^{\frac{1}{q}},$$

where  $\mathbf{A} = (\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_n)$ .

Note that here we use an intuitive notation for the above norm. It is different from the induced norm of linear mapping:  $\mathbb{R}^p \rightarrow \mathbb{R}^q$ .

When  $q = 1$ , minimizing  $\|\mathbf{A}\|_{p,q}$  will encourage the columns of  $\mathbf{A}$  to be all zeros, which is called the group sparsity.

**思考题20.** Is  $(p, q)$ -Norm  $\|\cdot\|_{p,q}$  a matrix norm?

**练习21.** Let  $\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{pmatrix}$ , compute its 1-norm, 2-norm, Frobenious norm,  $\infty$ -norm, nuclear norm, and  $(2, 1)$ -norm.

## 2.10 矩阵的导数

如果  $y = y(\mathbf{X})$  是一个标量, 而  $\mathbf{X}$  是向量或矩阵, 则定义  $y$  对  $\mathbf{X}$  的导数为

$$\frac{\partial y}{\partial \mathbf{X}} \triangleq \left( \frac{\partial y}{\partial X_{ij}} \right). \quad (2.15)$$

使用上面的定义, 可以很方便地把 Taylor 展开式写成:

$$\begin{aligned} y(\mathbf{X}) &= y(\mathbf{X}_0) + \sum_{ij} \frac{\partial y}{\partial X_{ij}} \Big|_{\mathbf{X}=\mathbf{X}_0} \cdot (X_{ij} - X_{0,ij}) + \dots \\ &= y(\mathbf{X}_0) + \langle \frac{\partial y}{\partial \mathbf{X}}, \mathbf{X} - \mathbf{X}_0 \rangle + \dots \end{aligned} \quad (2.16)$$

**例22.** 对  $y(\mathbf{X}) = \|\mathbf{X}\|_F^2$  进行求导。

如果  $\mathbf{Y} = (Y_{ij}(x))$  是一个矩阵, 而  $x$  是标量, 则定义  $\mathbf{Y}$  对  $x$  的导数为

$$\frac{\partial \mathbf{Y}}{\partial x} \triangleq \left( \frac{\partial Y_{ij}}{\partial x} \right). \quad (2.17)$$

$\mathbf{Y}$  其实就是把一堆函数放在一起, 所以它对  $x$  的求导是各函数自己进行。

**例23.** 对  $Y(x) = \begin{pmatrix} \sin(x) & \cos(x) \\ -\cos(x) & \sin(x) \end{pmatrix}$  进行求导。

如果  $\mathbf{Y} = (Y_{ij}(\mathbf{X}))$  是一个矩阵, 而  $\mathbf{X}$  是向量或矩阵, 则定义  $\mathbf{Y}$  对  $\mathbf{X}$  的导数为

$$\frac{\partial \mathbf{Y}}{\partial \mathbf{X}} = \left( \frac{\partial \mathbf{Y}}{\partial X_{ij}} \right) \triangleq \left( \frac{\partial}{\partial X_{ij}} \right) \otimes \mathbf{Y}. \quad (2.18)$$

矩阵对矩阵的求导主要是为了函数对矩阵求导所涉及的链式法则用的。

**例24.** 对  $\mathbf{Y}(\mathbf{X}) = \mathbf{X}\mathbf{X}^T$  进行求导。

矩阵的导数有如下性质。

(1) 当自变量为标量 $t$ 时:

$$\frac{\partial(\mathbf{X} + \mathbf{Y})}{\partial t} = \frac{\partial \mathbf{X}}{\partial t} + \frac{\partial \mathbf{Y}}{\partial t}, \quad (2.19)$$

$$\frac{\partial(\mathbf{XY})}{\partial t} = \frac{\partial \mathbf{X}}{\partial t} \mathbf{Y} + \mathbf{X} \frac{\partial \mathbf{Y}}{\partial t}, \quad (2.20)$$

$$\frac{\partial(\mathbf{X} \otimes \mathbf{Y})}{\partial t} = \frac{\partial \mathbf{X}}{\partial t} \otimes \mathbf{Y} + \mathbf{X} \otimes \frac{\partial \mathbf{Y}}{\partial t}, \quad (2.21)$$

$$\frac{\partial \mathbf{X}^T}{\partial t} = \left( \frac{\partial \mathbf{X}}{\partial t} \right)^T, \quad (2.22)$$

$$\frac{\partial(\mathbf{AXB})}{\partial X_{ij}} = \mathbf{A}\mathbf{E}_{ij}\mathbf{B}, \quad (2.23)$$

$$\frac{\partial \mathbf{X}^{-1}}{\partial t} = -\mathbf{X}^{-1} \frac{\partial \mathbf{X}}{\partial t} \mathbf{X}^{-1}, \quad (2.24)$$

其中 $\mathbf{E}_{ij}$ 是 $(i, j)$ 位置为1、其他位置全为0的矩阵。

(2) 当自变量为向量 $\mathbf{x}$ 时:

$$\frac{\partial(\mathbf{a}^T \mathbf{x})}{\partial \mathbf{x}} = \mathbf{a}, \quad (2.25)$$

$$\frac{\partial(\mathbf{x}^T \mathbf{Ax})}{\partial \mathbf{x}} = (\mathbf{A} + \mathbf{A}^T)\mathbf{x}, \quad (2.26)$$

$$\frac{\partial(\mathbf{Ax})}{\partial \mathbf{x}} = \text{vec}(\mathbf{A}^T), \quad (2.27)$$

$$\frac{\partial(\mathbf{Ax})^T}{\partial \mathbf{x}} = \mathbf{A}^T. \quad (2.28)$$

(3) 当自变量为 $m \times n$ 矩阵 $\mathbf{X}$ 时:

$$\frac{\partial(\mathbf{F}(\mathbf{X}) + \mathbf{G}(\mathbf{X}))}{\partial \mathbf{X}} = \frac{\partial \mathbf{F}(\mathbf{X})}{\partial \mathbf{X}} + \frac{\partial \mathbf{G}(\mathbf{X})}{\partial \mathbf{X}}, \quad (2.29)$$

$$\frac{\partial(\mathbf{F}(\mathbf{X})\mathbf{G}(\mathbf{X}))}{\partial \mathbf{X}} = \frac{\partial \mathbf{F}(\mathbf{X})}{\partial \mathbf{X}} (\mathbf{I}_n \otimes \mathbf{G}(\mathbf{X})) + (\mathbf{I}_m \otimes \mathbf{F}(\mathbf{X})) \frac{\partial \mathbf{G}(\mathbf{X})}{\partial \mathbf{X}}, \quad (2.30)$$

$$\frac{\partial(f(\mathbf{X})\mathbf{Y})}{\partial \mathbf{X}} = \frac{\partial f(\mathbf{X})}{\partial \mathbf{X}} \mathbf{Y} + f(\mathbf{X}) \frac{\partial \mathbf{Y}}{\partial \mathbf{X}}, \quad (2.31)$$

$$\frac{\partial(\mathbf{A} \otimes \mathbf{B})}{\partial \mathbf{X}} = \frac{\partial \mathbf{A}}{\partial \mathbf{X}} \otimes \mathbf{B} + \left( \mathbf{A} \otimes \frac{\partial \mathbf{B}}{\partial X_{ij}} \right), \quad (2.32)$$

$$\frac{\partial \mathbf{A}^{-1}}{\partial \mathbf{X}} = -(\mathbf{I}_m \otimes \mathbf{A}^{-1}) \frac{\partial \mathbf{A}}{\partial \mathbf{X}} (\mathbf{I}_n \otimes \mathbf{A}^{-1}), \quad (2.33)$$

$$\frac{\partial(\text{vec}(\mathbf{AXB})^T)}{\partial \text{vec}(\mathbf{X})} = \mathbf{B} \otimes \mathbf{A}^T, \quad (2.34)$$

$$\frac{\partial |\mathbf{X}|}{\partial \mathbf{X}} = |\mathbf{X}|(\mathbf{X}^{-1})^T, \quad (2.35)$$

$$\frac{\partial \ln |\mathbf{X}|}{\partial \mathbf{X}} = (\mathbf{X}^T)^{-1}. \quad (2.36)$$

**练习25.** Prove (2.20)、(2.25)、(2.26)、求 $\frac{\partial \mathbf{X}}{\partial \mathbf{X}}$ 并写成简洁形式.

## 2.11 伴随算子 (Adjoint Operator)

给定线性算子  $\mathcal{A} : \mathbb{R}^m \rightarrow \mathbb{R}^n$ , 其伴随算子 (Adjoint Operator) 定义为满足以下条件的线性算子:

$$\langle \mathcal{A}^*(\mathbf{x}), \mathbf{y} \rangle = \langle \mathbf{x}, \mathcal{A}(\mathbf{y}) \rangle, \quad \forall \mathbf{x} \in \mathbb{R}^n, \mathbf{y} \in \mathbb{R}^m. \quad (2.37)$$

如果  $\mathcal{A}(\mathbf{x}) = \mathbf{Ax}$ , 则  $\mathcal{A}^*(\mathbf{x}) = \mathbf{A}^T \mathbf{x}$ . 如果  $\mathcal{A}(\mathbf{X}) = \mathbf{x}$  为抽取矩阵  $\mathbf{X}$  的若干元素的线性算子, 则  $\mathcal{A}^*(\mathbf{x})$  为把  $\mathbf{x}$  的元素放在  $\mathbf{X}$  的相应位置、其余位置填0的线性算子。特别地, 设  $\text{diag}(\mathbf{X})$  为抽取对角线元素的算子, 则  $\text{diag}^*(\mathbf{x})$  为以  $\mathbf{x}$  为对角线元素的对角矩阵。

**练习26.** 假设  $\mathbf{X} \in \mathbb{R}^{3 \times 3}$ ,  $\mathcal{A}(\mathbf{X}) = X_{11} + X_{12} - X_{31} + 2X_{33}$ , 求  $\mathcal{A}^*$ .

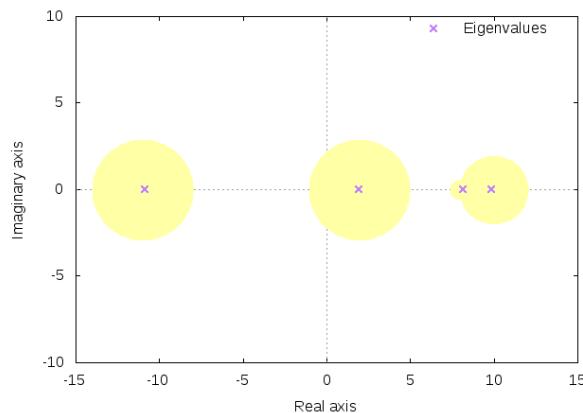


图 2.1: Gershgorin圆盘定理所指示的特征值所在的四个圆盘。前两个圆盘有重叠, 它们的并包含了两个特征值。后两个与其他圆盘不相交, 它们各自有一个特征值。

## 2.12 Gershgorin圆盘定理

(取自[http://en.wikipedia.org/wiki/Gershgorin\\_circle\\_theorem](http://en.wikipedia.org/wiki/Gershgorin_circle_theorem))

Gershgorin圆盘定理经常被用于估计方阵的特征值的范围。它是苏联数学家Semyon Aranovich Gershgorin于1931年发表的。在稀疏表示理论中, Gershgorin圆盘定理被经常使用。

设  $\mathbf{A}$  是一个  $n \times n$  复矩阵, 其元素为  $a_{ij}$ . 对  $i \in \{1, \dots, n\}$ , 定义  $R_i = \sum_{j \neq i} |a_{ij}|$  为第  $i$  行非对角元元素的绝对值的和,  $D(a_{ii}, R_i)$  为以  $a_{ii}$  为中心、半径为  $R_i$  的闭圆, 称为 Gershgorin 圆盘。

**定理27.**  $\mathbf{A}$  的每一个特征值在至少一个 Gershgorin 圆盘  $D(a_{ii}, R_i)$  中。

*Proof.* 设 $\lambda$ 为 $\mathbf{A}$ 的一个特征值,  $\mathbf{x} = (x_j)$ 为对应的特征向量。选 $i \in \{1, \dots, n\}$ 为满足 $|x_i| = \max_j |x_j|$ 的下标, 即*i*使 $x_i$ 绝对值最大。则 $|x_i| > 0$ , 否则 $\mathbf{x} = \mathbf{0}$ , 与特征向量定义不符。因为 $\mathbf{x}$ 是特征向量,  $\mathbf{Ax} = \lambda\mathbf{x}$ , 于是:

$$\sum_j a_{ij}x_j = \lambda x_i, \quad \forall i \in \{1, \dots, n\}.$$

把求和分开, 我们得到 $\sum_{j \neq i} a_{ij}x_j = \lambda x_i - a_{ii}x_i$ . 然后把两边同除非0的 $x_i$ 再取绝对值, 得

$$|\lambda - a_{ii}| = \left| \frac{\sum_{j \neq i} a_{ij}x_j}{x_i} \right| \leq \sum_{j \neq i} \left| \frac{a_{ij}x_j}{x_i} \right| \leq \sum_{j \neq i} |a_{ij}| = R_i,$$

其中最后一个不等式是因为

$$\left| \frac{x_j}{x_i} \right| \leq 1 \quad \text{for } j \neq i.$$

□

解读这个定理的方式之一是: 如果一个方阵的非对角元的绝对值之和很小, 则其特征值不能偏离其对角元太远。因此, 通过降低非对角元的绝对值之和, 我们可以很好地估计矩阵的特征值。当然, 在降低非对角元的过程中, 对角元也在变化。

**例28.** 利用Gershgorin圆盘定理估计下列矩阵的特征值:

$$\mathbf{A} = \begin{bmatrix} 10 & -1 & 0 & 1 \\ 0.2 & 8 & 0.2 & 0.2 \\ 1 & 1 & 2 & 1 \\ -1 & -1 & -1 & -11 \end{bmatrix}.$$

解: 从第一行开始, 我们逐一选 $a_{ii}$ 为圆盘的中心, 计算非对角元的绝对值之和

$$\sum_{j \neq i} |a_{ij}| = R_i$$

为其半径, 得到四个圆盘:

$$D(10, 2), D(8, 0.6), D(2, 3), \text{ 和 } D(-11, 3).$$

事实上我们可以改进后两个圆盘的半径, 将该定理应用于矩阵的列, 得到 $D(2, 1.2)$ 和 $D(-11, 2.2)$ .

特征值的精确值为: 9.8218, 8.1478, 1.8995, -10.86.

### 2.13 von Neumann迹定理

设  $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{m \times n}$ , 则

$$|\langle \mathbf{A}, \mathbf{B} \rangle| \leq \sum_{i=1}^{\min(m,n)} \sigma_i(\mathbf{A})\sigma_i(\mathbf{B}). \quad (2.38)$$

In particular, when both  $\mathbf{A}$  and  $\mathbf{B}$  are  $n \times n$  p.s.d. matrices, the inequality becomes

$$\sum_{i=1}^n \sigma_i(\mathbf{A})\sigma_{n-i+1}(\mathbf{B}) \leq \text{tr}(\mathbf{AB}) \leq \sum_{i=1}^n \sigma_i(\mathbf{A})\sigma_i(\mathbf{B}). \quad (2.39)$$

**练习29.** Use von Neumann inequality to prove that the solution to

$$\min_{\mathbf{Y}} \text{tr}(\mathbf{YK} \mathbf{Y}^T), \quad s.t. \quad \mathbf{YY}^T = \mathbf{I},$$

is the tailing eigenvectors of  $\mathbf{K}$ . The solution to

$$\max_{\mathbf{Y}} \text{tr}(\mathbf{YK} \mathbf{Y}^T), \quad s.t. \quad \mathbf{YY}^T = \mathbf{I},$$

is the leading eigenvectors of  $\mathbf{K}$ . Try whether the above can be deduced by using Lagrange multiplier.

更多的实用矩阵知识可查询《The Matrix Cookbook》(<http://orion.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf>) 或Matrix Differential Calculus with Applications in Statistics and Econometrics



## 第三章 数据几何

### 3.1 欧氏空间中的可微流形

因一般抽象流形的介绍需要大量的数学准备知识，而且流形学习所涉及数据都是欧氏空间里的，所以为直观计，仅介绍欧氏空间中的可微流形。但是欧氏空间中的可微流形既有特殊性也有代表性，因为根据Whitney 嵌入定理，任何 $k$ 维流形都可以嵌入到 $2k + 1$ 维欧氏空间中。（展示螺旋线，在二维空间中自交、三维空间中不自交）

#### 3.1.1 坐标卡与参数化[133]

**定义30.** 设一个多元函数  $f : U \subset \mathbb{R}^p \rightarrow V \subset \mathbb{R}^q$ 。如果  $f$  为双射，且  $f$  和  $f^{-1}$  都连续，则称  $f$  为同胚，且称  $U$  和  $V$  是同胚的（此时必然  $p = q$ ）。若  $f$  为同胚，且  $f$  和  $f^{-1}$  都是  $C^s$  映射，则称  $f$  为  $C^s$  微分同胚，且称  $U$  和  $V$  是  $C^s$  微分同胚的。如果  $f$  为同胚，且  $f$  和  $f^{-1}$  都是  $C^\infty$  映射，则称  $f$  为光滑同胚，且称  $U$  和  $V$  是光滑同胚的。

**定义31.** 设  $\mathcal{M} \subset \mathbb{R}^m$ 。若对每一点  $p \in \mathcal{M}$ ，都有  $p$  的一个邻域  $U$  与  $\mathbb{R}^k$  的一个开子集同胚，则称  $\mathcal{M}$  为  $k$  维拓扑流形。

即  $\mathcal{M}$  是由  $\mathbb{R}^k$  的一系列开子集“拼贴”而成的。

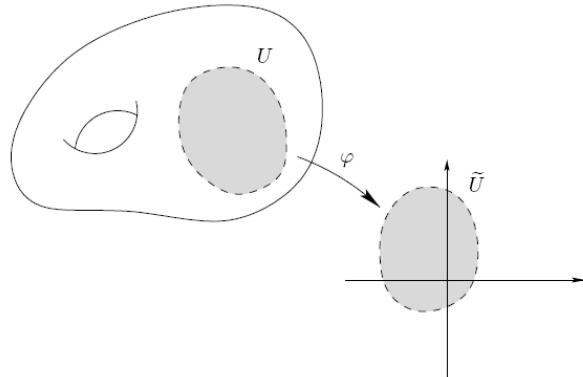


图 3.1: 拓扑流形

**定义32.** 设  $\mathcal{M} \subset \mathbb{R}^m$  非空。如果对  $\mathcal{M}$  上每一点  $x$ ，存在一个开集  $W \subset \mathcal{M}$  使得  $W$  与  $\mathbb{R}^k$  的开子集  $U$   $C^s$  微分同胚（光滑同胚），则  $\mathcal{M}$  称为  $k$  维  $C^s$  可微流形（光滑流形），简称  $k$ -流形。同胚  $g : U \rightarrow W$  称为  $W$  的参数化，其逆  $h = g^{-1} : W \rightarrow U$  称为坐标映射， $W$  称为坐标邻域， $(W, h)$  称为  $M$  的一个（局部）坐标卡。

在坐标卡 $(W, h)$ 里，一个点 $x \in W$ 具有如下坐标

$$h(x) = [h^1(x), h^2(x), \dots, h^k(x)]^T,$$

其中 $h^i$ 称为第*i*个坐标函数。如果有一个单一的坐标卡 $(\mathcal{M}, h)$ ，则称 $\mathcal{M}$ 为简单流形。

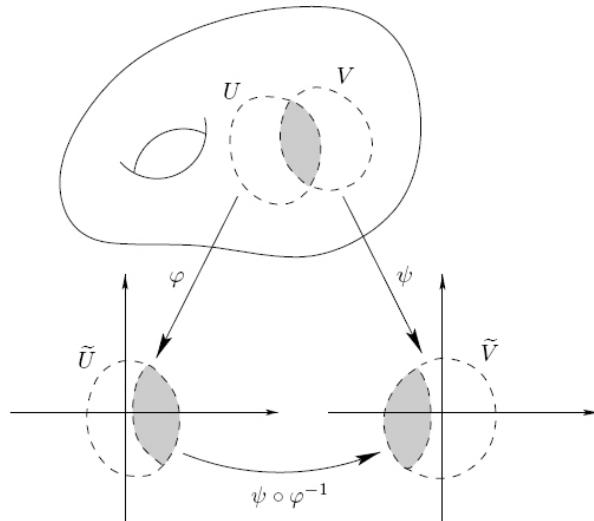


图 3.2: 可微流形

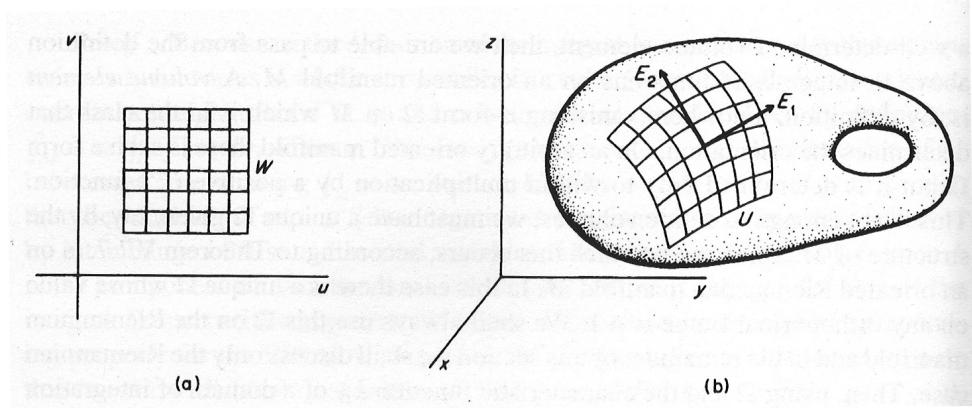


图 3.3: 坐标卡与坐标函数

流形的例子：

1.  $\mathbb{R}^m$  是  $m$  维简单光滑流形。
2.  $\mathcal{M}_{m \times k} = \{M \in \mathbb{R}^{m \times k}\}$  是  $mk$  维简单光滑流形。
3.  $\mathbb{R}^{m+1}$  中的单位球面  $S^m$  是  $m$  维光滑流形。

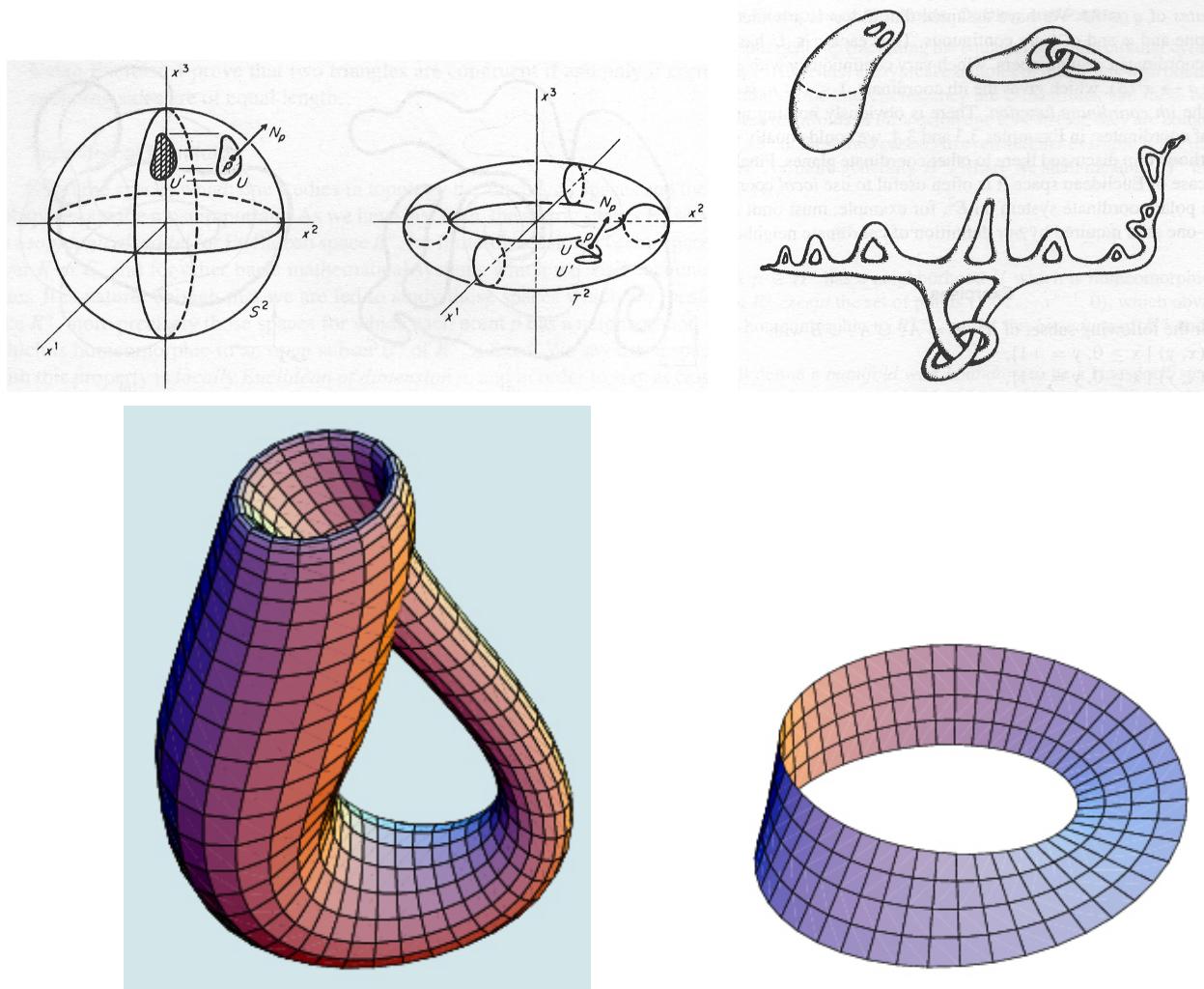


图 3.4: 光滑流形的例子

流形  $\mathcal{M}$  的开子集  $U$  是一个流形，称为开子流形，它的坐标卡是  $\mathcal{M}$  的坐标卡在  $U$  上的限制，它的维数和  $\mathcal{M}$  相同。由此推出一般线性群  $GL(n, \mathbb{R}) = \{M | M \in \mathbb{R}^{n \times n}, \det(M) \neq 0\}$  是  $\mathcal{M}_{n \times n}$  的开子流形。

### 3.1.2 切空间与切向量[133]

**定义33.** 设  $f : U \subset \mathbb{R}^k \rightarrow V \subset \mathbb{R}^l$ , 对任一点  $x \in U$ , 定义  $f$  的导数  $df_x : \mathbb{R}^k \rightarrow \mathbb{R}^l$  为

$$df_x(y) = \lim_{t \rightarrow 0} \frac{f(x + ty) - f(x)}{t}, \quad y \in \mathbb{R}^k. \quad (3.1)$$

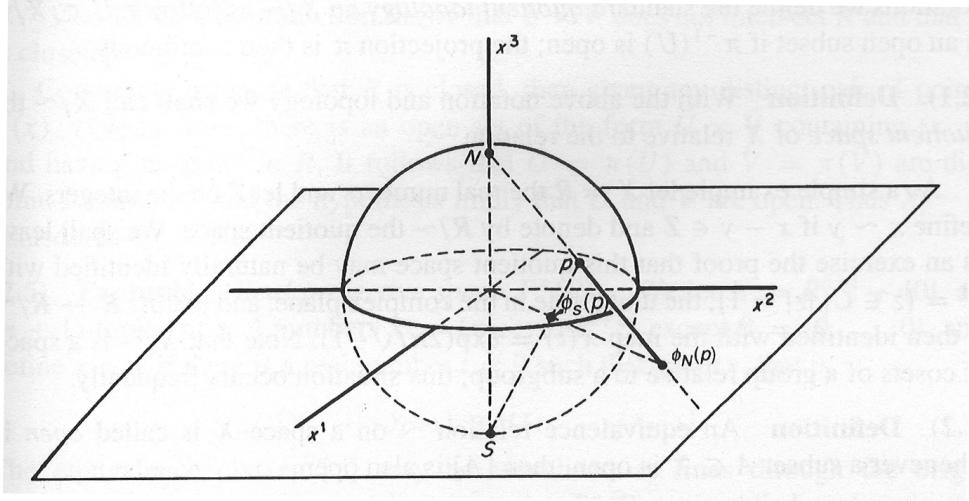


图 3.5: 球极投影

$\mathrm{d}f_x$  是一个线性变换，其矩阵表达为

$$\mathrm{d}f_x = \left[ \frac{\partial f_i(x)}{\partial x_j} \right]_{i,j=1}^{l,k} = \begin{pmatrix} \frac{\partial f_1(x)}{\partial x_1} & \dots & \frac{\partial f_1(x)}{\partial x_k} \\ \frac{\partial f_2(x)}{\partial x_1} & \dots & \frac{\partial f_2(x)}{\partial x_k} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_l(x)}{\partial x_1} & \dots & \frac{\partial f_l(x)}{\partial x_k} \end{pmatrix}. \quad (3.2)$$

特别地，当  $f$  是函数 ( $l = 1$ ) 时， $f$  的导数就是  $f$  的梯度：

$$\mathrm{d}f_x = \mathrm{grad}f_x.$$

**定义34.** 设  $\mathcal{M} \subset \mathbb{R}^m$  是  $k$ -流形， $U \subset \mathbb{R}^k$  为开集， $g : U \rightarrow \mathcal{M}$  为局部邻域  $g(U) \subset \mathcal{M}$  的参数化， $p \in \mathcal{M}$ ， $u \in U$ ， $g(u) = p$ 。则  $\mathcal{M}$  在  $p$  点的切空间定义为  $T_p\mathcal{M} \triangleq dg_u(\mathbb{R}^k)$ ，即线性变换  $dg_u$  的像空间。 $T_p\mathcal{M}$  的元素称为切向量。

当不需要强调  $\mathcal{M}$  时， $T_p\mathcal{M}$  简记为  $T_p$ 。 $T_p\mathcal{M}$  是  $k$  维线性子空间。直观上，我们经常把在  $p$  点的仿射子空间  $H_p \triangleq p + T_p$  也称为  $\mathcal{M}$  在  $p$  点的切空间，因为  $H_p$  和  $\mathcal{M}$  刚好在  $p$  点相切。

记

$$\frac{\partial g}{\partial u^i} = \left[ \frac{\partial g_1}{\partial u^i}, \dots, \frac{\partial g_m}{\partial u^i} \right]^T, \quad 1 \leq i \leq k.$$

则

$$\left\{ \frac{\partial g}{\partial u^1}, \dots, \frac{\partial g}{\partial u^k} \right\} \quad (3.3)$$

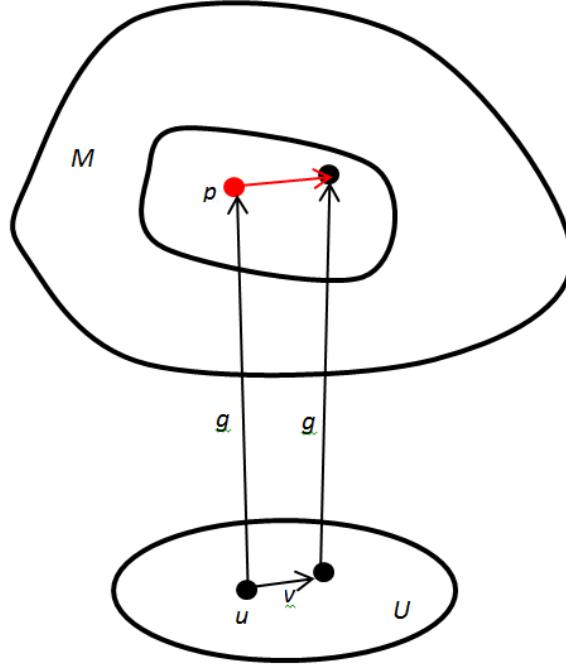


图 3.6: 切空间与切向量

是  $T_p\mathcal{M}$  的一组基。定义基矩阵为

$$\frac{\partial g}{\partial u} = \left[ \frac{\partial g_i}{\partial u_j} \right]_{i,j=1}^{m,k} = \begin{pmatrix} \frac{\partial g_1(x)}{\partial u_1} & \dots & \frac{\partial g_1(x)}{\partial u_k} \\ \frac{\partial g_2(x)}{\partial u_1} & \dots & \frac{\partial g_2(x)}{\partial u_k} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_m(x)}{\partial u_1} & \dots & \frac{\partial g_m(x)}{\partial u_k} \end{pmatrix}, \quad (3.4)$$

则  $p$  点的切向量  $X_p$  都可以表达成

$$X_p = \frac{\partial g}{\partial u} a.$$

### 3.1.3 黎曼度量与黎曼流形[133]

**定义35.** 设  $\mathcal{M}$  为  $k$ -流形,  $T_p$  为  $p \in \mathcal{M}$  的切空间。假定  $g$  是  $\mathcal{M}$  的参数化,  $(W, h)$  是  $\mathcal{M}$  的坐标卡, 其中  $h = g^{-1}$ 。对每一点  $p \in \mathcal{M}$ , 令  $u = h(p) \in \mathbb{R}^k$ , 则  $g$  定义了  $T_p\mathcal{M}$  的一组基(3.3), 表达成(3.4)。则  $\mathcal{M}$  在  $p$  处的黎曼度量定义为

$$G_g(p) = \left( \frac{\partial g}{\partial u} \right)^T \frac{\partial g}{\partial u}. \quad (3.5)$$

$G_g(p)$  是一个正定矩阵, 有时简记为  $G(p)$  或  $G$ 。

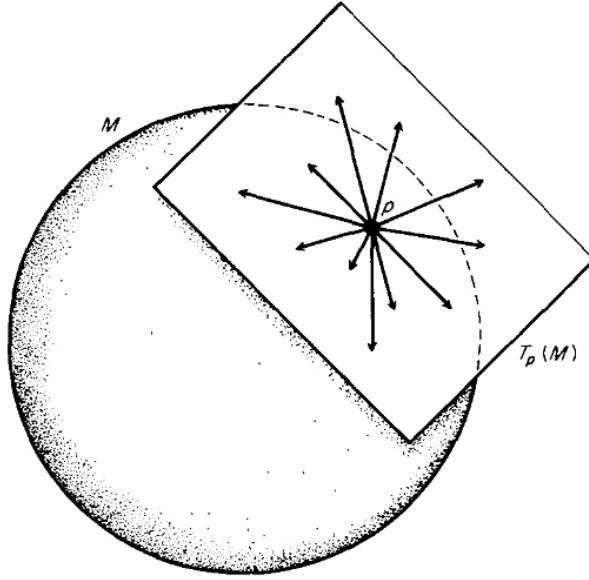


图 3.7: 切空间与切向量

上面定义的黎曼度量实际上是外围 (ambient) 欧氏空间的欧氏距离在流形上诱导出来的度量: 若  $X, Y \in T_p\mathcal{M}$ , 则欧氏空间的内积为  $\langle X, Y \rangle_p$ , 如果  $X, Y$  在切空间中的表示分别为  $X = \frac{\partial g}{\partial u}x$  和  $Y = \frac{\partial g}{\partial u}y$ , 则

$$\langle X, Y \rangle_p = x^T G(p) y. \quad (3.6)$$

因此  $G(p)$  可以看成是  $\mathcal{M}$  在  $p$  点的度量。

**定义36.** 设  $\mathcal{M}$  为  $k$ -流形, 如果  $\mathcal{M}$  在每一点  $p$  被赋予度量  $G(p)$ , 且对  $T_p\mathcal{M}$  的每一对切向量  $X, Y$ , 其内积定义为 (3.6), 则  $\mathcal{M}$  称为黎曼流形, 记为  $(\mathcal{M}, g)$  或  $(\mathcal{M}, G)$ 。

内积有如下性质:

1. 线性性:  $\langle aX + bY, z \rangle_p = a\langle X, z \rangle_p + b\langle Y, z \rangle_p$ ;
2. 对称性:  $\langle X, Y \rangle_p = \langle Y, X \rangle_p$ ;
3. 正定性:  $\langle X, X \rangle_p \geq 0$ , 且等号成立当且仅当  $X = 0$ 。

于是可以定义切向量的模为:

$$\|X\|_p = \sqrt{\langle X, X \rangle_p}. \quad (3.7)$$

定义切向量间的夹角为:

$$\theta = \frac{\langle X, Y \rangle}{\|X\|\|Y\|}, \quad \theta \in [0, \pi]. \quad (3.8)$$

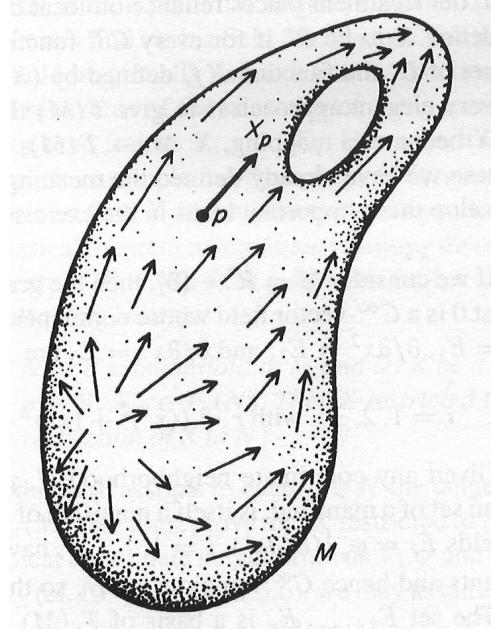


图 3.8: 切向量场

设  $C : [a, b] \rightarrow \mathcal{M}$ ,  $t \mapsto C(t)$ , 为  $\mathcal{M}$  上的曲线, 则积分

$$l(C) = \int_a^b \left\| \frac{dC}{dt} \right\| dt \quad (3.9)$$

与参数的选取无关, 它称为  $C$  的弧长。

如果在定义 35 中,  $G_p$  不是正定的, 但是对称、非退化的, 则称  $(\mathcal{M}, g)$  为广义 Riemann 流形或伪 Riemann 流形。这类流形的一个著名例子是 Minkowski 空间, 它具有 Lorentz 度量  $ds^2 = (dx^1)^2 - (dx^2)^2 - \cdots - (dx^m)^2$ 。它是狭义相对论的几何模型。本人最早做了基于半(伪)黎曼流形的流形学习, 参见本人 CVPR2008、ACCV2009、PR2010、TKDE2011 论文。

### 3.1.4 协变微分、平行移动与测地线[16]

设  $Y$  是流形  $\mathcal{M} \subset \mathbb{R}^m$  上的向量场,  $p(t)$  是  $\mathcal{M}$  上的一条曲线, 则

$$\frac{dY}{dt} = \lim_{\varepsilon \rightarrow 0} \frac{Y(p(t + \varepsilon)) - Y(p(t))}{\varepsilon}$$

是  $\mathbb{R}^m$  中的向量场。

**定义 37.**  $Y$  关于曲线  $p(t)$  的协变导数定义为  $\frac{dY}{dt}$  在  $\mathcal{M}$  的切空间上的投影:

$$\frac{DY}{dt} = \pi_{T_{p(t)}} \left( \frac{dY}{dt} \right). \quad (3.10)$$

**定义38.** 如果  $\frac{DY}{dt} \equiv 0$ , 则称  $Y$  沿  $p(t)$  平行移动。

和欧氏空间不同, 一般流形上的向量场沿封闭曲线平行移动时不一定会回到初始向量。这和流形的挠率有关。欧氏空间的子流形总是无挠的。

**定义39.** 如果一条曲线的切向量关于该曲线自己是平行移动的, 则称该曲线为测地线。

对于充分接近的两个点, 测地线是连接它们的最短曲线。

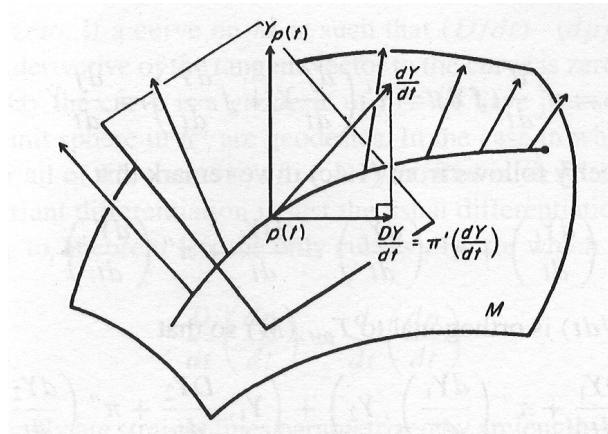


图 3.9: 协变导数

### 3.1.5 指数映射、法坐标系[16]

**定义40.** 设  $v \in T_p(\mathcal{M})$  且假定存在一条测地线  $\gamma : [0, 1] \rightarrow \mathcal{M}$ , 使得

$$\gamma(0) = p, \quad \gamma'(0) = v.$$

用  $\exp_p(tv)$  来表示上述测地线上的点  $\gamma(t)$ , 即

$$\exp_p(tv) = \gamma(t), \quad t \in [0, 1].$$

这样就确定了  $T_p(\mathcal{M})$  中原点的一个邻域  $B$  到  $\mathcal{M}$  的一个映射:

$$\exp_p : B \rightarrow \mathcal{M}.$$

称为关于  $p$  点的指数映射。

指数映射的几何意义是沿  $\gamma$  由  $p$  到  $\gamma(t) = \exp_p(tv)$  的弧长等于  $t\|v\|$ 。当  $\|v\|$  充分小时, 指数映射总是唯一确定的。

我们可以选取  $p$  的邻域  $W$  使得  $W$  内任何两点间的测地线全都在  $W$  中。这样的邻域称为点  $p$  的测地凸邻域。对于任意  $q \in W$ , 存在  $T_q(\mathcal{M})$  上的开  $\varepsilon$ -球  $N_\varepsilon(q)$  和  $\mathcal{M}$  的开

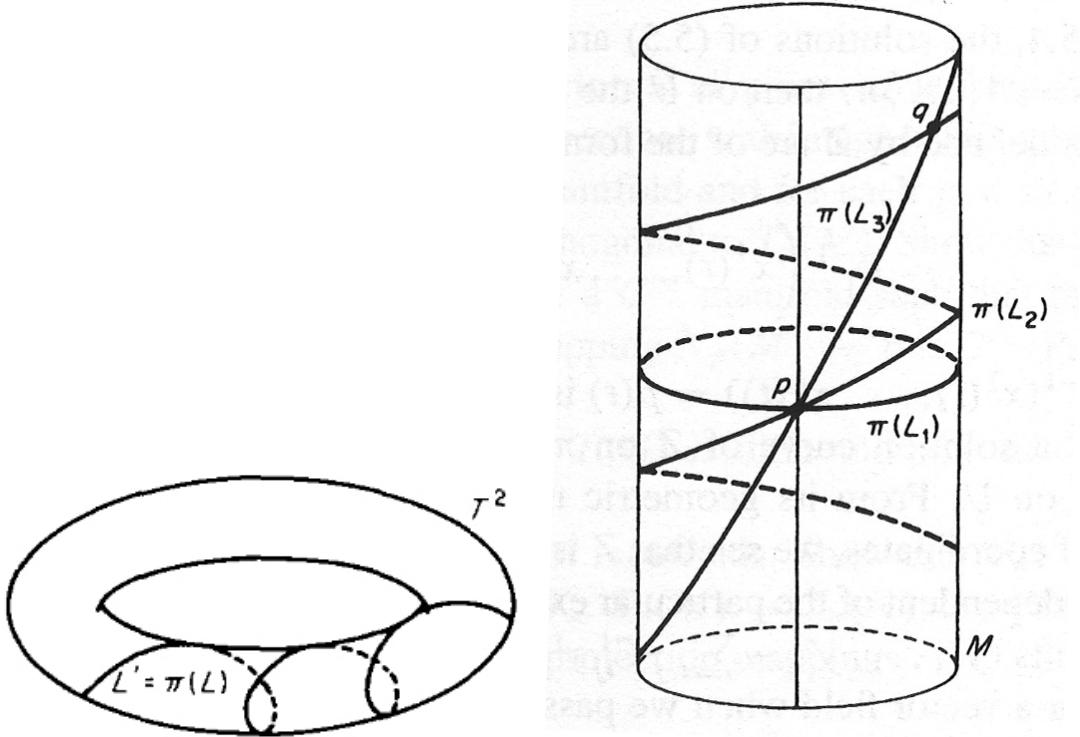


图 3.10: 测地线

集  $U_q$ , 使得指数映射  $\exp_q : N_\varepsilon(q) \rightarrow U_q$  是微分同胚。我们就可以用  $T_q(\mathcal{M})$  上的坐标来参数化  $q$  的邻域  $U_q$ , 得到  $q$  的一个坐标图。这个坐标图称为  $q$  点的法坐标图。具体描述如下。我们取  $T_q(\mathcal{M})$  的一个标准正交基  $\{e_i | i = 1, \dots, k\}$ 。则任何  $v \in T_q(\mathcal{M})$  可表为  $v = y^i e_i$ 。定义微分同胚  $\psi : T_q(\mathcal{M}) \rightarrow \mathbb{R}^k, v \mapsto (y^1, \dots, y^k)$ , 它把  $N_\varepsilon$  映射为  $\mathbb{R}^k$  中的一个开  $\varepsilon$ -球  $B_\varepsilon^k(0)$ 。这样复合映射  $\varphi = \psi \circ \exp_q^{-1} : U_q \rightarrow B_\varepsilon^k(0)$  就把  $q$  的一个开邻域  $U_q$  微分同胚地映射成  $B_\varepsilon^k(0)$ 。因此  $(U_q, \varphi)$  是  $\mathcal{M}$  上的一个坐标图。 $(U_q, \varphi; y^i)$  称为  $q$  点的法坐标图,  $(y^1, \dots, y^k)$  称为关于点  $q$  的法坐标系。

指数映射的逆函数称为对数映射, 它在  $\mathcal{M}$  上每一点的充分小的邻域内也是唯一确定的。

### 3.1.6 Operators on Manifolds[133]

Let the Riemannian metric derived from  $g$  be denoted by  $\mathbf{G} \triangleq \mathbf{G}_g = [g_{ij}]$ , which is a matrix function on  $\mathcal{M}$ . Since it is positive definite,  $|\mathbf{G}| > 0$  everywhere. For convenience, we write the inverse  $\mathbf{G}^{-1} = [g^{ij}]$ .

The gradient of  $f$  on  $\mathcal{M}$  w.r.t.  $g$  is a vector field defined by

$$\text{grad } f = [(\text{grad } f)^1, (\text{grad } f)^2, \dots, (\text{grad } f)^k]^T, \quad (3.11)$$

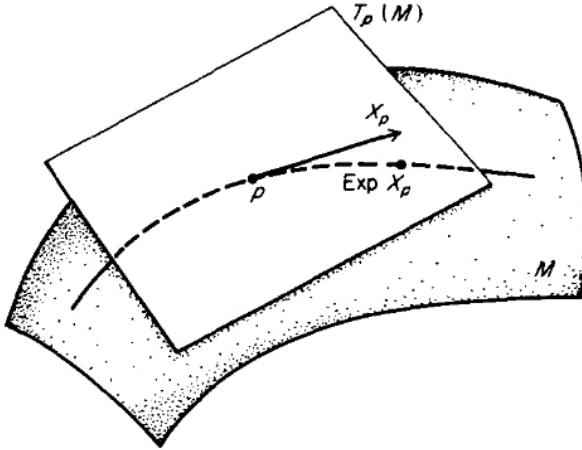


图 3.11: 指数映射与对数映射

with

$$(\text{grad } f)^i = g^{ij} \partial_j f. \quad (3.12)$$

The divergence of a vector field  $\mathbf{X} = [X^1, X^2, \dots, X^k]^T$  on  $\mathcal{M}$  is defined by

$$\text{div } \mathbf{X} = \frac{1}{\sqrt{|\mathbf{G}|}} \partial_i (\sqrt{|\mathbf{G}|} X^i). \quad (3.13)$$

**定义41.** Laplace-Beltrami operator on  $C^2(\mathcal{M})$  is defined by

$$\Delta f = \text{div grad } f = \frac{1}{\sqrt{|\mathbf{G}|}} \partial_i (\sqrt{|\mathbf{G}|} g^{ij} \partial_j f). \quad (3.14)$$

Note that in reality, it is  $-\Delta$  that is called the Laplace-Beltrami operator. When  $g$  is canonic, grad, div, and  $\Delta$  reduce to the traditional definition on Euclidean spaces.

### 3.1.7 Stiefel流形和Grassmann流形[43]

所有秩为  $k$  的  $m \times k$  矩阵的集合是一个流形（因为它是矩阵流形  $\mathcal{M}_{m \times k}$  的开子集），称为  $k$ -标架流形，记作  $F(m, k)$ 。它的维数为  $mk$ 。

所有列单位正交的  $m \times k$  矩阵的集合是一个流形，称为 Stiefel 流形，记作  $V(m, k)$ 。它的维数为  $mk - k(k + 1)/2$ 。

$\mathbb{R}^m$  所有维数为  $k$  的子空间的集合也是一个流形，称为 Grassmann 流形，记作  $G(m, k)$ 。它的维数为  $k(m - k)$ 。

如果在 Stiefel 流形  $V(m, k)$  上定义等价关系：如果存在  $k \times k$  正交阵  $O$  使得  $V_1 = V_2 O$ ，则  $V_1 \sim V_2$ ，则 Grassmann 流形是 Stiefel 流形在该等价关系下的商流形。

如果在  $k$ -标架流形  $F(m, k)$  上定义等价关系：如果存在  $k \times k$  非奇异矩阵  $M$  使得  $F_1 = F_2 M$ ，则  $F_1 \sim F_2$ ，则 Grassmann 流形是  $k$ -标架流形在该等价关系下的商流形。

## 3.2 谱图理论

在模式识别中经常要寻找近邻样本和计算样本间的相似性，然后再根据相似性进行识别和聚类。因此，样本之间按照相似性自然构成图，样本间的相似度由权重矩阵描述。由于需要用到样本属性的连续性，即相近样本的属性也要相近，就自然导出权重矩阵的Laplace矩阵。在归一化的条件下，样本属性是Laplace矩阵的特征向量。可以说样本间的相似性图是Riemann流形的离散化，Laplace矩阵是Riemann流形上Laplace-Beltrami算子的离散化。最后，样本的聚类可以通过计算Laplace矩阵的特征向量来实现，导出谱聚类方法。

### 3.2.1 Similarity and Dissimilarity of Data[133]

Dimensionality reduction is based on the assumption that the observed high-dimensional data set  $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\} \subset \mathbb{R}^D$  lies on a low-dimensional  $d$ -manifold  $\mathcal{M} \subset \mathbb{R}^D$ ,  $d \ll D$ . In many applications, the data sets are finite. Hence, we consider the data set  $\mathcal{X}$  as a set of sampling points on the manifold  $\mathcal{M}$ . However, the data usually is not directly sampled from an existent manifold. For example, when we digitize a text document into a vector, the vector is built by counting the keywords, not sampled from a known manifold. Similarly, in face recognition, a vector in a data set may be an ordered gray-level values of pixels, and in the representation of a hyperspectral image, the vector is a discrete spectral curve at a raster cell of the HSI cube. All of these data sets are not directly sampled from existent manifolds, but from digitized objects. Therefore, the similarities and/or dissimilarities between the objects of a data set are mathematically described by a *neighborhood* system. There are several ways to define the similarities and/or dissimilarities.

#### 3.2.1.1 Similarities and/or Dissimilarities from Distances

There are multiple ways to compute the distances between samples. Typical distances include:

1. Euclidean distance:

$$d(\mathbf{x}_1, \mathbf{x}_2) = \|\mathbf{x}_1 - \mathbf{x}_2\|.$$

2. Mahalanobis distance:

$$d(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{(\mathbf{x}_1 - \mathbf{x}_2)^T \mathbf{A} (\mathbf{x}_1 - \mathbf{x}_2)} \quad (\mathbf{A} \succeq 0).$$

3.  $\ell_p$  distance:

$$d(\mathbf{x}_1, \mathbf{x}_2) = \|\mathbf{x}_1 - \mathbf{x}_2\|_p, (p \geq 1).$$

4. Angular distance:

$$d(\mathbf{x}_1, \mathbf{x}_2) = \left\| \frac{\mathbf{x}_1}{\|\mathbf{x}_1\|} - \frac{\mathbf{x}_2}{\|\mathbf{x}_2\|} \right\|.$$

5. Cosine distance:

$$d(\mathbf{x}_1, \mathbf{x}_2) = 1 - \left| \frac{\mathbf{x}_1^T \mathbf{x}_2}{\|\mathbf{x}_1\| \|\mathbf{x}_2\|} \right|.$$

6.  $\chi^2$  distance:

$$d(\mathbf{x}_1, \mathbf{x}_2) = \sum_{i=1}^D \frac{(x_{1i} - x_{2i})^2}{x_{1i} + x_{2i}}.$$

7. Earth mover's distance (distance between distributions):

Cosine distance is often used in text retrieval.  $\chi^2$  distance is often used to compute the distance between histograms, such LBP and HoG features. A mapping by Gaussian function may be involved to define the similarities between samples. When the pairwise distances/similarities are computed we can find the  $\varepsilon$ -neighborhood or  $k$ -NN neighborhood to construct a graph.

**练习42.** 用 *Data for MATLAB hackers* (<http://www.cs.nyu.edu/~roweis/data.html>) 的 *UMist Faces* 人脸数据来测试以上 1、3 ( $p = 1$ )、4、5 哪个距离相对来说最适合分类，判断准则为：类内样本平均距离 / 类间样本平均距离。

### 3.2.1.2 Similarities and/or Dissimilarities from Sparse Representation

Before Yan et al.'s work [32],  $\varepsilon$ -neighborhood and  $k$ -NN neighborhood are almost the only methods to construct a graph. In 2010, Yan et al. proposed computing the similarity via sparse representation. They first solve the following standard bases pursuit problem:

$$\min_{\alpha_i} \|\alpha_i\|_1, \quad s.t. \quad \mathbf{x}_i = [\mathbf{X}_{\hat{i}}, \mathbf{I}] \alpha_i, \quad (3.15)$$

where  $\mathbf{X}_{\hat{i}} = [\mathbf{x}_1, \dots, \mathbf{x}_{i-1}, \mathbf{x}_{i+1}, \dots, \mathbf{x}_n]$ . Then the similarity between  $\mathbf{x}_i$  and  $\mathbf{x}_j$  is assigned as  $|\alpha_i[j]|$ . Zhuang et al. [167] also used low-rank representation and sparsity to compute the similarity between samples. Similar ideas also appeared in [45] and [82].

### 3.2.2 无向图及其Laplace矩阵[34]

设样本  $\{x_i\}$ ,  $i = 1, \dots, k$ , 是  $n$  维向量。如果样本间的相似性是对称的, 即  $d(x_i, x_j) = d(x_j, x_i)$ , 则对应的图  $G$  是无向图, 权重矩阵  $\mathbf{W} = (W_{ij})$  是对称非负矩阵,  $W_{ij} = d(x_i, x_j)$ 。和权重矩阵紧密相关的是邻接矩阵  $\mathbf{A} = (A_{ij})$ :

$$A_{ij} = \begin{cases} 1, & \text{if } W_{ij} \neq 0, \\ 0, & \text{if } W_{ij} = 0. \end{cases} \quad (3.16)$$

如果一个图有 $m$ 个连通分支，则其邻接矩阵可以排成块对角形。

给定一个顶点 $\mathbf{x}_i$ , 其邻域定义为

$$N(i) \triangleq \{j, A_{ij} \neq 0\}. \quad (3.17)$$

其顶点容积 (node volume, 也称为顶点的度, degree) 定义为

$$d(i) \triangleq \sum_{j \in N(i) \cup \{i\}} W_{ij}, \quad (3.18)$$

即 $\mathbf{W}$ 的行和。

给定顶点集的子集 $C$ , 其容积定义为

$$\text{vol}(C) = \sum_{i \in C} d(i). \quad (3.19)$$

其边界容积定义为

$$\text{vol}(\partial C) = \sum_{i \in C, j \notin C} W_{ij}. \quad (3.20)$$

如果 $f(x)$ 是样本 $x$ 的属性值，则我们自然要求相近样本的属性也要相近，因此经常要求解如下优化问题：

$$\min_{\mathbf{f}} \frac{1}{2} \sum_{i,j=1}^k W_{ij} (f_i - f_j)^2, \quad s.t. \quad \|\mathbf{f}\| = 1. \quad (3.21)$$

把上式整理成矩阵形式

$$\min_{\mathbf{f}} \mathbf{f}^T \mathbf{L}_W \mathbf{f}, \quad s.t. \quad \|\mathbf{f}\| = 1, \quad (3.22)$$

其中

$$\mathbf{L}_W = \mathbf{D}_W - \mathbf{W}, \quad \mathbf{D}_W = \text{diag}(\{d_i\}), \quad d_i = \sum_j W_{ij}$$

称为 $\mathbf{W}$ 的Laplace矩阵， $d_i$ 称为顶点*i*的度。显然 $\mathbf{L}_W$ 非负定，且有平凡的特征值0和对应的平凡特征向量 $\mathbf{1}$ 。应用上我们要求的是对应于次小特征值的特征向量。非负对称矩阵的Laplace矩阵还有如下性质。

**定理43.** Laplace矩阵的0特征值的代数重数等于图 $G$ 的连通分支的个数。

该定理在Feng et al. CVPR2014论文中被用于强化表示矩阵的块对角结构[48]。

Laplace矩阵的形式依赖于优化问题的形式。比如，如果(3.21)的归一化条件改为

$$\sum_i d_i f_i^2 = 1,$$

则新问题的矩阵形式变成

$$\min_{\mathbf{g}} \mathbf{g}^T \tilde{\mathbf{L}}_W \mathbf{g}, \quad s.t. \quad \|\mathbf{g}\| = 1, \quad (3.23)$$

其中  $\mathbf{g} = \mathbf{D}_W^{\frac{1}{2}} \mathbf{f}$ 。对应的Laplace矩阵变成：

$$\tilde{\mathbf{L}}_W = \mathbf{D}_W^{-\frac{1}{2}} (\mathbf{D}_W - \mathbf{W}) \mathbf{D}_W^{-\frac{1}{2}} = \mathbf{I} - \mathbf{D}_W^{-\frac{1}{2}} \mathbf{W} \mathbf{D}_W^{-\frac{1}{2}},$$

称为归一化的Laplace矩阵。它在基于normalized cut的谱聚类里出现，见第3.2.4节。

Laplace矩阵可以被认为是Laplace-Beltrami算子 $-\Delta$ 的离散形式。

**练习44.** 1. 证明： $\frac{1}{2} \sum_{i,j} W_{ij} \|\mathbf{f}_i - \mathbf{f}_j\|^2 = \text{tr}(\mathbf{F}^T \mathbf{L}_W \mathbf{F})$ , 其中  $\mathbf{F} = (\mathbf{f}_1, \dots, \mathbf{f}_n)$ .

2. 请推导  $\frac{1}{2} \sum_{i,j} W_{ij} (f_i - f_j)^2$  在约束  $\sum_i W_{ii} f_i^2 = 1$  下的Laplacian矩阵。

### 3.2.3 有向图及其Laplace矩阵[164]

如果样本间的相似性是不对称的，即  $d(x_i, x_j) \neq d(x_j, x_i)$ ，则对应的图是有向图，它的权重矩阵  $\mathbf{W}$  是非对称的， $\mathbf{W}_{ij} = d(x_i, x_j)$  代表的是有向边  $x_i \rightarrow x_j$  的权重。对于有向图，其邻接矩阵是不对称的：

$$A_{ij} = \begin{cases} 1, & \text{if } W_{ij} \neq 0, \\ 0, & \text{if } W_{ij} = 0. \end{cases} \quad (3.24)$$

此时我们需要区分顶点的入度和出度。它们分别定义为：

$$d^-(i) = \sum_j W_{ji}, \quad \text{即矩阵列和} \quad (3.25)$$

$$d^+(i) = \sum_j W_{ij} \quad \text{即矩阵行和.} \quad (3.26)$$

可以类似定义一个顶点集  $C$  的入容积 (in-volume) 和出容积 (out-volume) 为

$$\text{vol}^-(C) = \sum_{i \in C} d^-(i), \quad (3.27)$$

$$\text{vol}^+(C) = \sum_{i \in C} d^+(i), \quad (3.28)$$

边界入容积 (boundary in-volume) 和边界出容积 (boundary out-volume) 为

$$\text{vol}^-(\partial C) = \sum_{i \in C, j \notin C} W_{ji}, \quad (3.29)$$

$$\text{vol}^+(\partial C) = \sum_{i \in C, j \notin C} W_{ij}. \quad (3.30)$$

给定有向图 $G$ , 其上有一个自然的随机游走。它从顶点 $i$ 到顶点 $j$ 的转移概率是 $p_{ij} = W_{ij}/d_i^+$ 。如果 $G$ 是强连通的, 即任何顶点对 $(i, j)$ , 都存在从 $i$ 到 $j$ 的路径, 则存在一个唯一的平稳分布 $\pi$ 满足

$$\pi_j = \sum_i \pi_i p_{ij}, \quad \forall j, \quad (3.31)$$

$$\pi_i > 0, \quad \forall i, \quad (3.32)$$

即

$$\pi^T = \pi^T \mathbf{P}, \quad \pi > 0. \quad (3.33)$$

这是Perron-Frobenius定理的推论。

于是可以定义目标函数:

$$\frac{1}{2} \sum_{i,j=1}^k \pi_i p_{ij} \left( \frac{g_i}{\sqrt{\pi_i}} - \frac{g_j}{\sqrt{\pi_j}} \right)^2, \quad s.t. \quad \|\mathbf{g}\| = 1. \quad (3.34)$$

它可以写成矩阵形式:

$$\mathbf{g}^T (\mathbf{I} - \Theta) \mathbf{g}, \quad s.t. \quad \|\mathbf{g}\| = 1, \quad (3.35)$$

其中 $\Theta = \frac{1}{2}(\Pi^{\frac{1}{2}} \mathbf{P} \Pi^{-\frac{1}{2}} + \Pi^{-\frac{1}{2}} \mathbf{P}^T \Pi^{\frac{1}{2}})$ ,  $\Pi = \text{diag}(\pi)$ 。于是Laplace矩阵为 $\tilde{\mathbf{L}}_G = \mathbf{I} - \Theta$ 。它是目标函数

$$\frac{1}{2} \sum_{i,j=1}^k \pi_i p_{ij} (f_i - f_j)^2, \quad s.t. \quad \sum_i \pi_i f_i^2 = 1 \quad (3.36)$$

的Laplace矩阵

$$\mathbf{L}_G = \Pi - \frac{1}{2}(\Pi \mathbf{P} + \mathbf{P}^T \Pi) \quad (3.37)$$

的归一化。

当 $G$ 是无向图时, 其上的随机游走的平稳分布有解析解:

$$\pi_i = \frac{d_i}{\sum_j d_j}. \quad (3.38)$$

则目标泛函(3.34)退化成(3.21) 的目标泛函,  $\Theta$ 退化成 $\mathbf{D}_W^{-\frac{1}{2}} \mathbf{W} \mathbf{D}_W^{-\frac{1}{2}}$ 。

矩阵 $\Theta$ 有如下性质:

**定理45.**  $\Theta$ 的特征值都在区间 $[-1, 1]$ 内, 对应于1的特征向量是 $\sqrt{\pi}$ 。

Proof. 由 $\tilde{\mathbf{L}}_G$ 的非负定性易得 $\Theta$ 的特征值都不超过1. 困难在于证明 $\Theta$ 的特征值都不小于-1。 (3.33)可改写为

$$\sum_j p_{ji} \pi_j = \pi_i, \quad \forall i.$$

对任意  $\mathbf{x}$ ,

$$\begin{aligned}
 |\mathbf{x}^T \Theta \mathbf{x}| &= |\mathbf{x}^T \Pi^{-\frac{1}{2}} \mathbf{P}^T \Pi^{\frac{1}{2}} \mathbf{x}| = \sum_{ij} x_i x_j \pi_i^{-\frac{1}{2}} p_{ji} \pi_j^{\frac{1}{2}} \leq \sum_i |x_i| \pi_i^{-\frac{1}{2}} \sum_j |x_j| p_{ji} \pi_j^{\frac{1}{2}} \\
 &\leq \sum_i |x_i| \pi_i^{-\frac{1}{2}} \sqrt{\left( \sum_j x_j^2 p_{ji} \right) \left( \sum_j p_{ji} \pi_j \right)} = \sum_i |x_i| \pi_i^{-\frac{1}{2}} \sqrt{\left( \sum_j x_j^2 p_{ji} \right) \pi_i} \\
 &= \sum_i |x_i| \sqrt{\left( \sum_j x_j^2 p_{ji} \right)} \leq \sqrt{\left( \sum_i x_i^2 \right) \left( \sum_i \sum_j x_j^2 p_{ji} \right)} \\
 &= \sqrt{\left( \sum_i x_i^2 \right) \left( \sum_j x_j^2 \sum_i p_{ji} \right)} = \sqrt{\left( \sum_i x_i^2 \right) \left( \sum_j x_j^2 \right)} = \|\mathbf{x}\|^2.
 \end{aligned}$$

练习46. 证明 (3.34) 等价于 (3.35)。

### 3.2.4 谱聚类[119]

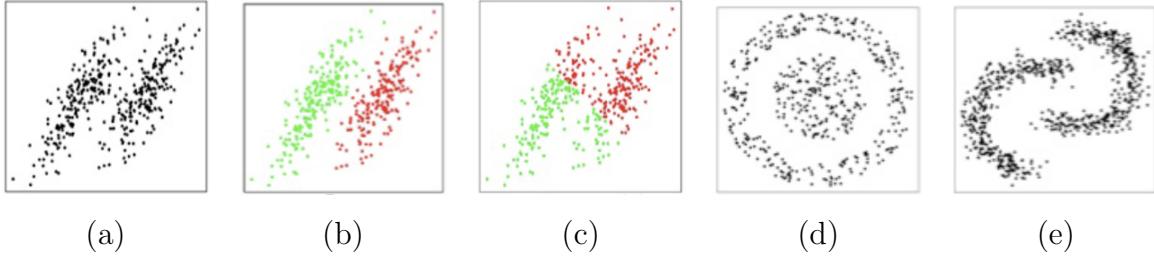


图 3.12: K-means一般只适用每一类为球形分布的数据。(a)、(d)、(e): K-means不适用的例子, 可以用谱聚类来给出正确的聚类。(b): (a)中数据的正确聚类结果。(c): (a)中数据的K-means聚类结果。

给定图  $G = (V, E)$ , 我们要通过去除一些边把它分成两个不相交的子图。设两子图的顶点集分别为  $V_1$  和  $V_2$ 。我们希望子图间的连接相对于子图和外界的连接来说尽可能疏松, 这样才能得到原图的合理的分割。定义子图间的切割代价为

$$cut(V_1, V_2) = \sum_{i \in V_1, j \in V_2} W_{ij}. \quad (3.39)$$

子图和外部的关联度为

$$assoc(V_1, V) = \sum_{i \in V_1, t \in V} W_{it}, \quad assoc(V_2, V) = \sum_{j \in V_2, t \in V} W_{jt}. \quad (3.40)$$

于是得到归一化的切割 (normalized cut) 准则:

$$Ncut(V_1, V_2) = \frac{cut(V_1, V_2)}{assoc(V_1, V)} + \frac{cut(V_2, V_1)}{assoc(V_2, V)}. \quad (3.41)$$

直接极小化 $cut(V_1, V_2)$ 称为最小切割 (minimum cut) 准则, 但它会导致一些少数的孤立点集合被分割出来。

为了求解normalized cut, 我们用 $x_i = 1$ 来表示顶点*i*属于 $V_1$ ,  $x_i = -1$ 来表示顶点*i*属于 $V_2$ 。定义 $c = \frac{\sum_{x_i > 0} d_i}{\sum_i d_i}$ 。则

$$Ncut(\mathbf{x}) = \frac{\sum_{x_i > 0, x_j < 0} -W_{ij}x_i x_j}{\sum_{x_i > 0} d_i} + \frac{\sum_{x_i < 0, x_j > 0} -W_{ij}x_i x_j}{\sum_{x_i < 0} d_i} \quad (3.42)$$

$$= \frac{(\mathbf{1} + \mathbf{x})^T \mathbf{L}_W (\mathbf{1} + \mathbf{x})}{4c\mathbf{1}^T \mathbf{D}_W \mathbf{1}} + \frac{(\mathbf{1} - \mathbf{x})^T \mathbf{L}_W (\mathbf{1} - \mathbf{x})}{4(1 - c)\mathbf{1}^T \mathbf{D}_W \mathbf{1}}. \quad (3.43)$$

Proof.

$$\begin{aligned} (\mathbf{1} + \mathbf{x})^T \mathbf{L}_W (\mathbf{1} + \mathbf{x}) &= \sum_i (1 + x_i)^2 d_i - \sum_{ij} (1 + x_i) W_{ij} (1 + x_j) \\ &= 4 \sum_{x_i > 0} d_i - 4 \sum_{x_i > 0, x_j > 0} W_{ij} = 4 \left( \sum_{x_i > 0} \sum_j W_{ij} - \sum_{x_i > 0, x_j > 0} W_{ij} \right) \\ &= 4 \sum_{x_i > 0, x_j < 0} W_{ij} = -4 \sum_{x_i > 0, x_j < 0} W_{ij} x_i x_j. \end{aligned}$$

同样,

$$(\mathbf{1} - \mathbf{x})^T \mathbf{L}_W (\mathbf{1} - \mathbf{x}) = - \sum_{x_i < 0, x_j > 0} W_{ij} x_i x_j.$$

定义 $b = \frac{c}{1 - c}$ ,  $\mathbf{y} = [(\mathbf{1} + \mathbf{x}) - b(\mathbf{1} - \mathbf{x})]/2$ , 则在相差常数项和常数系数的情况下

$$Ncut(\mathbf{x}) = \frac{\mathbf{y}^T \tilde{\mathbf{L}}_W \mathbf{y}}{\mathbf{y}^T \mathbf{D}_W \mathbf{y}}, \quad (3.44)$$

其中 $y_i \in \{1, -b\}$ ,  $\mathbf{y}^T \mathbf{D}_W \mathbf{1} = 0$ 。实际求解时, 我们可以把条件 $y_i \in \{1, -b\}$ 放宽成 $y_i \in \mathbb{R}$ , 则 $\mathbf{y}^T \mathbf{D}_W \mathbf{1} = 0$ 刚好就是要求 $\mathbf{y}$ 和平凡特征向量 $\mathbf{1}$ 正交。此时最优的 $\mathbf{y}$ 就是对应于 $\tilde{\mathbf{L}}_W$ 的次小特征值的特征向量。求得连续的 $\mathbf{y}$ 之后, 我们可以根据 $y_i$ 的符号判定第*i*个顶点应当属于 $V_1$ 还是 $V_2$ 。更准确的做法是要回头检验 $y_i$ 离1近还是离 $-b$ 近。

类似地, 有向图的normalized cut也将导致(3.35)中的目标函数, 其中

$$Ncut(S) = \frac{vol(\partial S)}{vol(S)} + \frac{vol(\partial S^c)}{vol(S^c)}, \quad vol(X) = \sum_{i \in X} \pi_i, \quad vol(\partial X) = \sum_{(i \rightarrow j) \in \partial X} \pi_i p_{ij}. \quad (3.45)$$

对于多类分割, 可以重复两类分割的过程, 或把目标函数(3.41)推广成多类分割, 详见[147]。

**练习47.** 用 *Data for MATLAB hackers* (<http://www.cs.nyu.edu/~roweis/data.html>) 的 *UMist Faces* 两个人的人脸数据，计算出图像间的欧氏距离，再用 *Gauss核*  $\exp(-d/\sigma)$  转化为相似度矩阵  $\mathbf{W}$ ，其中  $\sigma$  取作类内距离标准差（方差开根号）的平均值，最后利用 *NCut* 进行聚类，汇报一下准确率。

## 第四章 线性降维技术

本章内容主要取自[133]。测试数据集：Data for MATLAB hackers (<http://www.cs.nyu.edu/~roweis/data.html>)、UCI Machine Learning Repository (<http://archive.ics.uci.edu/ml/>)

### 4.1 线性降维技术简介

线性降维主要有三种方法：主元分析（Principal Component Analysis, PCA）、多维标度法（Multi-Dimensional Scaling, MDS）、和随机投影（Random Projection, RP）。

### 4.2 Principal Component Analysis

#### 4.2.1 总体主成分

##### 4.2.1.1 总体主成分的定义

设 $X_1, X_2, \dots, X_p$ 为某实际问题所涉及的 $p$ 个随机变量。记 $\mathbf{X} = (X_1, X_2, \dots, X_p)^T$ ，其协方差矩阵为：

$$\Sigma = (\sigma_{ij})_{p \times p} = E[(\mathbf{X} - E(\mathbf{X}))(\mathbf{X} - E(\mathbf{X}))^T],$$

它是一个 $p$ 阶非负定矩阵。设 $\mathbf{l}_i = (l_{i1}, l_{i2}, \dots, l_{ip})^T$ 为 $p$ 个常数构成的向量。考虑如下线性组合：

$$\mathbf{Y} = \mathbf{L}^T \mathbf{X}, \quad (4.1)$$

其中 $\mathbf{Y} = (Y_1, Y_2, \dots, Y_p)^T$ ,  $\mathbf{L} = (\mathbf{l}_1, \mathbf{l}_2, \dots, \mathbf{l}_p)$ 。则有

$$\text{Var}(Y_i) = \text{Var}(\mathbf{l}_i^T \mathbf{X}) = \mathbf{l}_i^T \Sigma \mathbf{l}_i, \quad i = 1, 2, \dots, p,$$

$$\text{Cov}(Y_i, Y_j) = \text{Cov}(\mathbf{l}_i^T \mathbf{X}, \mathbf{l}_j^T \mathbf{X}) = \mathbf{l}_i^T \Sigma \mathbf{l}_j, \quad i, j = 1, 2, \dots, p.$$

如果我们希望用 $Y_1$ 代替原来 $p$ 个分量 $X_1, X_2, \dots, X_p$ ，这就要求 $Y_1$ 尽可能地反映原 $p$ 个变量的信息。这里“信息”用 $Y_1$ 的方差来度量，即要求 $\text{Var}(Y_1) = \mathbf{l}_1^T \Sigma \mathbf{l}_1$ 达到最大。但对任意常数 $k$ ，若取 $\tilde{\mathbf{l}}_1 = k\mathbf{l}_1$ ，则 $\text{Var}(\tilde{\mathbf{l}}_1^T \mathbf{X}) = k^2 \text{Var}(\mathbf{l}_1^T \mathbf{X}) = k^2 \mathbf{l}_1^T \Sigma \mathbf{l}_1$ 。因此，必须对 $\mathbf{l}_1$ 加以限制，否则 $\text{Var}(Y_1)$ 无界。最方便的限制是要求 $\mathbf{l}_1$ 具有单位长度，即我们在约束条件 $\mathbf{l}_1^T \mathbf{l}_1 = 1$ 之下求 $\mathbf{l}_1$ 使 $\text{Var}(Y_1)$ 达到最大。由此 $\mathbf{l}_1$ 所确定的随机变量 $Y_1 = \mathbf{l}_1^T \mathbf{X}$ 称为 $X_1, X_2, \dots, X_p$ 的第一主成分。

如果第一主成分 $Y_1$ 还不足以反映原变量的信息，可以进一步求第二主成分 $Y_2$ 。为了使 $Y_1$ 和 $Y_2$ 所反映的原变量的信息不相重叠，要求 $Y_1$ 与 $Y_2$ 不相关，即

$$\text{Cov}(Y_1, Y_2) = \mathbf{l}_1^T \boldsymbol{\Sigma} \mathbf{l}_2 = 0.$$

于是，在约束条件 $\mathbf{l}_2^T \mathbf{l}_2 = 1$ 及 $\mathbf{l}_1^T \boldsymbol{\Sigma} \mathbf{l}_2 = 0$ 之下，求 $\mathbf{l}_2$ 使 $\text{Var}(Y_2)$ 达到最大。由此 $\mathbf{l}_2$ 所确定的随机变量 $Y_2 = \mathbf{l}_2^T \mathbf{X}$ 称为 $X_1, X_2, \dots, X_p$ 的第二主成分。

一般地，在约束 $\mathbf{l}_i^T \mathbf{l}_i = 1$ 及 $\mathbf{l}_i^T \boldsymbol{\Sigma} \mathbf{l}_k = 0$  ( $k = 1, 2, \dots, i-1$ )之下，求 $\mathbf{l}_i$ 使 $\text{Var}(Y_i)$ 达到最大。由此 $\mathbf{l}_i$ 所确定的随机变量 $Y_i = \mathbf{l}_i^T \mathbf{X}$ 称为 $X_1, X_2, \dots, X_p$ 的第*i*个主成分。

#### 4.2.1.2 总体主成分的求法

有如下定理：

**定理48.** 设 $\boldsymbol{\Sigma}$ 的特征值及相应的特征向量分别为 $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_p \geq 0$ 及 $\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_p$ ，则 $\mathbf{X}$ 的第*i*个主成分为

$$Y_i = \mathbf{p}_i^T \mathbf{X}, \quad i = 1, 2, \dots, p, \quad (4.2)$$

且有

$$\text{Var}(Y_i) = \lambda_i, \quad i = 1, 2, \dots, p,$$

$$\text{Cov}(Y_i, Y_j) = 0, \quad i \neq j.$$

Proof. 令 $\mathbf{P} = (\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_p)$ 。则 $\mathbf{P}$ 为一正交矩阵，且

$$\mathbf{P}^T \boldsymbol{\Sigma} \mathbf{P} = \boldsymbol{\Lambda} = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_p).$$

设 $Y_1 = \mathbf{l}_1^T \mathbf{X}$ 为 $\mathbf{X}$ 的第一主成分，其中 $\mathbf{l}_1^T \mathbf{l}_1 = 1$ 。令

$$\mathbf{z}_1 = (z_{11}, z_{12}, \dots, z_{1p})^T = \mathbf{P}^T \mathbf{l}_1,$$

则

$$\begin{aligned} \text{Var}(Y_1) &= \mathbf{l}_1^T \boldsymbol{\Sigma} \mathbf{l}_1 = \mathbf{z}_1^T \mathbf{P}^T \boldsymbol{\Sigma} \mathbf{P} \mathbf{z}_1 = \mathbf{z}_1^T \boldsymbol{\Lambda} \mathbf{z}_1 = \lambda_1 z_{11}^2 + \lambda_2 z_{12}^2 + \dots + \lambda_p z_{1p}^2 \\ &\leq \lambda_1 (z_{11}^2 + z_{12}^2 + \dots + z_{1p}^2) = \lambda_1 \mathbf{z}_1^T \mathbf{z}_1 = \lambda_1 \mathbf{l}_1^T \mathbf{P} \mathbf{P}^T \mathbf{l}_1 = \lambda_1 \mathbf{l}_1^T \mathbf{l}_1 = \lambda_1, \end{aligned}$$

并且当 $\mathbf{z}_1 = \mathbf{e}_{p,1}$ 时，等号成立。此时

$$\mathbf{l}_1 = \mathbf{P} \mathbf{z}_1 = \mathbf{p}_1.$$

设 $Y_2 = \mathbf{l}_2^T \mathbf{X}$ 为 $\mathbf{X}$ 的第二主成分，则有

$$\mathbf{l}_2^T \mathbf{l}_2 = 1, \mathbf{l}_2^T \mathbf{p}_1 = 0.$$

令

$$\mathbf{z}_2 = (z_{21}, z_{22}, \dots, z_{2p})^T = \mathbf{P}^T \mathbf{l}_2,$$

则有

$$0 = \mathbf{l}_2^T \mathbf{p}_1 = \mathbf{z}_2^T \mathbf{P}^T \mathbf{p}_1 = z_{21}.$$

从而

$$\begin{aligned} \text{Var}(Y_2) &= \mathbf{l}_2^T \Sigma \mathbf{l}_2 = \mathbf{z}_2^T \mathbf{P}^T \Sigma \mathbf{P} \mathbf{z}_2 = \mathbf{z}_2^T \Lambda \mathbf{z}_2 = \lambda_1 z_{21}^2 + \lambda_2 z_{22}^2 + \dots + \lambda_p z_{2p}^2 \\ &= \lambda_2 z_{22}^2 + \dots + \lambda_p z_{2p}^2 \leq \lambda_2 (z_{21}^2 + z_{22}^2 + \dots + z_{2p}^2) = \lambda_2 \mathbf{z}_2^T \mathbf{z}_2 \\ &= \lambda_2 \mathbf{l}_2^T \mathbf{P} \mathbf{P}^T \mathbf{l}_2 = \lambda_2 \mathbf{l}_2^T \mathbf{l}_2 = \lambda_2, \end{aligned}$$

并且当  $\mathbf{z}_2 = \mathbf{e}_{p,2}$  时，等号成立。此时

$$\mathbf{l}_2 = \mathbf{P} \mathbf{z}_2 = \mathbf{p}_2.$$

类似可证明，对所有的  $p$  个主成分，定理的结论均成立。

## 4.2.2 总体主成分的性质

### 4.2.2.1 主成分的协方差矩阵及总方差

记  $\mathbf{Y} = (Y_1, Y_2, \dots, Y_p)^T$  为主成分向量，则  $\mathbf{Y} = \mathbf{P}^T \mathbf{X}$ ，其中  $\mathbf{P} = (\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_p)$ ，且

$$\text{Cov}(\mathbf{Y}) = \text{Cov}(\mathbf{P}^T \mathbf{X}) = \mathbf{P}^T \Sigma \mathbf{P} = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_p).$$

由此得出主成分的总方差为

$$\sum_{i=1}^p \text{Var}(Y_i) = \sum_{i=1}^p \lambda_i = \text{tr}(\mathbf{P}^T \Sigma \mathbf{P}) = \text{tr}(\Sigma \mathbf{P} \mathbf{P}^T) = \text{tr}(\Sigma) = \sum_{i=1}^p \text{Var}(X_i),$$

即主成分分析是把  $p$  的原始变量  $X_1, X_2, \dots, X_p$  的总方差  $\sum_{i=1}^p \text{Var}(X_i)$  分解为  $p$  个不相关变量  $Y_1, Y_2, \dots, Y_p$  的方差之和。由于  $\text{Var}(Y_i) = \lambda_i$ ， $\lambda_i / \sum_{k=1}^p \lambda_k$  描述了第  $k$  个主成分提取的信息占总信息的份额，我们称此为第  $k$  个主成分  $Y_k$  的贡献率。第一主成分的贡献率最大，表明  $Y_1 = \mathbf{p}_1^T \mathbf{X}$  综合原始变量  $X_1, X_2, \dots, X_p$  所含信息的能力最强，而  $Y_2, \dots, Y_p$  的综合能力依次减弱。前  $m$  个主分量的贡献率之和  $\sum_{i=1}^m \lambda_i / \sum_{i=1}^p \lambda_i$  称为  $Y_1, Y_2, \dots, Y_m$  的累计贡献率，它表明前  $m$  个主成分  $Y_1, Y_2, \dots, Y_m$  综合提供  $X_1, X_2, \dots, X_p$  中信息的能力。实际应用中，通常选取  $m < p$ ，使前  $m$  个主成分的累计贡献率达到较高的比例（如 80% 到 90%）。这样用前  $m$  个主成分  $Y_1, Y_2, \dots, Y_m$  代替原始变量  $X_1, X_2, \dots, X_p$  不但使变量维数降低，而且也不至于损失原始变量中的太多信息。

### 4.2.2.2 主成分 $Y_i$ 与变量 $X_j$ 的相关系数

由于 $\mathbf{Y} = \mathbf{P}^T \mathbf{X}$ , 故 $\mathbf{X} = \mathbf{P} \mathbf{Y}$ , 从而

$$X_j = p_{1j} Y_1 + p_{2j} Y_2 + \cdots + p_{pj} Y_p,$$

$$\text{Cov}(Y_i, X_j) = \lambda_i p_{ij}.$$

由此可得 $Y_i$ 与 $X_j$ 的相关系数为:

$$\rho_{Y_i, X_j} = \frac{\text{Cov}(Y_i, X_j)}{\sqrt{\text{Var}(Y_i)} \sqrt{\text{Var}(X_j)}} = \frac{\lambda_i p_{ij}}{\sqrt{\lambda_i} \sqrt{\sigma_{jj}}} = \sqrt{\frac{\lambda_i}{\sigma_{jj}}} p_{ij}. \quad (4.3)$$

它给出了主成分 $Y_i$ 与原始变量 $X_j$ 的关联性的度量。在实际应用中, 我们一般只关心前 $m$ 个主成分与原始变量的相关系数。

**例49.** [3]例4.1。

### 4.2.3 标准化变量的主成分

在实际问题中, 不同的变量往往有不同的量纲, 由于不同的量纲会引起个别变量取值的分散程度差异较大, 这时总体方差主要受方差较大的变量的控制。若用 $\Sigma$ 求主成分, 则优先照顾了方差大的变量, 有时会造成很不合理的结果。为了消除由于量纲的不同可能带来的影响, 常采用变量标准化的方法, 即令

$$X_i^* = \frac{X_i - \mu_i}{\sqrt{\sigma_{ii}}}, \quad i = 1, 2, \dots, p, \quad (4.4)$$

其中 $\mu_i = E(X_i)$ ,  $\sigma_{ii} = \text{Var}(X_i)$ 。这时 $\mathbf{X}^* = (X_1^*, X_2^*, \dots, X_p^*)^T$ 的协方差矩阵便是 $\mathbf{X} = (X_1, X_2, \dots, X_p)^T$ 的相关矩阵 $\rho = (\rho_{ij})_{p \times p}$ , 其中

$$\rho_{ij} = E(X_i^* X_j^*) = \frac{\text{Cov}(X_i, X_j)}{\sqrt{\sigma_{ii} \sigma_{jj}}}.$$

利用 $\mathbf{X}$ 的相关矩阵 $\rho$ 做主成分分析, 我们有如下结论:

$\mathbf{X}^*$ 的第 $i$ 个主成分为:

$$Y_i^* = (\mathbf{p}_i^*)^T \mathbf{X}^* = p_{i1}^* \frac{X_1 - \mu_1}{\sqrt{\sigma_{11}}} + p_{i2}^* \frac{X_2 - \mu_2}{\sqrt{\sigma_{22}}} + \cdots + p_{ip}^* \frac{X_p - \mu_p}{\sqrt{\sigma_{pp}}}. \quad i = 1, \dots, p,$$

并且

$$\sum_{i=1}^p \text{Var}(Y_i^*) = \sum_{i=1}^p \lambda_i^* = \sum_{i=1}^p \text{Var}(X_i^*) = p,$$

其中 $\lambda_1^* \geq \lambda_2^* \geq \cdots \geq \lambda_p^* \geq 0$ 为 $\rho$ 的特征值,  $\mathbf{p}_i^* = (p_{i1}^*, p_{i2}^*, \dots, p_{ip}^*)^T$ 为相应于特征值 $\lambda_i^*$ 的单位正交特征向量。这时, 第 $i$ 个主成分的贡献率为 $\lambda_i^*/p$ , 前 $m$ 个主成分的累计贡献率为 $\sum_{i=1}^m m \lambda_i^*/p$ ,  $Y_i^*$ 与 $X_j^*$ 的相关系数为 $\rho_{Y_i^*, X_j^*} = \sqrt{\lambda_i^*} p_{ij}^*$ 。

下面例子说明分别从协方差矩阵和相关矩阵出发求主成分的差异。

例50. [3]例4.2。

在实际应用中，当涉及的各变量的变化范围差异较大时，从相关矩阵 $\rho$ 出发求主成分比较合理。

思考题51. 如果我们按照别的方式对各变量做无量纲化，比如减去中值再除以四分位极差再做自相关矩阵的特征分解，相应的结论是什么？

#### 4.2.4 样本主成分

前面讨论的是总体主成分。但在实际问题中，一般 $\Sigma$ 或 $\rho$ 是未知的，需要通过样本来估计。设

$$\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{ip})^T, \quad i = 1, \dots, n$$

为取自 $\mathbf{X} = (X_1, \dots, X_p)^T$ 的一个容量为 $n$ 的简单随机样本，则样本协方差矩阵及样本相关矩阵分别为

$$\begin{aligned} \mathbf{S} = (s_{ij})_{p \times p} &= \frac{1}{n-1} \sum_{k=1}^n (\mathbf{x}_k - \bar{\mathbf{x}})(\mathbf{x}_k - \bar{\mathbf{x}})^T, \\ \mathbf{R} = (r_{ij})_{p \times p} &= \left( \frac{s_{ij}}{\sqrt{s_{ii}s_{jj}}} \right), \end{aligned} \quad (4.5)$$

其中 $\bar{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i$  为样本均值向量。分别以 $\mathbf{S}$ 或 $\mathbf{R}$ 按前述的方法求得的主成分称为样本主成分。具体有如下结论：

设 $\mathbf{S}$ 的特征值为 $\hat{\lambda}_1 \geq \hat{\lambda}_2 \geq \dots \geq \hat{\lambda}_p \geq 0$ ，相应的单位正交特征向量为 $\hat{\mathbf{p}}_1, \hat{\mathbf{p}}_2, \dots, \hat{\mathbf{p}}_p$ ，这里 $\hat{\mathbf{p}}_i = (\hat{p}_{i1}, \hat{p}_{i2}, \dots, \hat{p}_{ip})^T$ 。则第*i*个样本主成分为：

$$y_i = \hat{\mathbf{p}}_i^T \mathbf{x},$$

其中 $\mathbf{x}$ 为 $\mathbf{X}$ 的任一观测值。当依次代入 $\mathbf{X}$ 的*n*个观测值 $\mathbf{x}_i$ 时，便得到第*i*个样本主成分 $y_i$ 的*n*个观测值 $y_{ki}$ ，我们称为第*i*个样本主成分的得分。这时

$$\begin{cases} y_i \text{的样本方差} = \hat{\mathbf{p}}_i^T \mathbf{S} \hat{\mathbf{p}}_i = \hat{\lambda}_i, \\ y_i \text{与 } y_j \text{的样本协方差} = \hat{\mathbf{p}}_i^T \mathbf{S} \hat{\mathbf{p}}_j = 0, \quad i \neq j \\ \text{样本总方差} = \sum_{i=1}^p s_{ii} = \sum_{i=1}^p \hat{\lambda}_i. \end{cases}$$

第*i*个样本主成分的贡献率定义为 $\hat{\lambda}_i / \sum_{k=1}^p \hat{\lambda}_k$ ，前*m*个样本主成分的累计贡献率定义为 $\sum_{i=1}^m \hat{\lambda}_i / \sum_{k=1}^p \hat{\lambda}_k$ 。选取前*m*个样本主成分，使其累计贡献率达到一定的要求（如80%至90%），以前*m*个样本主成分的得分代替原始数据作分析，便可以达到降低数据维数的目的。

同样，为了消除量纲的影响，我们可以对样本进行标准化，即令

$$\mathbf{x}_i^* = \left( \frac{x_{i1} - \bar{x}_1}{\sqrt{s_{11}}}, \frac{x_{i2} - \bar{x}_2}{\sqrt{s_{22}}} \dots, \frac{x_{ip} - \bar{x}_p}{\sqrt{s_{pp}}} \right)^T,$$

则标准化数据的样本协方差矩阵即为原数据的样本相关矩阵 $\mathbf{R}$ 。由 $\mathbf{R}$ 出发求得的样本主成分称为标准化样本主成分。只要求出 $\mathbf{R}$ 的特征值及相应的正交单位特征向量，类似上述结果可求得标准化样本主成分。这时标准化样本的样本总方差为 $p$ 。

**例52.** [3]例4.3。

最后需要指出的是，关于主成分的实际意义，要结合具体问题和有关专业知识才能给出合理的解释。虽然利用主成分本身可以对所研究的问题在一定程度上作分析，但主成分分析本身往往不是最终目的，更重要的是要利用主成分综合原始变量的信息，达到降低原始变量维数的目的，进而利用前几个主成分的得分的地位数据做进一步分析，如主成分回归分析、聚类分析等。

**练习53.** 用Data for MATLAB hackers (<http://www.cs.nyu.edu/~roweis/data.html>) 的UMist Faces一个人的人脸数据做主成分分析和标准化变量的主成分分析，分别显示平均脸和前6个主成分。

#### 4.2.5 Testing PCA on Artificial Surfaces

PCA is a linear projection method. When the data reside on a subspace or a hyperplane, it works well. However, if the data reside on a nonlinear manifold, it often fails. In this subsection, we demonstrate the validity of PCA on the data sets sampled from several artificial surfaces.

#### 4.2.6 Applications of PCA

The motivation of DR is to convert high-dimensional data to low-dimensional ones to that the low-dimensional data can be effectively used in data processing systems. To understand how to apply DR technique in applications, we give two examples.

##### 4.2.6.1 PCA in Machine Learning

Let  $\mathcal{X} \in \mathbb{R}^D$  be a data set with extrinsic dimension  $D$ . Assume that the size of the data set, denoted by  $|\mathcal{X}|$ , is very large. In machine learning, as main task is to find a (vector-valued) function  $f$  on  $\mathcal{X}$ , which captures characteristics of interest of the data. These characteristics can be used in certain application, for example, in the classification of  $\mathcal{X}$ . We call such a function a *feature extractor* or a *feature function*, and call  $\mathcal{X}$  the

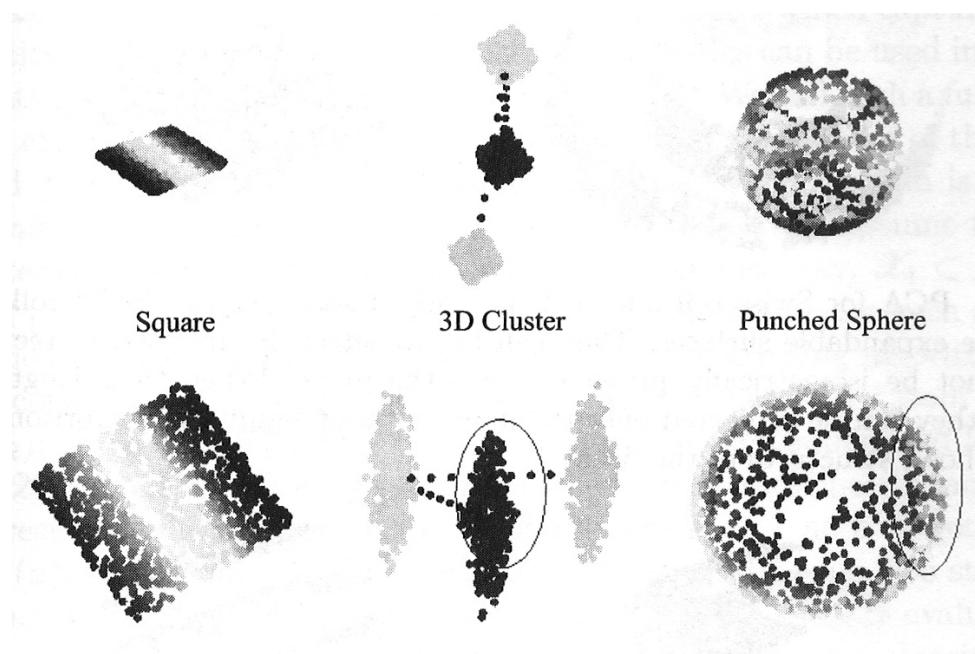


图 4.1: PCA for three surfaces. On the top, the left is a square in the space, the middle is a 3-D cluster, and the right is a punched sphere. Their 2-D DR data sets are displayed at the bottom correspondingly. It is shown that PCA cannot make valid DR for 3-D cluster and punched sphere since these data are samples from nonlinear surfaces.

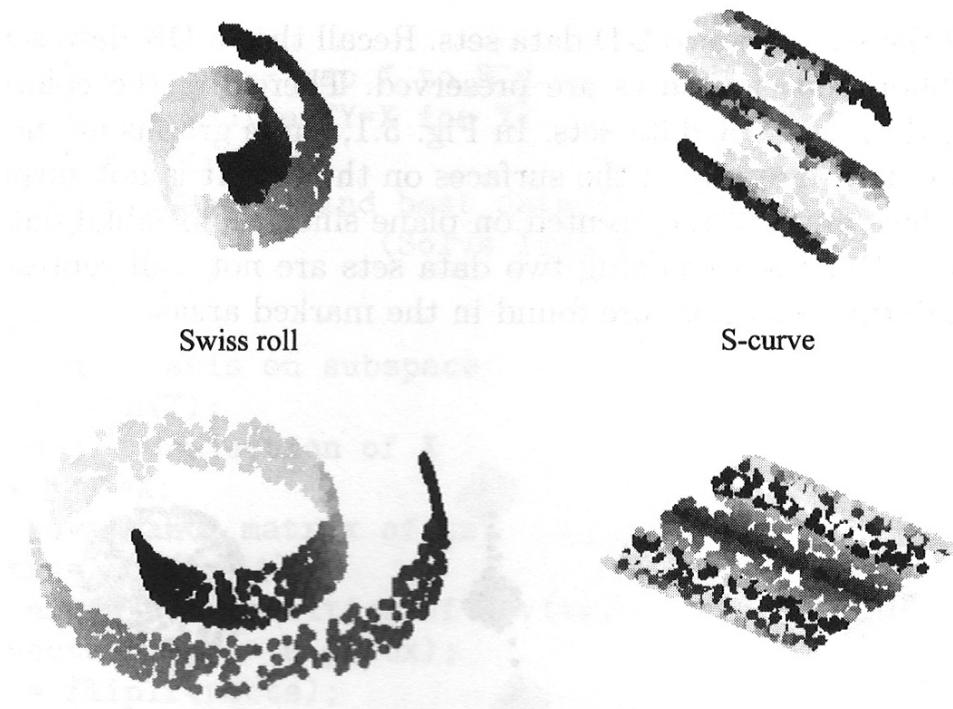


图 4.2: PCA for Swiss roll and S-curve with shorter lengths. Swiss roll and S-curve are expandable surfaces. They can be isometrically unfolded to rectangles, but cannot be isometrically projected onto the plane. When their lengths are short, they can be projected onto the plane without significant distortion on the neighborhood structure of the data.

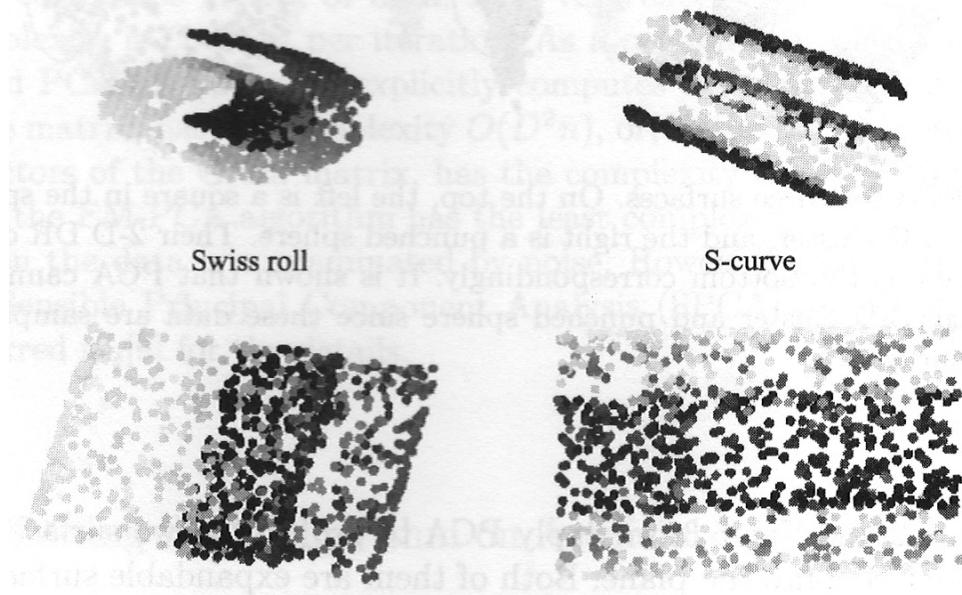


图 4.3: PCA for Swiss roll and S-curve with longer lengths. When their lengths are long, we cannot find the projection that preserve the neighborhood structure of the data.

*examples* of the learning and  $f(\mathcal{X})$  the features of  $\mathcal{X}$ . A machine learning algorithm is applied to learning the feature extractor  $f$  from the examples  $\mathcal{X}$ . Assume that the characteristics of interest of a small subset of examples, say,  $\mathcal{X}_1 \subset \mathcal{X}$  with  $|\mathcal{X}_1| \ll |\mathcal{X}|$ , are known in advance, i.e.,  $c = f(\mathbf{x})$  is known for each  $\mathbf{x} \in \mathcal{X}_1$ . We denote by  $\mathcal{L}_1$  the set  $f(\mathcal{X}_1) = \{f(\mathbf{x})\}_{\mathbf{x} \in \mathcal{X}_1}$ . A learning algorithm then can be considered as an operator  $L$ , which learns the function  $f$  from the pair  $(\mathcal{X}_1, \mathcal{L}_1) : f = L(\mathcal{X}_1, \mathcal{L}_1)$ . This type of learning is called supervised learning. Write  $\mathcal{X}_2 = \mathcal{X} \setminus \mathcal{X}_1$ . Once  $f$  is learned, then, to extract the characteristics of interest of each vector  $\mathbf{z} \in \mathcal{X}_2$ , we simply evaluate  $f$  at  $\mathbf{z}$  to get its features  $f(\mathbf{z})$ . Hence the whole machine learning processing had two steps:

1. constructing the feature extractor  $f$  vis the examples, and
2. evaluating  $f$  at unknown data.

Therefore, a (supervised) machine learning algorithm can be summarized by the diagram in Figure 4.4. If the dimension of data is high, these two steps in the processing are involved in high-dimensional computation. To avoid the high-dimensional data in the processing, DR techniques can be applied.

On the example set  $\mathcal{X}_1$ , we first determine the intrinsic dimension  $d$  of  $\mathcal{X}_1$ , then construct a DR mapping  $h$ , which maps  $\mathcal{X}_1$  onto  $\mathcal{Y}_1 \subset \mathbb{R}^d$ :  $h(\mathbf{x}) = \mathbf{y}$ . Since  $\mathcal{Y}_1$  preserves the characteristics of  $\mathcal{X}_1$ , we can learn the feature extractor from  $\mathcal{Y}_1$ . Let the new feature

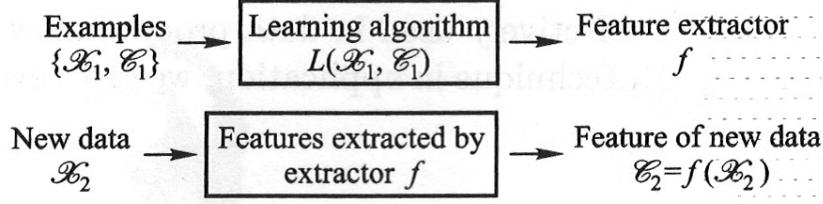


图 4.4: The data feature extraction via machine learning. Top: A learning algorithm is used to find a feature extractor from examples. Bottom: The learned feature extractor is applied to extracting the features of the new data.

extractor learned from  $\mathcal{Y}_1$  be denoted by  $g$ . Ideally, we shall have  $f(\mathbf{x}) = g(\mathbf{z})$ . Extend the DR mapping  $h$  on the example set  $\mathcal{X}_2$  and write  $\mathcal{Y}_2 = h(\mathcal{X}_2)$ . Then extend the extractor  $g$  on  $\mathcal{Y}_2$ . We now can learn the features of  $\mathcal{X}_2$  via the DR data set  $\mathcal{Y}_2$ . Hence, DR techniques enable us to convert the learning processing on the high dimensional examples in Figure 4.4 to the process on the DR data sets  $\mathcal{Y}_1$  and  $\mathcal{Y}_2$ , respectively. Since the new data processing performs on low-dimensional sets, it enables us to overcome the difficulties caused by the curse of dimensionality.

Note that a DR algorithm itself serves an unsupervised learning algorithm. In an unsupervised learning problem, the characteristics of interest of the example set  $\mathcal{X}_1$  are unknown and must be learned from examples. The projection in PCA can also be considered as a feature extractor, which extracts the principal components of the data. The principal components present features of the data.

We now introduce some applications, where PCA plays the roles of either feature extractor or a data simplifier. Due to the linearity of PCA, using PCA to reduce data dimension and extend feature functions is computationally economic. Hence, PCA is widely applied in real-world applications which need the feature extractions.

#### 4.2.6.2 PCA in Eigenfaces

For demonstration, we present a simplest eigenface algorithm. Let the training set be denoted by  $\mathbf{X}$ , with each column being a face image. There are  $n$  face images in total, each image being vectorized into a  $d$  dimensional vector.

**Step 1. Construction of eigenfaces.** Obtain the average face  $\bar{\mathbf{x}}$  and center all the face images. Then apply SVD to the centered face images, collected in  $\hat{\mathbf{X}}$ , to obtain eigenfaces  $F_1, \dots, F_d$  as the leading left singular vectors of  $\hat{\mathbf{X}}$ , and the  $d \times n$  weight matrix  $\mathbf{Y} = (\mathbf{y}_1, \dots, \mathbf{y}_n)$ . The eigen faces from an orthonormal basis of the face

space  $S_d$ . In the weight matrix, the  $i$ th column  $\mathbf{y}_i = (y_{1i}, \dots, y_{di})^T$  is the weight vector, which approximately represents the  $i$ th face  $X_i$  in the training set such that  $X_i \approx \sum_{j=1}^d y_{ji} F_j$ .

Step 2. **Computation of the weights.** Compute the weights of a new input face with respect to eigenfaces and the distance from the new input face to the face space. Assume that  $\mathbf{Z}$  is an input image. The weights of  $\mathbf{Z}$  with respect to eigenfaces are computed by

$$\mathbf{z} = \mathbf{F}^T(\mathbf{Z} - \bar{\mathbf{x}}),$$

and the distance from  $\mathbf{Z}$  to the face space  $S_d$  is computed by

$$d_1 = \|\mathbf{Z} - (\mathbf{F}\mathbf{z} + \bar{\mathbf{x}})\|.$$

Step3. **Face recognition.** Let  $\varepsilon_1$  and  $\varepsilon_2$  be the thresholds for determining whether an input image is a face and whether it is known face, respectively. If  $d_1 < \varepsilon_1$ , then  $Z$  is a face. Otherwise, it is not. In the case that  $Z$  is a face, compute the sum of distances between the weight vector  $\mathbf{z}$  and the weight vectors of examples by

$$d_2 = \sum_{i=1}^n \|\mathbf{z} - \mathbf{y}_i\|^2.$$

If  $d_2 < \varepsilon_2$ ,  $Z$  is assumed to be the face of a person in the training set, and the person is recognized from the nearest neighbors of  $Z$ . Otherwise, it is assumed to be a new face.

**练习54.** Apply the above algorithm to UMist Faces in Data for MATLAB hackers (<http://www.cs.nyu.edu/~roweis/data.html>) and report recognition rates and the two thresholds you have chosen.

### 4.3 Classical Multidimensional Scaling

Classical multidimensional scaling (CMDS) is a technique that displays the structure of distance-like data as a geometrical picture. It is a member of the family of MDS methods. The input for an MDS algorithm usually is not an objective data set, but the similarities of a set of objects that may not be digitalized. The input distance matrix of CMDS is of Euclidean type. There exists an  $r$ -dimensional vector set  $\mathcal{X}$  such that the Euclidean distance matrix of  $\mathcal{X}$  is equal to the input one. The set  $\mathcal{X}$  is called a configuration of the input matrix. In the case that the dimension of the set  $\mathcal{X}$  is too high



图 4.5: The face images in a training set are chosen from the UMIST Face Database-1a. The faces are ordered from left to right, and top to down.

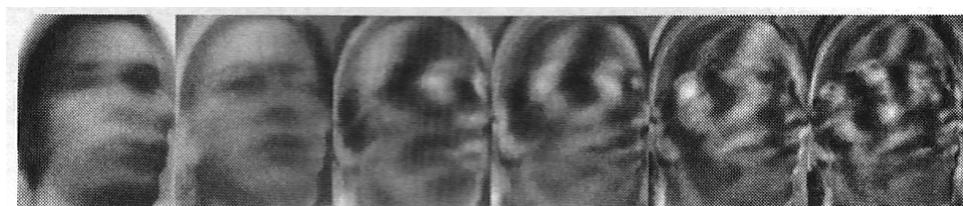


图 4.6: The average face and eigenfaces for the images in Figure 4.5. The leftmost one is the average face, and the others are eigenfaces.



图 4.7: The faces in the training set in Figure 4.5 are recovered by the average face and eigenfaces in Figure 4.6.

to be visualized, we then need to reduce the dimension of the configuration to 2 or 3 for visualization. Sometimes, a little higher dimension is acceptable. CMDS is equivalent to PCA when the input of CMDS is a data set. The data model in CMDS is linked to a complete weighted graph, in which the nodes are objects and the weights are the similarities between the objects.

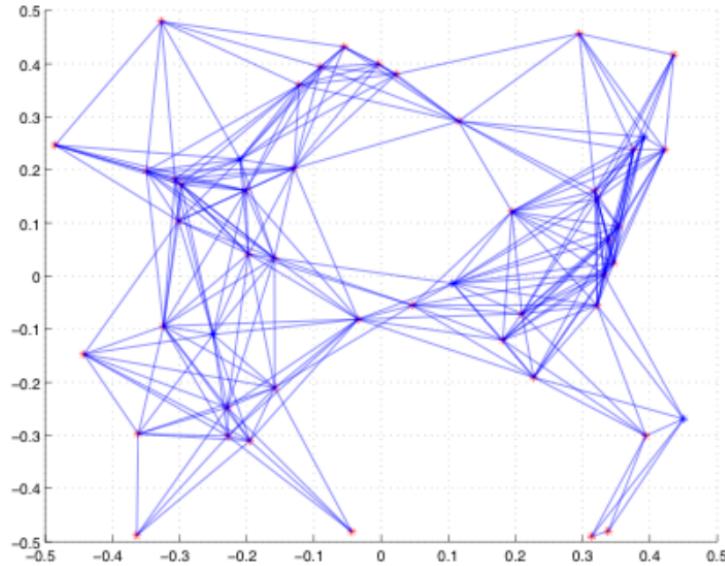


图 4.8: 50-node 2D sensor network localization.

#### 4.3.1 Data Similarities and Configuration

MDS is a set of data analysis techniques that display the structure of distance-like data as a geometrical picture. It is a powerful tool in data visualization and other data processing area. MDS has its origins in psychometrics, where it was proposed to help to understand people's judgments of the similarity of members of a set. Torgerson proposed the first MDS method and coined the term. MDS has now become a general data analysis technique used in a wide variety of fields, such as marketing, sociology, physics, political science, biology, and others.

MDS pictures the structure of a set of objects from data that approximate the distance between pairs of the objects. The data are called similarities or dissimilarities, that reflect the amount of similarity or dissimilarity between pairs of the objects.

In addition to the traditional human similarity judgment, the data can also be an “objective” similarity measure, depending on the similarity of interest. In mathematics and statistics, we often use *distance* to describe the dissimilarity between two objects, and use *covariance* to describe their similarity. Perhaps, distance measurement is more commonly used in application because it is easy to measure.

### 4.3.2 Euclidean Distance Matrices and Gram Matrices

In the classical MDS, the similarities between the original objects are assumed to be represented by the Euclidean metric in the form of a Euclidean distance matrix. The task of CMDS is to find a configuration  $\mathcal{Y} \subset \mathbb{R}^d$  such that the Euclidean distance matrix of  $\mathcal{Y}$  best approximates the given similarities.

From the view point of geometry, in a Euclidean space, the distance describes the dissimilarity of a pair of points while the inner product describes the similarity. They have a close relationship. In this section we study the relation between Euclidean distance matrix and the Gram matrix.

In the following, we represent a data set  $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subset \mathbb{R}^D$  in the matrix form  $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n) \in \mathcal{M}_{D,n}$ , where each column of  $\mathbf{X}$  is a point of  $\mathcal{X}$ . For convenience, we shall identify the data matrix  $\mathbf{X}$  with the data set  $\mathcal{X}$ .

#### 4.3.2.1 Euclidean Distance Matrices

Recall that the Euclidean distance between two vectors  $\mathbf{a} = (a_1, a_2, \dots, a_D)^T$  and  $\mathbf{x} = (x_1, x_2, \dots, x_D)^T$  in the space  $\mathbb{R}^D$  is defined by

$$d_2(\mathbf{a}, \mathbf{x}) = \|\mathbf{a} - \mathbf{x}\| = \sqrt{\sum_{i=1}^D (x_i - a_i)^2}.$$

The Euclidean inter-point distance matrix, or simply the Euclidean distance matrix, on the data set  $\mathcal{X}$  is

$$\mathbf{D} \triangleq (D_{ij}) = (d_2(\mathbf{x}_i, \mathbf{x}_j))_{i,j=1}^n, \quad (4.6)$$

which defines a metric on  $\mathcal{X}$  called Euclidean metric. We also need the Euclidean square-distance matrix

$$\mathbf{S} = (d_2^2(\mathbf{x}_i, \mathbf{x}_j))_{i,j=1}^n.$$

Both  $\mathbf{D}$  and  $\mathbf{S}$  are symmetric, and also invariant of shift and rotation. We now define the Euclidean metric in a general sense.

**定义55.** An  $n \times n$  symmetric matrix  $\mathbf{D}$  is called a Euclidean distance matrix (or Euclidean metric) if there exists an integer  $m > 0$  and a vector set  $\mathcal{Z} = \{\mathbf{z}_1, \dots, \mathbf{z}_n\} \subset \mathbb{R}^m$  such that

$$\mathbf{D} \triangleq (D_{ij}) = (d_2(\mathbf{z}_i, \mathbf{z}_j))_{i,j=1}^n. \quad (4.7)$$

The vector set  $\mathcal{Z}$  is called a configurative point set (or a configuration) of  $\mathbf{D}$ .

We often need to determine whether a matrix is a Euclidean metric directly by the properties of the matrix, without the aid of its configuration. For this purpose, we establish the relation between Euclidean metric and psd matrix.

### 4.3.2.2 Gram Matrix on Data Set

Recall that in the Euclidean space  $\mathbb{R}^D$  the inner product of two vectors  $\mathbf{x}$  and  $\mathbf{y}$  is defined by

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^D x_i y_i,$$

and the Gram matrix on the data set  $\mathcal{X}$  is defined by

$$\mathbf{G} = (G_{ij}) = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle)_{i,j=1}^n.$$

It is clear that  $\mathbf{G}$  is a positive semi-definite (psd) matrix. On the other hand, each psd matrix represents a Gram matrix of a certain data set. Indeed, if an  $n \times n$  psd matrix  $\mathbf{G}$  has rank  $m$ , then it has a Cholesky decomposition

$$\mathbf{G} = \mathbf{X}^T \mathbf{X}, \quad (4.8)$$

where  $\mathbf{X} \in \mathcal{M}_{m,n}$ . By (4.8),  $\mathbf{G}$  is the Gram matrix of the data set  $\mathbf{X}$ . Therefore, we can identify a Gram matrix with a psd matrix.

### 4.3.2.3 Relationship between Euclidean Distance Matrix and Gram Matrix

We now reveal the relation between the Gram matrix  $\mathbf{G}$  and the Euclidean distance matrix  $\mathbf{D}$  of a data set  $\mathcal{X}$ . Let  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^D$ . By the law of Cosine,

$$d_2(\mathbf{x}, \mathbf{y}) = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle + \langle \mathbf{y}, \mathbf{y} \rangle - 2\langle \mathbf{x}, \mathbf{y} \rangle},$$

which yields

$$D_{ij} = \sqrt{G_{ii} + G_{jj} - 2G_{ij}}. \quad (4.9)$$

The entries of a Gram matrix  $\mathbf{G}$  are vector inner products which are not shift invariant. In order to establish a relation between  $\mathbf{G}$  and  $\mathbf{D}$ , we shift the data  $\mathcal{X}$  by its center. Assume that  $\mathcal{X}$  resides on a  $d$ -dimensional hyperplane  $H \subset \mathbb{R}^D$ . Then the center of  $\mathcal{X} : \bar{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i$  is in  $H$ , and  $S = H - \bar{\mathbf{x}} \subset \mathbb{R}^D$  is a  $d$ -dimensional subspace parallel to  $H$ . Let the  $\bar{\mathbf{x}}$ -shift of  $\mathcal{X}$  be denoted by

$$\hat{\mathcal{X}} = \{\hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_n\},$$

where  $\hat{\mathbf{x}}_i = \mathbf{x}_i - \bar{\mathbf{x}}$ . We call  $\hat{\mathcal{X}}$  a centered data set and call the corresponding data matrix  $\hat{\mathbf{X}}$  the centered data matrix.

**定义56.** For a given data set  $\mathcal{X}$ , let  $\hat{\mathcal{X}}$  be the centered data set of  $\mathcal{X}$ . Then the Gram matrix of  $\hat{\mathcal{X}}$ :

$$\mathbf{G}^c = (\langle \hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j \rangle)_{i,j=1}^n = \hat{\mathbf{X}}^T \hat{\mathbf{X}}, \quad (4.10)$$

is called the centering Gram matrix of  $\mathcal{X}$ .

In general, given  $\mathbf{a} \in \mathbb{R}^D$ , the  $\mathbf{a}$ -shifted Gram matrix of  $\mathcal{X}$  is defined by

$$\mathbf{G}^{\mathbf{a}} = (\langle \hat{\mathbf{x}}_i - \mathbf{a}, \hat{\mathbf{x}}_j - \mathbf{a} \rangle)_{i,j=1}^n = \hat{\mathbf{X}}_{\mathbf{a}}^T \hat{\mathbf{X}}_{\mathbf{a}}. \quad (4.11)$$

It is easy to verify that the centering Gram matrix  $\mathbf{G}^c$  and the Gram matrix  $\mathbf{G}$  have the same relationship with the Euclidean distance matrix  $\mathbf{D}$  described in (4.9), namely,

$$D_{ij} = \sqrt{G_{ii}^c + G_{jj}^c - 2G_{ij}^c}. \quad (4.12)$$

Since the centering Gram matrix plays an important role in data representation, we further elaborate our discussion on this topic.

**定义57.** Write  $\mathbf{1} = (1, 1, \dots, 1)^T \in \mathbb{R}^n$ ,  $\mathbf{E} = \mathbf{1}\mathbf{1}^T$ , and let  $\mathbf{I}$  denote the  $n \times n$  identity matrix. Then the  $n \times n$  matrix  $\mathbf{H} = \mathbf{I} - \frac{1}{n}\mathbf{E}$  is called the  $n$ -centralizing matrix.

If the dimension  $n$  is understood, we shall simplify the term “ $n$ -centralizing” to “centralizing.”

**引理58.** The centralizing matrix  $\mathbf{H}$  has the following properties:

1.  $\mathbf{H}^2 = \mathbf{H}$ ,
2.  $\mathbf{1}^T \mathbf{H} = \mathbf{H} \mathbf{1} = \mathbf{0}$ ,
3.  $\mathcal{X}$  is a centered data set iff  $\mathbf{XH} = \mathbf{X}$ .
4. A psd matrix  $\mathbf{C}$  is a centering Gram matrix iff  $\mathbf{HCH} = \mathbf{C}$ .

**引理59.** Let  $\mathbf{X}$  be a data matrix and  $\mathbf{G}$  be its Gram matrix. Then the centered data set of  $\mathbf{X}$  is  $\mathbf{XH}$ , and the centering Gram matrix of  $\mathbf{X}$  is  $\mathbf{G}^c = \mathbf{HGH}$ .

In general, the centering matrix of a symmetric matrix  $\mathbf{A}$  (not necessary psd) is defined as  $\mathbf{A}^c = \mathbf{H}\mathbf{AH}$ . It is obvious that a symmetric matrix  $\mathbf{S}$  is centering iff  $\mathbf{S} = \mathbf{HSH}$ . The notion of centering symmetric matrix enables us to represent the centering Gram matrix in terms of Euclidean square-distance matrix, reducing the relation (4.9) to a very simple form.

**定理60.** Euclidean square-distance matrix  $\mathbf{S}$  and the centering Gram matrix  $\mathbf{G}^c$  of a data set  $\mathcal{X}$  have the following relation:

$$\mathbf{G}^c = -\frac{1}{2}\mathbf{S}^c. \quad (4.13)$$

Proof. It follows from Lemma 58 that  $\mathbf{G}^c$  has the property  $\sum_{i=1}^n G_{ij}^c = 0$ . Hence, the relation in (4.12) immediately yields both

$$\sum_{i=1}^n D_{ij}^2 = nG_{jj}^c + \sum_{i=1}^n G_{ii}^c, \quad j = 1, \dots, n, \quad (4.14)$$

and

$$\sum_{j=1}^n D_{ij}^2 = nG_{ii}^c + \sum_{j=1}^n G_{jj}^c, \quad i = 1, \dots, n. \quad (4.15)$$

Summing (4.14) together, we have

$$\sum_{i,j=1}^n D_{ij}^2 = n \sum_{j=1}^n G_{jj}^c + n \sum_{i=1}^n G_{ii}^c. \quad (4.16)$$

From  $\mathbf{S}^c = \mathbf{H}(\mathbf{D} \circ \mathbf{D})\mathbf{H}$ , we have

$$S_{ij}^c = D_{ij}^2 - \frac{1}{n} \sum_{i=1}^n D_{ij}^2 - \frac{1}{n} \sum_{j=1}^n D_{ij}^2 + \frac{1}{n^2} \sum_{i,j=1}^n D_{ij}^2. \quad (4.17)$$

By (4.14) and (4.15), the above can be rewritten as

$$S_{ij}^c = D_{ij}^2 - G_{jj}^c - \frac{1}{n} \sum_{i=1}^n G_{ii}^c - G_{ii}^c - \frac{1}{n} \sum_{j=1}^n G_{jj}^c + \frac{1}{n^2} \sum_{i,j=1}^n D_{ij}^2. \quad (4.18)$$

By (4.16), the above reduces to

$$S_{ij}^c = D_{ij}^2 - G_{jj}^c - G_{ii}^c = \|(\mathbf{x}_i - \mathbf{c}) - (\mathbf{x}_j - \mathbf{c})\|^2 - \|\mathbf{x}_i - \mathbf{c}\|^2 - \|\mathbf{x}_j - \mathbf{c}\|^2 \quad (4.19)$$

$$= -2 \langle \mathbf{x}_i - \mathbf{c}, \mathbf{x}_j - \mathbf{c} \rangle = -2G_{ij}^c. \quad (4.20)$$

The following is a consequence of Theorem 60 and (4.8).

**定理61.** Let  $\mathbf{A}$  be a symmetric matrix. Then

1.  $\mathbf{A}$  is a Gram matrix of a data set iff it is a psd matrix.
2.  $\mathbf{A}$  is a centering Gram matrix iff it is a centering psd matrix.
3.  $\mathbf{A}$  is a Euclidean square-distance matrix iff  $-\frac{1}{2}\mathbf{A}^c$  is a centering psd matrix.

Theorem 61 is important, for it characterizes Euclidean distance matrix and Gram matrix without using the information of the underlying data set.

### 4.3.3 A More General Theory of CMDS

**定义62.** A square matrix  $\mathbf{A}$  is called conditionally negative definite (c.n.d.), if for all  $\mathbf{x} \in \mathbb{R}^n$  such that  $\mathbf{x}^T \mathbf{1} = 0$ , we have

$$\mathbf{x}^T \mathbf{A} \mathbf{x} \leq 0.$$

It can be verified that the squared Euclidean distance matrix is conditionally negative definite.

**思考题63.** Is a c.n.d. matrix always a squared Euclidean distance matrix?

Let  $\mathbf{x}$  be a signed distribution, i.e.,  $\mathbf{x}^T \mathbf{1} = 1$ , and  $\mathbf{H}(\mathbf{x}) = \mathbf{I} - \mathbf{1}\mathbf{x}^T$  be the Householder centering matrix. Given a square matrix  $\mathbf{A}$  define  $\mathbf{A}^c(\mathbf{x}) = -\frac{1}{2}\mathbf{H}(\mathbf{x})\mathbf{A}\mathbf{H}(\mathbf{x})^T$ . Observe that  $\mathbf{H}(\mathbf{y}) = \mathbf{H}(\mathbf{y})\mathbf{H}(\mathbf{x})$  and hence  $\mathbf{A}^c(\mathbf{y}) = \mathbf{H}(\mathbf{y})\mathbf{A}^c(\mathbf{x})\mathbf{H}(\mathbf{y})^T$ . Then we have:

**定理64.** (Young-Householder-Shoenberg) For any signed distribution  $\mathbf{x}$ ,  $\mathbf{A}^c(\mathbf{x})$  is positive semi-definite if and only if  $\mathbf{A}$  is conditionally negative definite.

*Proof.* Suppose  $\mathbf{A}$  is c.n.d., then  $\mathbf{z}^T \mathbf{A} \mathbf{z} \leq 0$ , for all  $\mathbf{z}$  satisfying  $\mathbf{1}^T \mathbf{z} = 0$ . For any  $\mathbf{y}$ , we define  $\mathbf{z} = [\mathbf{H}(\mathbf{x})]^T \mathbf{y}$ , then  $\mathbf{1}^T \mathbf{z} = \mathbf{1}^T (\mathbf{I} - \mathbf{x}\mathbf{x}^T) \mathbf{y} = (\mathbf{1}^T - \mathbf{1}^T) \mathbf{y} = 0$ . So  $\mathbf{y}^T \mathbf{A}^c(\mathbf{x}) \mathbf{y} = -\frac{1}{2} \mathbf{z}^T \mathbf{A} \mathbf{z} \geq 0$ .

Suppose  $\mathbf{A}^c(\mathbf{x})$  is p.s.d. For all  $\mathbf{y}$  satisfying  $\mathbf{1}^T \mathbf{y} = 0$ , we can always rewrite  $\mathbf{y}$  as  $\mathbf{y} = [\mathbf{H}(\mathbf{x})]^T \mathbf{y}$ . Then  $\mathbf{y}^T \mathbf{A} \mathbf{y} = -2 \mathbf{y}^T \mathbf{A}^c(\mathbf{x}) \mathbf{y} \leq 0$ .  $\square$

**定理65.** Let  $\mathbf{D}$  be a squared Euclidean distance matrix. Then for any  $\lambda \geq 0$ ,  $\mathbf{B}(\lambda) = (\exp(-\lambda D_{ij}))$  is p.s.d. and  $\mathbf{A}(\lambda) = \mathbf{I} - \mathbf{B}(\lambda)$  is a squared Euclidean distance matrix.

**定义66.** A Schoenberg transform is a function  $\phi$  from  $\mathbb{R}^+$  to  $\mathbb{R}^+$  of the form:

$$\phi(d) = \int_0^{+\infty} \frac{1 - \exp(-\lambda d)}{\lambda} g(\lambda) d\lambda,$$

where  $g(\lambda) d\lambda$  is a nonnegative measure on  $[0, +\infty)$  such that

$$\int_0^{+\infty} \frac{g(\lambda)}{\lambda} d\lambda < +\infty.$$

**定理67.** (Schoenberg) Let  $\mathbf{D}$  be a squared Euclidean distance matrix. Define  $\mathbf{A} = (\phi(D_{ij}))$ . Then  $\mathbf{A}$  is a squared Euclidean distance matrix iff  $\phi$  is a Schoenberg transform.

#### 4.3.4 Description of Classical Multidimensional Scaling

##### 4.3.4.1 CMDS Method Description

Let  $\mathbf{D}$  be a given distance matrix of a set of  $n$  objects, and  $d > 0$  be an integer. The method of metric multidimensional scaling (MMDS) is a procedure for finding a configuration  $\mathcal{Y} = (\mathbf{y}_1, \dots, \mathbf{y}_n) \subset \mathbb{R}^d$ , such that a certain distance matrix associated with  $\mathcal{Y}$  is as close as possible to the matrix  $\mathbf{D}$ , i.e.,

$$d_Y(\mathbf{y}_i, \mathbf{y}_j) \approx d_{ij}, \quad \forall i, j. \quad (4.21)$$

In practice, a lost function is adopted to measure the closeness in (4.21). The CMDS adopts a Euclidean metric  $\mathbf{D}$  as its input. The following lemma plays the central role in CMDS.

**引理68.** Assume that an  $n \times n$  matrix  $\mathbf{D} = (d_{ij})$  is a Euclidean metric and  $\mathbf{S} = (d_{ij}^2)$  is the corresponding square-distance matrix. Let  $\mathbf{G}^c = -\frac{1}{2}\mathbf{S}^c$ . If the rank of  $\mathbf{G}^c$  is  $r$ , then there is an  $r$ -dimensional centered vector set  $\mathcal{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n) \subset \mathbb{R}^r$  such that

$$d_2(\mathbf{x}_i, \mathbf{x}_j) = d_{ij}, \quad \forall i, j. \quad (4.22)$$

We call  $r$  in Lemma 68 the intrinsic configuration dimension of  $\mathbf{D}$  and call  $\mathcal{X}$  the exact configuration of  $\mathbf{D}$ . If we want to use the configuration for visualization, and  $r$  is too large to meet the goal, then we seek for a low-dimensional configuration, say, a  $d$ -dimensional configuration  $\mathcal{Y}$  with  $d \ll r$ . Let both  $\mathbf{Y}$  and  $\mathbf{X}$  be treated as random vectors. Intuitively,  $\mathbf{Y}$  ought to be  $d$  principal components of  $\mathbf{X}$ . Hence, the lost function of CMDS is set to be

$$\eta(\mathcal{Y}) = \sum_{i,j=1}^n (d_{ij}^2 - d_2^2(\mathbf{y}_i, \mathbf{y}_j)), \quad s.t. \quad \mathcal{Y} = T(\mathcal{X}),$$

where  $T$  is an orthogonal projection from  $\mathbb{R}^r$  to a  $d$ -dimensional subspace  $S_d \subset \mathbb{R}^r$  and  $\mathcal{X}$  is an exact configuration of  $\mathbf{D}$ . Then the configurative point set  $\mathcal{Y}$  is the solution to

$$\mathbf{Y} = \underset{\mathbf{Y} \in \mathcal{M}_{d,n}}{\operatorname{argmin}} \eta(\mathcal{Y}), \quad s.t. \quad \mathcal{Y} = T(\mathcal{X}). \quad (4.23)$$

To simplify the minimization problem (4.23), we establish some lemmas.

**引理69.** Let  $\mathcal{Z} \subset \mathbb{R}^r$  be a given data set with corresponding Euclidean square-distance matrix  $\mathbf{S}_Z = (s_{ij})$ , where  $s_{ij} = d_2^2(\mathbf{z}_i, \mathbf{z}_j)$ , and let  $\mathbf{G}_Z^c$  be its associated centering Gram matrix. Then

$$\operatorname{tr}(\mathbf{G}_Z^c) = \frac{1}{2n} \sum_{i,j=1}^n s_{ij}. \quad (4.24)$$

**引理70.** Let  $\mathbf{D}_Z = (d_2(\mathbf{z}_i, \mathbf{z}_j))$  and  $\hat{\mathbf{Z}} = (\hat{\mathbf{z}}_1, \dots, \hat{\mathbf{z}}_n)$ . Then

$$\|\hat{\mathbf{Z}}\|_F = \frac{1}{\sqrt{2n}} \|\mathbf{D}_Z\|_F. \quad (4.25)$$

We now establish the main result in this section.

**定理71.** Let  $\mathcal{X} \subset \mathbb{R}^r$  be the configuration of  $\mathbf{D}$  in Lemma 68, where  $\mathcal{X}$  is centered, and the SVD of  $\mathbf{X}$  is:

$$\mathbf{X} = \mathbf{U}\Sigma_r\mathbf{V}^T,$$

where  $\Sigma_r = \text{diag}(\sigma_1, \dots, \sigma_r)$ ,  $\mathbf{U} = (\mathbf{u}_1, \dots, \mathbf{u}_r)$ , and  $\mathbf{V} = (\mathbf{v}_1, \dots, \mathbf{v}_r)$ . For a given  $d \leq r$ , let  $\mathbf{U}_d = (\mathbf{u}_1, \dots, \mathbf{u}_d)$  and  $\mathbf{Y} = \mathbf{U}_d^T \mathbf{X} (= \Sigma_d \mathbf{V}_d^T)$ . Then  $\mathbf{Y}$  is a solution to problem (4.23) with

$$\eta(\mathcal{Y}) = \sum_{i=d+1}^r \sigma_i^2. \quad (4.26)$$

#### 4.3.4.2 Relationship between PCA and CMDS

The motivations and data models of PCA and CMDS are different: In PCA, a data set  $\mathcal{X}$  is given and the purpose of PCA is to find its leading  $d$  principal components that preserve the variance. In CMDS, an inter-point distance matrix is given on a set of  $n$  objects and the purpose is to find a configuration point set in a low-dimensional Euclidean space that preserves the maximal similarities. However, if the samples exactly distribute on a  $d$ -dimensional subspace ( $d \leq D$ ) and the distances in CMDS are exactly Euclidean distances, then PCA coincides with CMDS as their low-dimensional representations both exactly preserve the Euclidean distances between samples. (Reason: PCA exactly projects sample points to a low-dimensional space, which preserve the distances between samples, hence forming a configuration of the high-dimensional samples.)

#### 4.3.4.3 Weighted Graphic Description of CMDS

Assume that the node set of a graph is a set of  $n$  objects  $N = [o_1, \dots, o_N]$  and the weight matrix of the graph is  $\mathbf{W}(N)$ . Then the weighted graph can be written as  $G_N = [N, \mathbf{W}(N)]$ . In CMDS, the weight matrix  $\mathbf{W}(N)$  is an  $n \times n$  Euclidean matrix, representing the similarities between the objects in a Euclidean space. Since the distances of all pairs are measured, the graph  $G_N$  is a complete one. From the viewpoint of graph, CMDS is a method to find a new graph  $G_Y = [\mathcal{Y}, \mathbf{W}(\mathcal{Y})]$ , where  $\mathcal{Y} \subset \mathbb{R}^d$  is a low-dimensional set, and  $\mathbf{W}(\mathcal{Y})$  is the best approximation of  $\mathbf{W}(N)$  in the sense that  $\mathbf{W}(\mathcal{Y})$ , as Euclidean distances among  $\mathcal{Y}$ , minimizes the lost function  $\|\mathbf{W}(N) - \mathbf{W}(\mathcal{Y})\|^2$ .

#### 4.3.4.4 CMDS Algorithm

Given distance matrix  $\mathbf{D}$ .

**Step 1. Create centering Gram matrix.** Apply the formula (4.13) to create a centering Gram matrix of the configuration: a)  $\mathbf{S} = \mathbf{D} \circ \mathbf{D}$ . b)  $\mathbf{S}^c = \mathbf{H} \mathbf{S} \mathbf{H}$  ( $\mathbf{S}_{ij}^c = \mathbf{S}_{ij} - \frac{1}{n} \sum_{i=1}^n \mathbf{S}_{ij} - \frac{1}{n} \sum_{j=1}^n \mathbf{S}_{ij} + \frac{1}{n^2} \sum_{i,j=1}^n \mathbf{S}_{ij}$ ). c)  $\mathbf{G}^c = -\frac{1}{2} \mathbf{S}^c$ .

**Step 2. Make spectral decomposition of  $\mathbf{G}^c$ .** Assume that the rank of  $\mathbf{G}^c$  is  $r$ . Let the spectral decomposition of  $\mathbf{G}^c$  be  $\mathbf{G}^c = \mathbf{U} \Lambda \mathbf{U}^T$ , where  $\mathbf{U} = (\mathbf{u}_1, \dots, \mathbf{u}_r)$  and  $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_r)$  with  $\lambda_1 \geq \dots \geq \lambda_r$ .

**Step 3. Find configuration.** Set  $\mathbf{U}_d = (\mathbf{u}_1, \dots, \mathbf{u}_d)$  and  $\Lambda_d = \text{diag}(\lambda_1, \dots, \lambda_d)$ . Then the configuration is  $\mathbf{Y} = \sqrt{\Lambda_d} \mathbf{U}_d^T$ .

**练习72.** Prove that the solution to

$$\begin{aligned} \min_{\mathbf{X} \succcurlyeq \mathbf{0}, \text{rank}(\mathbf{X}) \leq r} & \|\mathbf{X} - \mathbf{A}\|_F^2 \end{aligned}$$

is  $\mathbf{X} = \mathbf{U}_r \max(\Lambda_r, 0) (\mathbf{U}^r)^T$ , where  $\mathbf{A}$  is a symmetric matrix and  $\mathbf{U}_r$  and  $\Lambda_r$  consist of the first  $r$  eigenvectors and eigenvalues of  $\mathbf{A}$ , respectively.

**练习73.** Choose one distance for UMist Faces in Data for MATLAB hackers (<http://www.cs.nyu.edu/~roweis/data.html>), do CMDS with  $k = 2$ , and draw the points on a plane. Also report what distance you have used.

## 4.4 Random Projection

### 4.4.1 Introduction

PCA is a very important linear method for dimensionality reduction (DR). It measures data distortion globally by the Frobenius norm of the matrix of data difference. The reduced data of PCA consists of several leading eigenvectors of the covariance matrix of the data set. Hence, PCA may not preserve the local separation of the original data. To respect local properties of data in DR, we employ Lipschitz embeddings to realize DR with a high probability. Random projection (RP) does not introduce a significant distortion when the dimension and cardinality of data are both large. It randomly projects the original high-dimensional data into a lower-dimensional subspace. Because the projection costs linear computational time, the method is computationally efficient, yet produces sufficient accuracy with a high probability.

#### 4.4.2 Lipschitz Embeddings

We have introduced PCA and CMDS for DR, and proved their equivalence when the input is the data set. PCA employs the spectral decomposition of the covariance matrix if the input data set and the reduced data consists of several leading eigenvectors of the matrix. Hence, PCA is the linear method that projects the original data into a subspace, measuring data distortion globally by a quadratic sum of the Euclidean differences between the original data and the DR data. It is clear that such a notion of distortion only captures a significant global property, but does not offer any local guarantees. As a result, the distance between a pair of points in the DR data set can be arbitrarily smaller than its corresponding pair in the original data set if that is advantageous to minimizing the global distortion. For example, let  $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subset \mathbb{R}^D$  be the original (centered) data set and the singular value decomposition of the data matrix  $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)$  be

$$\mathbf{X} = \sum_{i=1}^r \sigma_i \mathbf{u}_i \mathbf{v}_i^T, \quad r \leq D.$$

Then the reduced data  $\mathcal{Y} \subset \mathbb{R}^d$  of PCA are represented by the matrix  $\mathbf{Y} = (\mathbf{u}_1, \dots, \mathbf{u}_r)^T \mathbf{X}$ . If  $d < r$ , all vectors in the subspace spanned by the set  $\{\mathbf{u}_{r+1}, \dots, \mathbf{u}_r\}$  will be mapped to zero vector, so that the distances between them cannot be preserved.

In mathematics, an embedding means a structure-preserving mapping. A dimensionality reduction method adopts a special mapping. In some applications, for example, in text document ordering, a DR method is required to preserve the separation of the objects of interest. Hence, studying the embedding that respects local properties is important. In the applications mentioned above, we need the embedding, which guaranteed that distance between all pairs are approximately maintained. In mathematics, these embeddings are called Lipschitz embeddings, or Lipschitz mappings.

**定义74.** A mapping  $f : \mathcal{X} \subset \mathbb{R}^D \rightarrow \mathbb{R}^k$  is called a Lipschitz mapping (on  $\mathcal{X}$ ) if there are two positive constants  $A$  and  $B$  such that for each pair of vectors  $\mathbf{u}, \mathbf{v} \in \mathcal{X}$ , the following holds,

$$A\|\mathbf{u} - \mathbf{v}\|^2 \leq \|f(\mathbf{u}) - f(\mathbf{v})\|^2 \leq B\|\mathbf{u} - \mathbf{v}\|^2.$$

In the above definition, the Euclidean norm is applied. But it can be replaced by other norms. Under a Lipschitz mapping, the distances between all pairs of vectors in the data set  $\mathcal{X}$  are approximately preserved.

#### 4.4.3 JL-Embeddings

Many deep and beautiful results already exist in the study of Lipschitz mappings. However, to construct Lipschitz mappings for data reduction is not easy, particularly,

in deterministic algorithms. RP proves a useful tool in the construction of Lipschitz mappings.

An RP is a linear mapping that projects the data set into a random subspace. Assume that the data set  $\mathcal{X}$  is a subset of the Euclidean subspace  $\mathbb{R}^D$ . We consider  $\mathcal{X}$  as a sample set of a  $D$ -dimensional random vector  $\mathbf{X} = (X_1, \dots, X_D)^T$ . Let  $\mathcal{R} = \{\mathbf{r}_1, \dots, \mathbf{r}_k\} \subset \mathbb{R}^D$  be a set of  $k$  ( $k < D$ ) random vectors and  $\mathbf{R} = (\mathbf{r}_1, \dots, \mathbf{r}_k)$  be the corresponding random matrix. The mapping  $f_r : \mathcal{X} \rightarrow \mathcal{Y} \subset \mathbb{R}^k$ ,  $\mathbf{Y} = \mathbf{R}^T \mathbf{X}$  is called a random projection. The relation  $\mathbf{Y} = \mathbf{R}^T \mathbf{X}$  indicates that  $\mathbf{Y}$  is a random vector sampled in a  $k$ -subspace of  $\mathbb{R}^D$ . The feasibility of RP algorithm is based on the Johnson-Lindenstrauss Lemma, which will be studied in the next section. Hence, the Lipschitz embeddings generated by RPs are often called JL-embeddings. Currently, JL-embeddings become an important part of modern algorithmic design. In this section, we shall introduce the random projection algorithms and present their justification.

RP also plays an important role in matrix decomposition. In many applications, SVD of matrices plays an important role. For example, PCA employs SVD to find principal components. When a data set is in a very high-dimensional space and the set contains a large number of points, computing SVD of the data matrix become very computationally expensive and unstable. For instance, if a data set  $\mathcal{X} \subset \mathbb{R}^D$  has  $n$  points and  $D \sim n$ , then SVD of  $\mathbf{X}$  requires  $O(n^3)$  operations and  $O(n^2)$  memory space. Hence, directly computing SVD of the data matrices often becomes unfeasible. RP maps the data into a  $k$ -dimensional subspace ( $k \ll D$ ), which “nearly” catches the required principal components. The mapping only costs  $O(n)$  if  $k \ll n$ . Hence, RP is computationally efficient and simple, yet produces sufficient accuracy with a high probability.

RP is like a kind of Monte Carlo method, while PCA is a deterministic algorithm. RP is employed to replace PCA when the deterministic algorithms become unfeasible or impossible to compute required results.

#### 4.4.4 RP Algorithms

##### 4.4.4.1 Random Matrices and Random Projection

RP is realized by a random matrix. A random variable  $r$  is denoted by  $r \sim N(0, 1)$  if its probability density function  $f_r$  is the Gaussian probability distribution  $N(0, 1)$ :

$$f_r = \frac{1}{\sqrt{2\pi}} \exp(-x^2/2).$$

It is denoted by  $r \sim U(\{-1, 1\})$  if its is uniformly distributed on  $\{-1, 1\}$ :

$$r = \begin{cases} 1, & \text{with probability } 1/2; \\ -1, & \text{with probability } 1/2; \end{cases} \quad (4.27)$$

and it is denoted by  $r \sim U(\{\pm\sqrt{3}, 0\})$  if

$$r = \begin{cases} \sqrt{3}, & \text{with probability } 1/6; \\ 0, & \text{with probability } 2/3; \\ -\sqrt{3}, & \text{with probability } 1/6; \end{cases} \quad (4.28)$$

For convenience, we shall call  $r \sim N(0, 1)$ ,  $\sim U(\{-1, 1\})$ , and  $\sim U(\{\pm\sqrt{3}, 0\})$  the random variables of Type-1, Type-2, and Type-3, respectively. It can be verified that each of these random variables has zero mean and unit variance.

A random matrix is a matrix such that all of its entries are independent and identically distributed (i.i.d.) random variables. To classify various random matrices, we introduce the following.

**定义75.** Let  $\mathbf{R} = (r_{ij})$  be a random matrix of independent variables  $r_{ij}$ . Then  $\mathbf{R}$  is called a random matrix of Type-1 (or Gaussian), Type-2, or Type-3, if  $r_{ij}$  is of Type-1, Type-2, or Type-3, respectively.

**定义76.** Let  $\mathbf{R} = (r_{ij})$  be a random matrix of a certain type. A normalized random projection  $R : \mathbb{R}^D \rightarrow \mathbb{R}^k$  is the projection that maps a  $D$ -dimensional vector  $\mathbf{u}$  to a  $k$ -dimensional vector  $\mathbf{v}$ , defined by:

$$\mathbf{v} = R(\mathbf{u}) = \frac{1}{\sqrt{k}} \mathbf{R}\mathbf{u}. \quad (4.29)$$

Since the random matrices of Types-2 and 3 are simpler than those of Type-1, they accelerate the computing speed. Note also that the random matrices of Type-3 are sparse. The random projection method provides a very simple strategy for random algorithm development. First, the data set is randomly projected into a lower dimensional space. Then, any desirable algorithm is applied to the transformed low-dimensional data set. Since RP preserves data features with high probability, this simple protocol has been followed to successfully design learning algorithms and other algorithms.

#### 4.4.4.2 Random Projection Algorithms

The RP algorithm is extremely simple. It has only two part: random matrix creation and matrix multiplication.

#### 4.4.5 Justification

In this subsection, we justify the validity of random projection for DR. Recall that RP is used to realize Lipschitz embedding. Note that, when  $k < D$ , any projection  $f : \mathbb{R}^D \rightarrow \mathbb{R}^k$  has a non-trivial null space. Therefore, it cannot be a Lipschitz embedding. Hence, Lipschitz embeddings only exist for certain subsets. On a given subset the existence of Lipschitz embeddings needs to be justified. Johnson and Lindenstrauss are pioneers in the study of this direction.

##### 4.4.5.1 Johnson and Lindenstrauss Lemma

Johnson and Lindenstrauss in their pioneer paper gave the following lemma, which reveals the dependency between the dimension  $k$  (of the space, into which data embedded) and the cardinality of the data set  $\mathcal{X}$ .

**引理77.** *Let  $\varepsilon > 0$  and integer  $n$  be given. Then for all positive integers  $k \geq k_0 = O(\varepsilon^{-2} \log n)$  and a set  $\mathcal{X}$  of  $n$  points, which are randomly selected in  $\mathbb{R}^D$ , there exists a projection  $f : \mathbb{R}^D \rightarrow \mathbb{R}^k$ , which satisfies*

$$(1 - \varepsilon)\|\mathbf{u} - \mathbf{v}\|^2 \leq \|f(\mathbf{u}) - f(\mathbf{v})\|^2 \leq (1 + \varepsilon)\|\mathbf{u} - \mathbf{v}\|^2, \quad \forall \mathbf{u}, \mathbf{v} \in \mathcal{X}. \quad (4.30)$$

The projection  $f$  in Lemma 77 is called the JL-embedding. In the original paper, the range of  $k$  is not given by an explicit formula. Various extensions of Lemma 77 have been established over the past two decades. It has also been pointed out that JL-embeddings could be realized by random projections with positive probabilities, and the explicit formula of the lower bound of  $k$  in Lemma 77 has been given.

We first introduce a lemma, which partially explains why RP can realize JL-embedding.

**引理78.** *Let  $\mathbf{R} = (r_{ij})$  be a  $k \times D$  matrix, in which all entries are i.i.d. random variables with zero mean and unit variance. Let the mapping  $f : \mathbb{R}^D \rightarrow \mathbb{R}^k$  be defined by*

$$f(\mathbf{a}) = \frac{1}{\sqrt{k}} \mathbf{R} \mathbf{a}, \quad \mathbf{a} \in \mathbb{R}^D. \quad (4.31)$$

*Then  $f(\mathbf{a})$  is a random vector in  $\mathbb{R}^k$ , of which all components are i.i.d. random variables with zero mean and variance  $\frac{1}{k}\|\mathbf{a}\|^2$ . Therefore,*

$$E(\|f(\mathbf{a})\|^2) = \|\mathbf{a}\|^2. \quad (4.32)$$

Proof. We denote the  $i$ -th row vector of  $\mathbf{R}$  by  $\mathbf{r}_i^T$  and denote the inner product  $\langle \mathbf{r}_i, \mathbf{a} \rangle$  by  $c_i$ . Then  $f(\mathbf{a}) = \mathbf{c} = (c_1, \dots, c_k)^T$ . We have

$$E(c_i) = \frac{1}{\sqrt{k}} \sum_{j=1}^D a_j E(r_{ij}) = 0,$$

$$E(c_i c_j) = \frac{1}{k} \sum_{m=1}^D \sum_{l=1}^D a_l a_m E(r_{il} r_{jm}) = 0, \quad i \neq j,$$

and

$$E(c_i^2) = E\left(\left(\sum_{j=1}^D a_j r_{ij}\right)^2\right) = \frac{1}{k} \left(\sum_{j=1}^D a_j^2 E(r_{ij}^2) + 2 \sum_{l \neq m} a_l a_m E(r_{il}) E(r_{im})\right) = \frac{1}{k} \sum_{j=1}^D a_j^2,$$

which yields that all entries of  $f(\mathbf{a})$  are i.i.d. random variables with 0 mean and variance  $\frac{1}{k} \|\mathbf{a}\|^2$ . Finally, we have

$$E(\|f(\mathbf{a})\|^2) = \sum_{i=1}^k E(c_i^2) = \|\mathbf{a}\|^2.$$

The lemma shows that the RP  $f$  has the properties  $E(\|f(\mathbf{a})\|^2) = \|\mathbf{a}\|^2$  for any independent matrix  $(r_{ij})$  with  $E(r_{ij}) = 0$  and  $\text{Var}(r_{ij}) = 1$ . Since the random matrices of Type 1-3 have such properties, they are good candidates for the construction of JL-embedding.

From Lemma 78, we have:

**推论79.** Let  $f$  be the mapping defined in Lemma 78 and  $\mathbf{a} \in \mathbb{R}^D$  is a unit vector. Then  $\sqrt{k}f(\mathbf{a})$  is a  $k$ -dimensional random vector whose entries are i.i.d. random variables with zero mean and unit variance.

Note that  $\sqrt{k}f(\mathbf{a}) = \mathbf{R}\mathbf{a}$ . The corollary claims that if a  $k \times D$  random matrix  $\mathbf{R}$  whose entries are i.i.d. random variables with zero mean and unit variance, and  $\mathbf{a} \in \mathbb{R}^D$  is a unit vector, then the entries of the random vector  $\mathbf{R}\mathbf{a}$  is also i.i.d. with zero mean and unit variance.

#### 4.4.5.2 Random Projection Based on Gaussian Distribution

We now introduce the result proved by Fasgupa and Gupta, which states that if the random matrix in the RP (4.29) is of Gaussian type, then the RP realizes a JL-embedding for a certain dimension  $k$ .

**定理80.** For any  $0 < \varepsilon < 1$  and any integer  $n > 0$ , let  $k$  be a positive integer that

$$k \geq 4(\varepsilon^2/2 - \varepsilon^3/3)^{-1} \ln n. \quad (4.33)$$

Then for any set  $\mathcal{V} \subset \mathbb{R}^D$  of  $n$  points, there is a linear mapping  $f : \mathbb{R}^D \rightarrow \mathbb{R}^k$  such that for all  $\mathbf{u}, \mathbf{v} \in \mathcal{V}$ ,

$$(1 - \varepsilon)\|\mathbf{u} - \mathbf{v}\|^2 \leq \|f(\mathbf{u}) - f(\mathbf{v})\|^2 \leq (1 + \varepsilon)\|\mathbf{u} - \mathbf{v}\|^2. \quad (4.34)$$

Furthermore, this mapping can be found in randomized polynomial time.

**备注81.** An algorithm has randomized polynomial time if it runs in polynomial time in the input size and, if the correct answer is NO, it always returns NO, while if the correct answer is YES, then it returns YES with a positive (constant) probability.

To prove this theorem, we need:

**引理82.** Let  $\mathbf{R}$  be a  $k \times D$  ( $k \leq D$ ) random matrix of Gaussian type and  $\mathbf{a} \in \mathbb{R}^D$  be a unit vector. Let  $\mathbf{y} = \mathbf{R}\mathbf{a}$  and  $\beta > 1$ . Then

$$\Pr(\|\mathbf{y}\|^2 \leq k/\beta) < \exp\left(\frac{k}{2}(1 - 1/\beta - \ln \beta)\right), \quad (4.35)$$

$$\Pr(\|\mathbf{y}\|^2 \geq k\beta) < \exp\left(\frac{k}{2}(1 - \beta + \ln \beta)\right). \quad (4.36)$$

We slightly deviate for concentration of measure inequalities.

**定理83.** (Markov不等式) 如果  $X$  是非负随机变量, 则

$$P(X > t) \leq \frac{E(X)}{t}, \quad \forall t > 0. \quad (4.37)$$

*Proof.*

$$\begin{aligned} P(X > t) &= \int_{x>t} p(x)dx \\ &\leq \int_{x>t} (x/t)p(x)dx \\ &= \frac{1}{t} \int_{x>t} xp(x)dx \\ &\leq \frac{1}{t} \int xp(x)dx \\ &= \frac{1}{t} E(X) \end{aligned}$$

□

**定理84.** (Chebyshev不等式)

$$P(|X - E(X)| > t) \leq \frac{\text{Var}(X)}{t^2}, \quad \forall t > 0. \quad (4.38)$$

*Proof.* 由Markov不等式,

$$\begin{aligned} P(|X - E(X)| > t) &= P((X - E(X))^2 > t^2) \\ &\leq \frac{1}{t^2} E[(X - E(X))^2] \\ &= \frac{1}{t^2} \text{Var}(X) \end{aligned}$$

□

**定理85.** (*Chernoff技巧*)

$$P(X > t) \leq \inf_{s>0} \frac{E(e^{sX})}{e^{st}}. \quad (4.39)$$

*Proof.* 任取  $s > 0$ , 由Markov不等式,

$$\begin{aligned} P(X > t) &= P(e^{sX} > e^{st}) \\ &\leq \frac{E(e^{sX})}{e^{st}}. \end{aligned}$$

□

Now we prove (4.35).

*Proof.* For any  $h > 0$ ,

$$Pr(\|\mathbf{y}\|^2 \leq k/\beta) = Pr(\exp(-h\|\mathbf{y}\|^2) \geq \exp(-hk/\beta)) \leq E \exp(-h\|\mathbf{y}\|^2) \exp(hk/\beta). \quad (4.40)$$

Note that  $y_i = \sum_{j=1}^D a_j r_{ij}$ , where  $r_{ij} \sim N(0, 1)$ . Hence,  $y_i$  is distributed normally. By Corollary 79,  $y_i \sim N(0, 1)$ ,  $1 \leq i \leq k$ . So we obtain

$$E(\exp(-hy_i^2)) = \int_{-\infty}^{\infty} \exp(-hy_i^2) \frac{1}{\sqrt{2\pi}} \exp(-y_i^2/2) dy_i = \frac{1}{\sqrt{1+2h}}, \quad (4.41)$$

which yields

$$E \exp(-h\|\mathbf{y}\|^2) \exp(hk/\beta) = (1+2h)^{-k/2} \exp(hk/\beta) \equiv g(h). \quad (4.42)$$

The function  $g(h)$  has its minimum at  $h^* = (\beta - 1)/2$  and the minimum value is

$$g(h^*) = \beta^{-k/2} \exp(k(1 - 1/\beta)/2) = \exp\left(\frac{k}{2}(1 - 1/\beta - \ln \beta)\right).$$

Hence (4.35) holds. □

**练习86.** Prove (4.36).

Now we return to the proof of Theorem 80.

*Proof.* Let  $\mathbf{R}$  be a  $k \times D$  random matrix whose entries are i.i.d. random variables  $\sim N(0, 1)$ . We define  $f : \mathbb{R}^D \rightarrow \mathbb{R}^k$  as in (4.31). For each pair of  $\mathbf{u}$  and  $\mathbf{v}$  in  $\mathcal{V}$  ( $\mathbf{u} \neq \mathbf{v}$ ), we set

$$\mathbf{a} = (\mathbf{u} - \mathbf{v}) / \|\mathbf{u} - \mathbf{v}\|, \mathbf{z} = f(\mathbf{a}), \mathbf{y} = \sqrt{k}\mathbf{z}.$$

Then

$$Pr\left(\frac{\|f(\mathbf{u}) - f(\mathbf{v})\|^2}{\|\mathbf{u} - \mathbf{v}\|^2} \leq 1 - \varepsilon\right) = Pr(\|\mathbf{z}\|^2 \leq 1 - \varepsilon) = Pr(\|\mathbf{y}\|^2 \leq k(1 - \varepsilon)).$$

Similarly,

$$Pr\left(\frac{\|f(\mathbf{u}) - f(\mathbf{v})\|^2}{\|\mathbf{u} - \mathbf{v}\|^2} \geq 1 + \varepsilon\right) = Pr(\|\mathbf{y}\|^2 \geq k(1 + \varepsilon)).$$

Applying Lemma 82 to  $\mathbf{y}$  with  $\beta = 1/(1 - \varepsilon)$ , we have

$$\begin{aligned} Pr(\|\mathbf{y}\|^2 \leq k(1 - \varepsilon)) &< \exp\{k[1 - (1 - \varepsilon) + \ln(1 - \varepsilon)]/2\} \\ &\leq \exp\{k[\varepsilon - (\varepsilon - \varepsilon^2/2)]/2\} \\ &\leq \exp(-k\varepsilon^2/4) \leq \exp(-2\ln n) = \frac{1}{n^2}, \end{aligned}$$

where the inequality  $\ln(1 - \varepsilon) \leq (-\varepsilon + \varepsilon^2/2)$  is used for getting the second line, and the condition (4.33) is applied to deriving the inequality in the last line. Similarly, we have

$$\begin{aligned} Pr(\|\mathbf{y}\|^2 \geq k(1 + \varepsilon)) &< \exp\{k[1 - (1 + \varepsilon) + \ln(1 + \varepsilon)]/2\} \\ &\leq \exp\{k[-\varepsilon + (\varepsilon - \varepsilon^2/2 + \varepsilon^3/3)]/2\} \\ &\leq \exp(-k(\varepsilon^2/2 - \varepsilon^3/3)/2) \leq \exp(-2\ln n) = \frac{1}{n^2}. \end{aligned}$$

Therefore, for each pair  $\mathbf{u}$  and  $\mathbf{v}$  in  $\mathcal{V}$ ,

$$Pr\left(\frac{\|f(\mathbf{u}) - f(\mathbf{v})\|^2}{\|\mathbf{u} - \mathbf{v}\|^2} \notin [1 - \varepsilon, 1 + \varepsilon]\right) < \frac{2}{n^2}.$$

Since the cardinality of  $\mathcal{V}$  is  $n$ , there are total  $n(n - 1)/2$  pairs. The probability of the event that at least one pair does not satisfy (4.34) is less than  $1 - \frac{n(n-1)}{2} \frac{2}{n^2} = \frac{1}{n} > 0$ . The theorem is proved.  $\square$

From the proof of Theorem 80, we claim that a random projection with i.i.d.  $N(0, 1)$  entries generates a JL-embedding. This is:

**推论87.** Let  $\mathcal{X} = \{\mathbf{a}_1, \dots, \mathbf{x}_n\} \subset \mathbb{R}^D$  be a given data set,  $k$  be an integer satisfying (4.33), and  $\mathbf{R}$  be a  $k \times D$  random matrix of Type-1. Let  $\mathbf{Y} = \frac{1}{\sqrt{k}} \mathbf{R} \mathbf{X}$ . Then for each pair  $\mathbf{x}_i$  and  $\mathbf{x}_j$  in  $\mathcal{X}$ , the following inequalities

$$(1 - \varepsilon)\|\mathbf{x}_i - \mathbf{x}_j\|^2 \leq \|\mathbf{y}_i - \mathbf{y}_j\|^2 \leq (1 + \varepsilon)\|\mathbf{x}_i - \mathbf{x}_j\|^2 \quad (4.43)$$

hold with the probability being at least  $1 - 2n^{-2}$ .

#### 4.4.5.3 Random Projection Based on Types 2 and 3

In this subsection, we prove that JL-embeddings can also be realized by the RP of Types 2 and 3.

**定理88.** Let  $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subset \mathbb{R}^D$  be an arbitrary data set and  $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)$  be corresponding  $D \times n$  matrix. For a given  $0 < \varepsilon < 1$ , let  $k$  be a positive integer that satisfies (4.33) and  $\mathbf{R}$  be a  $k \times D$  random matrix of Type-2 or Type-3. Let  $f : \mathbb{R}^D \rightarrow \mathbb{R}^k$  be the mapping defined by (4.31),  $\mathbf{y}_i = f(\mathbf{x}_i)$ ,  $1 \leq i \leq n$ . Then the probability for  $f$  to satisfy the Lipschitz condition on  $\mathcal{X}$ :

$$(1 - \varepsilon)\|\mathbf{x}_i - \mathbf{x}_j\|^2 \leq \|\mathbf{y}_i - \mathbf{y}_j\|^2 \leq (1 + \varepsilon)\|\mathbf{x}_i - \mathbf{x}_j\|^2, \quad \forall \mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}, \quad (4.44)$$

is at least  $1 - n^{-\beta}$ , where

$$\beta = \frac{k(\varepsilon^2/4 - \varepsilon^3/6)}{\ln n} - 2 > 0.$$

The random variables of Types 2 or 3 are different from Type-1 because the sum and difference of two independent random variables of Type-1 are still of Type-1, but this is not true for Types 2 and 3.

#### 4.4.6 Applications of Random Projections

In this section, we introduce some applications of RP methods. RP is a general data reduction method, in which the original high-dimensional data is simply projected into a low-dimensional subspace without using eigen-decomposition of matrices. So it runs extremely fast and stable. Besides, in contrast to other linear methods, such as PCA, RP does not use any optimization criteria. Therefore, it is data-independent. Moreover, it is a computationally simple and efficient method that approximately preserves pairwise distances between the points of a set without introducing significant distortion. Hence it is widely used in many DR applications that need to preserve the local structure of the data and require instantaneous results.

##### 4.4.6.1 Face Recognition Experiments with Random Projection

In [53], the authors evaluated RP for face recognition under various conditions and assumptions. They compared the performance of RP to PCA. The results indicate that RP can be compared quite favorably with PCA, while RP is simpler, more computationally efficient, and data-independent. To apply the RP method in face recognition, a  $d \times D$  random matrix  $\mathbf{R}$  is created and the rows of  $\mathbf{R}$  are orthogonalized so that they form an orthogonal system in  $\mathbb{R}^D$ . The main reason to use the orthogonalization is to better preserve the similarities between the original vectors. If the dimension of the original data is high enough, due to the fact that in high-dimensional spaces, there exists a much larger number of almost orthogonal vectors, then the rows of the created random matrix are nearly orthogonal. Hence, the orthogonalization step can be skipped.

Since RP method is data-independent, it does not require to update the projection when the new faces are added to the database or old faces are deleted from it. By contrast, after the database is changed, most deterministic DR techniques require recomputing the eigenfaces and re-projecting the faces to maintain their optimality. As we have mentioned, RP has much lower computational expense comparing to PCA. Performing an RP with an  $n \times m$  random matrix of Type-1 needs  $O(nm^2)$  time of operations, and with an  $n \times m$  random matrix of Types 2 or 3 needs  $O(nm)$ , while PCA needs  $O(n^2m)$ . When  $m \ll n$ , RP costs only  $O(n)$ , while PCA costs  $O(n^2)$ .

RP has several other properties that benefit face recognition. For example, when the original data form highly eccentric Gaussian clusters, RP preserves the clustering effect down in lower dimensions and produces more spherical cluster, which could improve recognition performance. In addition, it has been shown that for moderate dimensions, the inner product between the mapped vectors follows closely those of the original vectors, that is, random mappings do not introduce distortion in the data and preserve similarities approximately.

The RP algorithm for face recognition is very similar to the eigenface method introduced in Section 4.2.6.2, except that the projection matrix in RP is computed randomly. Assume that the resolution of the face images in the database  $\mathcal{X}$  is  $D = r \times c$ , where  $r \times c$  is the image dimension, and there are  $n$  faces in the database,  $|\mathcal{X}| = n$ . Assume also that the faces are already formatted as a  $D \times n$  matrix  $\mathbf{X}$ , where each column represents a face. Then the RP face projection algorithm consists of the following steps.

1. **Compute the average face.** This simple step produces an average face  $\bar{\mathbf{x}}$  of all faces in  $\mathcal{X}$ .
2. **Centralize the faces in the database.** This step produces a centered data:  $\hat{\mathbf{x}}_i = \mathbf{x}_i - \bar{\mathbf{x}}$ . We denote the new data matrix by  $\hat{\mathbf{X}}$ .
3. **Generate the random matrix.** This step generates an  $d \times D$  random matrix  $\mathbf{R}$  of Types 1, 2, or 3. Usually,  $d \ll D$ . If  $D$  is very large, then normalize  $\mathbf{R}$  to  $\mathbf{P} = \frac{1}{\sqrt{d}}\mathbf{R}$ . Otherwise, if  $D$  is moderate, orthogonalize its rows to produce a matrix  $\mathbf{P} \in \mathcal{O}_{d,D}$ .
4. **Project the centered faces into the random subspace.** In this step, a reduced set of faces is computed by  $\mathbf{Y} = \mathbf{P}\hat{\mathbf{X}}$ , in which each column of  $\mathbf{Y}$  is the randomly projected face.

Some experiments and results of RP for face recognition can be found in [53], where the authors used three popular face databases in their study: ORL (<http://www.face-rec.org/>

[org/databases/](#)), CVL (<http://www.lrv.fri.uni-lj.si/facedb.html>), and AR in ATT Laboratories Cambridge. In each experiment, the database under experimentation is divided into three subsets: training, gallery, and test. The training set was used only by PCA to compute the eigenspace for the comparison purpose. The detailed reports of the experiments and results are referred to [53].

### 4.4.7 RP Applications to Image and Text Data

The authors of [53] also applied RP method to the DR of high-dimensional image and text data. In their experiments, the image data were chosen from the monochrome images of natural sciences, in which each image is presented as a matrix of pixel brightness, and in the text data each document is presented as a vector whose  $i$ -th element indicates (some function of) the frequency of the  $i$ -th vocabulary term in a given dictionary. Document data are often highly sparse or peaked: only some terms from the vocabulary are presented in one document, and most entries of the document vector are zeros. Also, document data has a nonsymmetric, positively skewed distribution. In the experiments, RP method is compared to other well-known DR methods, including some nonlinear ones. The experimental results show that despite the computational simplicity, random projection does not introduce a significant distortion in the DR processing, working well in these two very different application areas. They also test RP method on images corrupted by noise. The results confirm that the insensitivity of random projection with respect to impulse noise. Thus, RP proves a promising alternative to some existing methods in noise reduction.

**练习89.** *Apply random projection to UMist Faces in Data for MATLAB hackers (<http://www.cs.nyu.edu/~rweis/data.html>), use the same recognition algorithm as eigenfaces (reorthogonalized  $\mathbf{R}$  replacing  $\mathbf{F}^T$  therein), and report recognition rates.*



## 第五章 非线性降维技术

本章内容主要取自[133]。测试数据集：Data for MATLAB hackers (<http://www.cs.nyu.edu/~roweis/data.html>)、UCI Machine Learning Repository (<http://archive.ics.uci.edu/ml/>)

线性降维假设了数据是分布在线性子空间里的，在实践中这个假设经常不成立，因此要考虑非线性降维。非线性降维的方法很多。除了Kernel PCA外，其他方法基本都是基于流形学习的算法。

将介绍Kernel PCA、Isomaps、LLE、LTSA、Laplacian Eigenmaps、Diffusion Maps、MVU等。最后视情况介绍本人的部分工作：Linear Laplacian Discrimination、Laplacian PCA、Semi-Riemannian manifold based feature extraction.

### 5.1 KPCA[15]

Kernel方法是把数据映射到高维空间中，利用高维空间数据容易变得线性可分的性质。

假设映射为 $\Phi : \mathbb{R}^m \rightarrow \mathbb{R}^M$ ，原始数据已经去除均值。则原始数据被映射为 $\mathbf{X}_\Phi = (\Phi(\mathbf{x}_1), \dots, \Phi(\mathbf{x}_k))$ 。先假定映射后样本的均值 $\bar{\mathbf{x}}_\Phi = \frac{1}{k} \sum_i \Phi(\mathbf{x}_i)$ 仍然为 $\mathbf{0}$ 。则经验协方差矩阵变成 $\Sigma_\Phi = \frac{1}{k-1} \sum_i \Phi(\mathbf{x}_i) \Phi(\mathbf{x}_i)^T$ 。假设 $(\lambda, \mathbf{v})$ 是 $\Sigma_\Phi$ 的特征对，则

$$\frac{1}{k-1} \sum_i \Phi(\mathbf{x}_i) \Phi(\mathbf{x}_i)^T \mathbf{v} = \lambda \mathbf{v}. \quad (5.1)$$

于是我们看到 $\mathbf{v}$ 必须是 $\Phi(\mathbf{x}_i)$ 的线性组合。假设

$$\mathbf{v} = \sum_{i=1}^k \alpha_i \Phi(\mathbf{x}_i).$$

代入(5.1)，两边再乘以 $\Phi(\mathbf{x}_l)^T$ 得

$$\frac{1}{k-1} \sum_i \Phi(\mathbf{x}_l)^T \Phi(\mathbf{x}_i) \sum_{j=1}^k \alpha_j \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j) = \lambda \sum_{i=1}^k \alpha_i \Phi(\mathbf{x}_l)^T \Phi(\mathbf{x}_i). \quad (5.2)$$

于是我们只需要知道 $\Phi(\mathbf{x})^T \Phi(\mathbf{y})$ ，而不需要知道 $\Phi$ 的具体表达式。令 $k(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x})^T \Phi(\mathbf{y})$ 为核函数，则上式可写成

$$\frac{1}{k-1} \sum_i k(\mathbf{x}_l, \mathbf{x}_i) \sum_{j=1}^k \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) = \lambda \sum_{i=1}^k \alpha_i k(\mathbf{x}_l, \mathbf{x}_i). \quad (5.3)$$

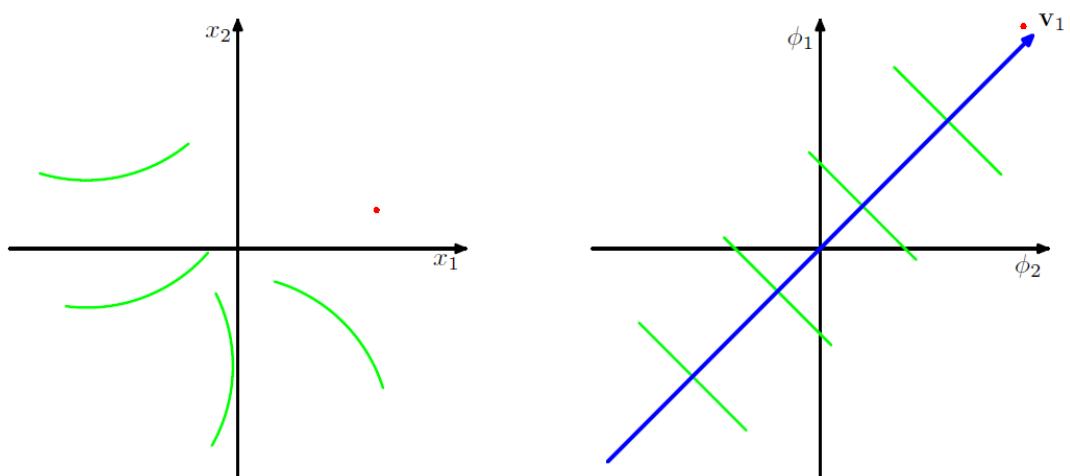


图 5.1: Schematic illustration of kernel PCA. A data set in the original data space (left-hand plot) is projected by a nonlinear transformation  $\phi(\mathbf{x})$  into a feature space (right-hand plot). By performing PCA in the feature space, we obtain the principal components, of which the first is shown in blue and is denoted by the vector  $\mathbf{v}_1$ . The green lines in feature space indicate the linear projections onto the first principal component, which correspond to nonlinear projections in the original data space. Note that in general it is not possible to represent the nonlinear principal component by a vector in  $\mathbf{x}$  space.

写成矩阵形式是

$$\mathbf{K}^2\alpha = (k-1)\lambda \mathbf{K}\alpha. \quad (5.4)$$

所以 $\alpha$ 可以作为

$$\mathbf{K}\alpha = (k-1)\lambda \alpha \quad (5.5)$$

的解，即 $\mathbf{K}$ 的特征向量。由于 $\|\mathbf{v}\| = 1$ ，所以

$$\sum_{i,j=1}^k \alpha_i \alpha_j \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j) = 1. \quad (5.6)$$

写成矩阵形式是

$$\alpha^T \mathbf{K} \alpha = 1. \quad (5.7)$$

由(5.5)，

$$\|\alpha\| = ((k-1)\lambda)^{-\frac{1}{2}}.$$

得到特征向量 $\mathbf{v}_i = \sum_{j=1}^k \alpha_{ij} \Phi(\mathbf{x}_j)$ 后， $\mathbf{x}$ 降维后的样本 $\mathbf{y}$ 在 $\mathbf{v}_i$ 上的分量为

$$y_i = \Phi(\mathbf{x})^T \mathbf{v}_i = \sum_{j=1}^k \alpha_{ij} \Phi(\mathbf{x})^T \Phi(\mathbf{x}_j) = \sum_{j=1}^k \alpha_{ij} k(\mathbf{x}, \mathbf{x}_j). \quad (5.8)$$

所以自始至终，我们都不需要知道 $\Phi$ 的具体表达式，而只需要核函数 $k$ 就行了。这称为kernel技巧。

当映射后样本的均值不为0时，我们需要使用核函数 $\tilde{k}(\mathbf{x}, \mathbf{y}) = (\Phi(\mathbf{x}) - \bar{\mathbf{x}}_\Phi)^T (\Phi(\mathbf{y}) - \bar{\mathbf{x}}_\Phi)$ 。把 $\bar{\mathbf{x}}_\Phi$ 的表达式代入，得

$$\tilde{\mathbf{K}} = \mathbf{K} - \frac{1}{k} \mathbf{E} \mathbf{K} - \frac{1}{k} \mathbf{K} \mathbf{E} + \frac{1}{k^2} \mathbf{E} \mathbf{K} \mathbf{E} \quad (5.9)$$

$$= \mathbf{K} - \frac{1}{k} \mathbf{E} \mathbf{K} - \frac{1}{k} \mathbf{K} \mathbf{E} + \frac{1}{k^2} (\mathbf{1}^T \mathbf{K} \mathbf{1}) \mathbf{E} \quad (5.10)$$

$$= \mathbf{K}^c, \quad (5.11)$$

其中 $\mathbf{E} = \mathbf{1} \mathbf{1}^T$ 是全1矩阵， $\mathbf{K}^c$ 是 $\mathbf{K}$ 的中心化矩阵。

核函数理论的基本定理是[131]：

**定理90.** (Mercer定理) 对称 $L_2$ 可积函数 $k(\mathbf{x}, \mathbf{y})$ 能以正的系数 $a_i > 0$ 展开成

$$k(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{+\infty} a_i \psi_i(\mathbf{x}) \psi_i(\mathbf{y})$$

的充分必要条件是对任意的非零 $L_2$ 可积函数 $g$ 有

$$\int g(\mathbf{x}) g(\mathbf{y}) k(\mathbf{x}, \mathbf{y}) d\mathbf{x} d\mathbf{y} > 0.$$

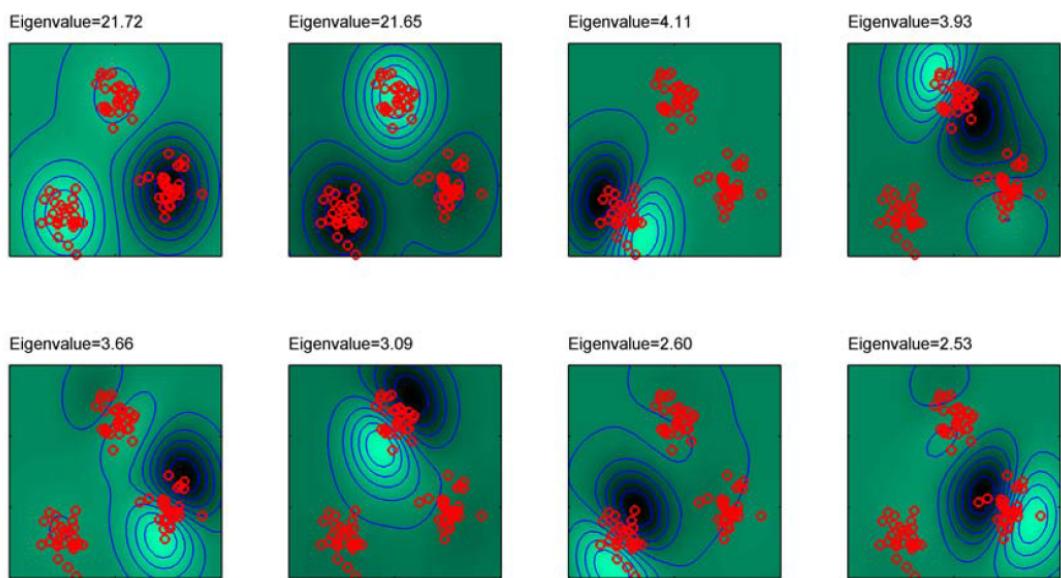


图 5.2: Example of kernel PCA, with a Gaussian kernel applied to a synthetic data set in two dimensions, showing the first eight eigenfunctions along with their eigenvalues. The contours are lines along which the projection onto the corresponding principal component is constant. Note how the first two eigenvectors separate the three clusters, the next three eigenvectors split each of the cluster into halves, and the following three eigenvectors again split the clusters into halves along directions orthogonal to the previous splits.

常用的核函数有：

1. Polynomial Kernel:  $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y} + c)^r$ ,  $c > 0$ ,  $r \geq 1$  is an integer.
2. Gaussian Kernel:  $k(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2/(2\sigma^2))$ .
3. Sigmoidal Kernel:  $k(\mathbf{x}, \mathbf{y}) = \exp(ax^T \mathbf{y} + b)$ ,  $a > 0$ ,  $b > 0$ .

Techniques of Constructing New Kernels: Given two valid kernels  $k_1$  and  $k_2$ , the following functions are also valid kernels:

$$k(\mathbf{x}, \mathbf{y}) = f(\mathbf{x})k_1(\mathbf{x}, \mathbf{y})f(\mathbf{y}), \quad (5.12)$$

$$k(\mathbf{x}, \mathbf{y}) = q(k_1(\mathbf{x}, \mathbf{y})), \quad (5.13)$$

$$k(\mathbf{x}, \mathbf{y}) = \exp(k_1(\mathbf{x}, \mathbf{y})), \quad (5.14)$$

$$k(\mathbf{x}, \mathbf{y}) = c_1 k_1(\mathbf{x}, \mathbf{y}) + c_2 k_2(\mathbf{x}, \mathbf{y}), \quad (5.15)$$

$$k(\mathbf{x}, \mathbf{y}) = k_1(\mathbf{x}, \mathbf{y})k_2(\mathbf{x}, \mathbf{y}), \quad (5.16)$$

$$k(\mathbf{x}, \mathbf{y}) = k_a(\mathbf{x}_a, \mathbf{y}_a) + k_b(\mathbf{x}_b, \mathbf{y}_b), \quad (5.17)$$

$$k(\mathbf{x}, \mathbf{y}) = k_a(\mathbf{x}_a, \mathbf{y}_a)k_b(\mathbf{x}_b, \mathbf{y}_b), \quad (5.18)$$

where  $f$  is any function,  $q$  is a polynomial with nonnegative coefficients,  $c_1 > 0$ ,  $c_2 > 0$ ,  $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$ , and  $k_a$  and  $k_b$  are valid kernels in their respective spaces.

**练习91.** Prove that the function in (5.14) is indeed a kernel.

**练习92.** Use Data for MATLAB hackers (<http://www.cs.nyu.edu/~roweis/data.html>) and the polynomial kernel with  $c = 1$  and  $r = 1, \dots, 5$  to repeat the experiment in Chapter PCA and see whether using KPCA can improve the recognition rate and which  $r$  is the best choice.

## 5.2 Isomaps

Isomap unfolds a manifold by keeping the geodesic metric on the original data set.

### 5.2.1 Description of Isomaps

In the previous part we have introduced the linear DR methods, which works effectively if the points of the observed data concentrate around a hyperplane. However, if the points concentrate around a nonlinear manifold, the linear DR methods do not work well because the geometry of the data is now characterized by its underlying manifold, not a subspace. In this case, the dissimilarities between the points of these data have to be

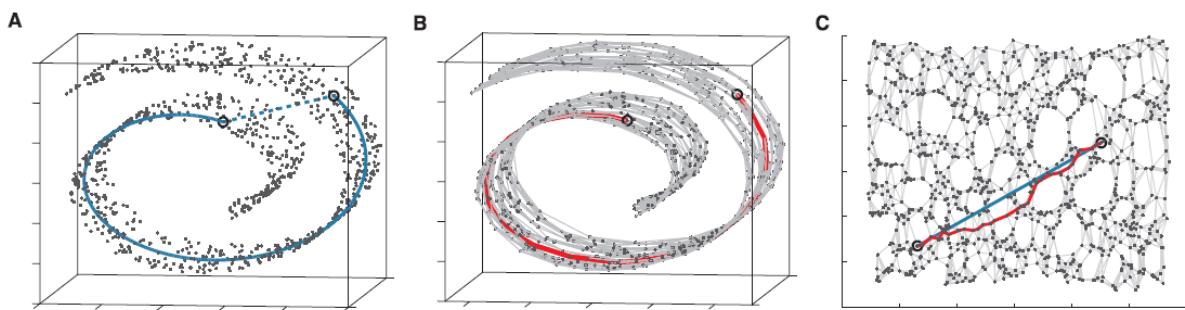


图 5.3: The “Swiss roll” data set, illustrating how Isomap exploits geodesic paths for nonlinear dimensionality reduction. (A) For two arbitrary points (circled) on a nonlinear manifold, their Euclidean distance in the high dimensional input space (length of dashed line) may not accurately reflect their intrinsic similarity, as measured by geodesic distance along the low-dimensional manifold (length of solid curve). (B) The neighborhood graph  $G$  constructed in step one of Isomap (with  $K = 7$  and  $N = 1000$  data points) allows an approximation (red segments) to the true geodesic path to be computed efficiently in step two, as the shortest path in  $G$ . (C) The two-dimensional embedding recovered by Isomap in step three, which best preserves the shortest path distances in the neighborhood graph (overlaid). Straight lines in the embedding (blue) now represent simpler and cleaner approximations to the true geodesic paths than do the corresponding graph paths (red).

measured by a non-Euclidean metric, for instance, the geodesic metric on the underlying manifold. The following example illustrated the necessity of non-Euclidean metric. Let  $\varepsilon > 0$  be very small. Let  $\mathbf{x} = (\cos \varepsilon, \sin \varepsilon)^T$  and  $\mathbf{y} = (\cos(2\pi - \varepsilon), \sin(2\pi - \varepsilon))^T$  be two points on observed data, inheriting the geometric structure from the circular arc  $\Gamma \subset \mathbb{R}^2$ :

$$\Gamma = \{(x_1, x_2)^T \in \mathbb{R}^2 | x_1 = \cos t, x_2 = \sin t, \varepsilon \leq t \leq 2\pi - \varepsilon\}.$$

Then the Euclidean distance between  $\mathbf{x}$  and  $\mathbf{y}$  is  $2 \sin \varepsilon \approx 2\varepsilon$ , while the geodesic distance is about  $2(\pi - \varepsilon)$ . To describe the similarities of points on  $\Gamma$ , we have to use the geodetic metric, not the Euclidean one.

In general, nonlinear DR methods measure the dissimilarity of points with the aid of neighborhood structure on the data. If two points are not in the same neighborhood, they are considered dissimilar. Then a metric will be defined on the data to measure the dissimilarities described by the neighborhood structure. The defined metric now does not preserve the Euclidean metric in global, but only (approximately) preserves the Euclidean metric in a neighborhood. The local similarity preservation meets the requirement of many data processes very well. Different nonlinear DR methods adopt different metrics for measuring dissimilarity/similarity. In Isomaps, the dissimilarity is measured by the geodesic metric of the underlying manifold. In practice, since the observed data are finite sets, to measure the true geodesic distance between the data points is infeasible. Hence, technically we use the graph distance to approximate the geodesic one.

We call a mapping an isometric mapping if it preserves the distance defined on the data set. Isomap DR method is a distance preserving method such that the Euclidean metric on the new data is equal to the geodesic metric on the original data. Isomap is an abbreviation of *isometric mapping*. It was first introduced by Tenenbaum, de Silva, and Langford [123], and is now widely used in many applications.

Mathematically, Isomap is based on the following consideration. Assume that an observed data set  $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subset \mathbb{R}^D$  lies on a  $d$ -dimensional Riemannian manifold  $M \subset \mathbb{R}^D$ , where the dimension  $d \ll D$ . We denote the geodesic distance on  $M$  by  $D_M$ , then there exists a coordinate mapping  $f : M \rightarrow \mathbb{R}^d$ ,  $f(\mathbf{x}_i) = \mathbf{y}_i$ , which preserves the geodesic distance:

$$d_2(f(\mathbf{x}), f(\mathbf{z})) = d_M(\mathbf{x}, \mathbf{z}), \quad \forall \mathbf{x}, \mathbf{z} \in M.$$

Particularly, we have

$$d_2(f(\mathbf{y}_i), f(\mathbf{y}_j)) = d_M(\mathbf{x}_i, \mathbf{x}_j), \quad \forall 1 \leq i, j \leq n. \quad (5.19)$$

Since the similarity on the data set  $\mathcal{Y} = \{\mathbf{y}_1, \dots, \mathbf{y}_n\}$  is the same as on the data set  $\mathcal{X}$ , the data  $\mathcal{Y}$  is a DR of  $\mathcal{X}$ . However, because the manifold  $M$  is underlying, we

cannot explicitly construct the isometric mapping  $f$  from the DR. Instead, we construct a (approximative) geodesic metric  $d_M$  on the data  $\mathcal{X}$ . By (5.19), the metric  $D_M$  induces a Euclidean metric on  $\mathbf{Y}$ . Using the technique introduced in CMDS, we can find  $\mathbf{Y}$  from the metric.

### 5.2.2 Geodesic Metric on Discrete Set

The key step of Isomap is the construction of the geodesic metric on the observed data. As we have mentioned above, we cannot measure the geodesic distances between points of a manifold without the formulation of the manifold. Hence Isomap uses the graph distance two substitute the geometric distance.

Assume that a neighborhood system is well defined on a data set  $\mathbf{X}$ , which created a graph  $G = [\mathbf{X}, E]$  such that  $(\mathbf{x}_i, \mathbf{x}_j) \in E$  if and only if  $\mathbf{x}_i$  and  $\mathbf{x}_j$  are adjacent. To construct the graph metric  $d_G$  on  $G$ , we define the graph distance between  $\mathbf{x}, \mathbf{y} \in \mathcal{X}$  as follows:

1. If  $(\mathbf{x}, \mathbf{y}) \in E$ , then  $d_G(\mathbf{x}, \mathbf{y}) = d_2(\mathbf{x}, \mathbf{y})$ .
2. If  $(\mathbf{x}, \mathbf{y}) \notin E$ , for a path

$$\gamma = (\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_{s+1})$$

that connects  $\mathbf{x}$  and  $\mathbf{y}$  with  $\mathbf{x} = \mathbf{x}_0$  and  $\mathbf{x}_{s+1} = \mathbf{y}$ , we define the path distance by

$$d_\gamma = \sum_{k=0}^s d_2(\mathbf{x}_k, \mathbf{x}_{k+1}).$$

Let  $\Gamma$  be the set of all paths that connect  $\mathbf{x}$  and  $\mathbf{y}$ . Then the graph distance between  $\mathbf{x}$  and  $\mathbf{y}$  is defined by:

$$d_G(\mathbf{x}, \mathbf{y}) = \min_{\gamma \in \Gamma} d_\gamma(\mathbf{x}, \mathbf{y}).$$

When the data points are dense enough on its domain on  $M$ , the graph distance  $d_G$  approximates the geodesic distance  $d_M$  under certain conditions. This is intuitive and can be analyzed rigorously.

Computing all graph distances between the points of a data set is computationally expensive, particularly, when the size of the data set is very large. The algorithm developed by Dijkstra in 1959 can be applied to the fast computation. We shall discuss the details of Dijkstra's algorithm in Section 5.2.4.

### 5.2.3 Isomap Kernel and Its Constant Shift

Assume that the data set  $\mathcal{X}$  lies on a manifold  $M \subset \mathbb{R}^D$ , and  $f$  is an isometric mapping from  $M$  to  $\mathbb{R}^d$  such that  $f(\mathbf{x}_i) = \mathbf{y}_i$ . The Isomap kernel for a given graph  $G = (\mathcal{X}, E)$  is constructed from the graph metric  $\mathbf{D}_G = (d_G(i, j))$  on  $G$ , where the  $(i, j)$ -th entry  $d_G(i, j)$  is the graph distance between  $\mathbf{x}_i$  and  $\mathbf{x}_j$ . It is easy to verify that the graph distance  $d_G$  satisfies the distance axioms:

1.  $d_G(i, j) \geq 0$ , and  $d_G(i, j) = 0$  if and only if  $i = j$ .
2.  $d_G(i, j) = d_G(j, i)$ .
3.  $d_G(i, j) \leq d_G(i, k) + d_G(k, j)$ .

Let  $d_M(i, j)$  denote the geodesic distance between  $\mathbf{x}_i$  and  $\mathbf{x}_j$ . It is obvious that when they are adjacent

$$d_G(i, j) \leq d_M(i, j).$$

When the data set  $\mathcal{X}$  is dense enough in its domain on  $M$ , the graph distance  $d_G(i, j)$  is close to the geodesic distance  $d_M(i, j)$  so that the graph metric  $\mathbf{D}_G$  well approximates the geodesic metric  $\mathbf{D}_M$ . As a result,  $\mathbf{D}_G$  also approximates the Euclidean metric on the set  $\mathcal{Y} = \{\mathbf{y}_1, \dots, \mathbf{y}_n\}$  very well:

$$\mathbf{D}_G \approx \mathbf{D} \triangleq (d_2(\mathbf{y}_i, \mathbf{y}_j)).$$

Let  $\mathbf{S} = \mathbf{D}_G \circ \mathbf{D}_G = (d_G^2(i, j))$  and  $\mathbf{S}^c$  be its centered matrix:  $\mathbf{S}^c = \mathbf{H}\mathbf{S}\mathbf{H}$ . By Theorem 60 in Section 4.3.2.3,

$$\mathbf{G}^c = -\frac{1}{2}\mathbf{S}^c \quad (5.20)$$

is the Gram matrix of a centered data set  $\mathcal{Y}$ . Applying PCA to  $\mathbf{G}^c$ , we obtain  $\mathcal{Y}$ . The matrix  $\mathbf{G}^c$  in (5.20) is called the Isomap kernel.

When the original data set  $\mathcal{X}$  is not dense enough in its domain on  $M$ , the graph distances of some point pairs may be much smaller than the geodesic distances. Then, the graph metric  $\mathbf{D}_G$  cannot approximate the geodesic metric  $\mathbf{D}_M$  very well. As a result, the Isomap kernel  $\mathbf{G}^c$  in (5.20) may not be positive semi-definite. Therefore, it cannot serve as a Gram matrix of any data set. A simple idea to solve the problems is to add a positive number  $\tau$  to the graph distance  $d_G(i, j)$ ,  $i \neq j$ , as a compensation:

$$d_\tau(i, j) = \begin{cases} d_G(i, j) + \tau, & i \neq j, \\ 0, & i = j. \end{cases} \quad (5.21)$$

After the modification, the matrix  $\mathbf{S}$  becomes:

$$\mathbf{S}_\tau = (d_r^2(i, j)), \quad (5.22)$$

and  $\mathbf{G}^c$  becomes

$$\mathbf{G}_\tau^c = -\frac{1}{2}\mathbf{S}_\tau^c. \quad (5.23)$$

We claim that there exists a  $\tau > 0$  that makes  $\mathbf{G}_\tau^r$  a psd matrix. Then  $\mathbf{G}_\tau^c$  can be used as a modification of the Isomap kernel. We call  $\mathbf{G}_\tau^c$  a constant-shifted Isomap kernel.

To confirm the existence of such a  $\tau$  and then give a formula to compute  $\tau$ , we write  $\mathbf{S}_\tau = \mathbf{S} + 2\tau\mathbf{D} + \tau^2(\mathbf{E} - \mathbf{I})$ , where  $\mathbf{E}$  is an all-one matrix. Then

$$\mathbf{G}_\tau^c = -\frac{1}{2}(\mathbf{H}\mathbf{SH} + 2\tau\mathbf{HDH} + \tau^2\mathbf{H}(\mathbf{E} - \mathbf{I})\mathbf{H}) = \frac{1}{2}(\tau^2\mathbf{H} - 2\tau\mathbf{D}^c - \mathbf{S}^c), \quad (5.24)$$

which is a quadratic form of  $\tau \geq 0$ . When  $\tau$  is large enough, the matrix  $\mathbf{G}_\tau^c$  is psd.

To find a lower bound of  $\tau$ , we study the eigenvalue of  $\mathbf{G}_\tau^c$ . The vector  $\mathbf{1}$  is a trivial eigenvector of  $\mathbf{G}_\tau^c$  achieving the eigenvalue 0. Then the other eigenvectors are perpendicular to it. Let  $\lambda_{\min}$  be the smallest eigenvalue of  $-\mathbf{S}^c$  (which is negative) and  $\mu_{\min}$  the smallest eigenvalue of  $-\mathbf{D}^c$ . Let  $\tau^+$  be the positive root of the quadratic equation:

$$\tau^2 + 2\tau\mu_{\min} + \lambda_{\min} = 0, \quad (5.25)$$

which gives

$$\tau^+ = \sqrt{\mu_{\min}^2 - \lambda_{\min}} - \mu_{\min}. \quad (5.26)$$

We claim that  $\mathbf{G}_\tau^c$  is psd when  $\tau \geq \tau^+$ . Indeed, for each vector  $\mathbf{z}$ , we have

$$\mathbf{z}^T \mathbf{G}_\tau^c \mathbf{z} \geq \frac{1}{2} \|\mathbf{Hz}\|^2 (\tau^2 + 2\tau\mu_{\min} + \lambda_{\min}) \geq 0.$$

Proof. Define  $\tilde{\mathbf{z}} = \mathbf{Hz}$ . Then  $\mathbf{1}^T \tilde{\mathbf{z}} = 0$ .

$$\begin{aligned} \mathbf{z}^T \mathbf{G}_\tau^c \mathbf{z} &= \mathbf{z}^T \mathbf{H} \mathbf{G}_\tau^c \mathbf{H} \mathbf{z} = \tilde{\mathbf{z}}^T \mathbf{G}_\tau^c \tilde{\mathbf{z}} \\ &= \frac{1}{2} (\tau^2 \tilde{\mathbf{z}}^T \mathbf{H} \tilde{\mathbf{z}} - 2\tau \tilde{\mathbf{z}}^T \mathbf{D}^c \tilde{\mathbf{z}} - \tilde{\mathbf{z}}^T \mathbf{S}^c \tilde{\mathbf{z}}) \\ &= \frac{1}{2} (\tau^2 \|\tilde{\mathbf{z}}\|^2 - 2\tau \tilde{\mathbf{z}}^T \mathbf{D}^c \tilde{\mathbf{z}} - \tilde{\mathbf{z}}^T \mathbf{S}^c \tilde{\mathbf{z}}) \\ &\geq \frac{1}{2} (\tau^2 \|\tilde{\mathbf{z}}\|^2 + 2\tau\mu_{\min} \|\tilde{\mathbf{z}}\|^2 + \lambda_{\min} \|\tilde{\mathbf{z}}\|^2) \\ &= \frac{1}{2} \|\tilde{\mathbf{z}}\|^2 (\tau^2 + 2\tau\mu_{\min} + \lambda_{\min}) \geq 0. \end{aligned}$$

Thus, the minimal value of  $\tau$  that makes  $\mathbf{G}_\tau^c$  psd is no greater than  $\tau^+$  in (5.26). This minimal value is given in the following theorem proven by Cailliez.

**定理93.** *The minimal number  $\tau^*$  that makes  $\mathbf{G}_\tau^c$  psd is the largest eigenvalue of the matrix*

$$\mathbf{B} = \begin{pmatrix} 0 & -\mathbf{S}^c \\ -\mathbf{I} & 2\mathbf{D}^c \end{pmatrix}.$$

The technique used in (5.21) is called the constant-shift technique, which is not only used for the modification of Isomap kernel, but also used whenever a symmetric matrix needs to be altered into a psd.

#### 5.2.4 Dijkstra's Algorithm

In Isomap, the computation of the graph distances between points in data set costs the most time. Assume that a graph has been constructed. To compute the distance between two nodes that are not directly connected by an edge, we need to compute all possible path distances and then minimize the results. An exhaustive iterative method known as Floyd's algorithm is optimal for dense graphs, whose cost is  $O(n^3)$ . However, most data graphs for DR are constructed by neighborhood systems, either  $k$ -neighborhood or  $\varepsilon$ -neighborhood. Therefore, their adjacency matrices are sparse. For sparse graphs, the computing cost can be reduced by using Dijkstra's algorithm, conceived by Dijkstra in 1959. The algorithm solves the shortest path searching problem for a weighted graph, producing a shortest path tree. Using the path tree, the graph distances computed in advance are reusable for computing the graph distances of new pairs.

Dijkstra's algorithm is a breadth-first search method. It propagates the graph distance step by step. Its complexity is  $O(n^2 \log n)$ .

The idea of Dijkstra's algorithm is simple. Suppose that you want to find the shortest path between two intersections on a geographic map, from a starting point to a destination. To accomplish this, you could highlight the streets (tracing the streets with a marker) in a certain order until you have a route highlighted from the starting point to the destination. The order is conceptually simple: at each iteration, create a set of intersections consisting of every unmarked intersection that is directly connected to a marked intersection, this will be your set of considered intersections. From that set, find the closest intersection to the destination and highlight it and mark the street to that intersection, draw an arrow with the direction, then repeat. In each stage mark just one new intersection. When you reach the destination, the drawn arrows indicate the shortest path.

We demonstrate the steps of Dijkstra's algorithm in Fig. 5.4. For convenience, we call the node at which we are starting the initial node, and call the distance from the

initial node to a node, say  $b$ , the distance of  $b$ . At the initial step, Dijkstra's algorithm assigns some initial distance values for all nodes, and then improves them step by step. In our demonstration, the first figure is a subgraph consisting of 6 nodes. The initial node is Node 1, marked as  $a$ , and the destination is Node 6, marked as  $b$ . The boxed number on each edge is the edge distance. We shall calculate all graph distances from  $a$  (Node 1) to other nodes.

Step 1. This is the initial step. We assign to every node a distance value. Since we assume that the initial node is  $a$ , we set the distance of Node 1 at zero. The distances of all other nodes are set at infinity. In the figures, the distance of a node is shown nearby. At the initial step, all nodes are considered unvisited, except the initial node  $a$  that is set as current.

Step 2. Calculate all tentative distances from the initial node to its neighbors. There are three nodes, Nodes 2, 3, and 6, directly connected to Node 1 by edges. The distance of Node 1 is 0, and the edge distance from Node 1 to Node 2 is 1. Hence we get the tentative distance  $0+1=1$  for Node 2. Similarly, we calculate the tentative distances 8 and 14 of Node 3 and Node 5, respectively. Mark these tentative distances. Since all distances of the nodes directly connecting Node 1 are calculated, Node 1 will not be used in the further calculation. We mark it as **out**.

Step 3. We set Node 2 as the current node, having the distance 1. It has two visitable neighbors: Nodes 3 and 4. The path distance of the path of Node 1 → Node 2 → Node 3 is calculated by adding the edge distance 3 to the distance of Node 2. We get the distance  $1+5=6$ , which is smaller than the current tentative distance 8. Hence, we update the distance of Node 3 to 6. The tentative distance of Node 4 is  $10+1=11$ . Record all computed distances, and mark Node 2 as **out**.

Step 4. We now set Node 3 as the current node, which has the distance 6. In the similar way, we calculate the distances of Node 5 and Node 4 starting from Node 3. The results are 12 and 11 respectively. Update the distance of Node 5 to 12 and keep 11 for Node 4. We then set Node 3 as out and make Node 4 and Node 5 the current nodes.

Step 5. Finally, we calculate the distance of Node 6 starting from Node 5 and Node 4 respectively, getting 16 and 13. We choose 13 as the distance of Node 6. The computation of all distances from Node 1 to other nodes is completed. Particularly, the distance from  $a$  to  $b$  is 13. The computed distances of the nodes can be used

in the further computation, For example, if  $b$  (Node 6) has neighbors Nodes 7 and 8, then the path distances from  $a$  to Node 7 and to Node 8 can be calculated by adding 13 to the edge distances Node 6 → Node 7 and Node 8, respectively.

### 5.2.5 Isomap Algorithm

1. **Neighborhood definition.** The neighborhood of a point in the set  $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$  can be constructed by either its  $k$ -neighborhood or  $\varepsilon$ -neighborhood. If the notion of  $k$ -neighborhood is used, then the method is called a  $k$ -Isomap, or else, it is called an  $\varepsilon$ -Isomap. We denote the neighborhood of  $\mathbf{x}_i$  by  $N(i)$ .
2. **Graph distance computation.** The defined neighborhood system creates a graph  $G = (\mathcal{X}, E)$  on the data set. We compute the graph metric on  $\mathcal{X}$  in terms of the graph distance  $d_G(i, j)$  between  $\mathbf{x}_i$  and  $\mathbf{x}_j$  for each pair of points on  $\mathcal{X}$ , by using Dijkstra's algorithm. Note that if  $k$ -neighborhood is adopted, then  $d_G(j, i)$  may not be equal to  $d_G(i, j)$  since  $\mathbf{x}_j \in N(i) \not\Rightarrow \mathbf{x}_i \in N(j)$ . To ensure  $d_G(i, j) = d_G(j, i)$ , we may either define the graph distance between  $\mathbf{x}_i$  and  $\mathbf{x}_j$  by  $\min(d_G(i, j), d_G(j, i))$ , or modify the graph  $G$  to an undirected graph. If there is no path connecting  $\mathbf{x}_i$  and  $\mathbf{x}_j$ , then we set  $d_G(i, j) = \infty$ .
3. **DR kernel construction.** Assume the points of  $\mathcal{X}$  is dense enough. Then  $\mathbf{D}_G$  is Euclidean. Compute the matrices  $\mathbf{S}_G = (d_G^2(i, j))$  and

$$\mathbf{G}^c = -\frac{1}{2}\mathbf{H}\mathbf{S}_G\mathbf{H},$$

where  $\mathbf{G}^c$  is the kernel of Isomap DR method. Otherwise, to ensure that the kernel is psd, the constant-shift technique may be applied.

4. **Eigen decomposition.** Let the eigen decomposition of  $\mathbf{G}^c$  be given by

$$\mathbf{G}^c = \mathbf{U}\Lambda\mathbf{U}^T,$$

and  $\mathbf{Y} = (\mathbf{y}_1, \dots, \mathbf{y}_n)$  be defined by:

$$\mathbf{Y} = (\sqrt{\lambda_1}\mathbf{u}_1, \sqrt{\lambda_2}\mathbf{u}_2, \dots, \sqrt{\lambda_d}\mathbf{u}_d)^T.$$

Then  $\mathbf{Y}$  is the DR of  $\mathbf{X}$ .

### 5.2.6 Experiments and Applications of Isomap

#### 5.2.6.1 Testing Isomap Algorithms on Artificial Surfaces

We now show that Isomaps work well for surface data. Its performance depends on, but is not very sensitive to, the neighborhood size.

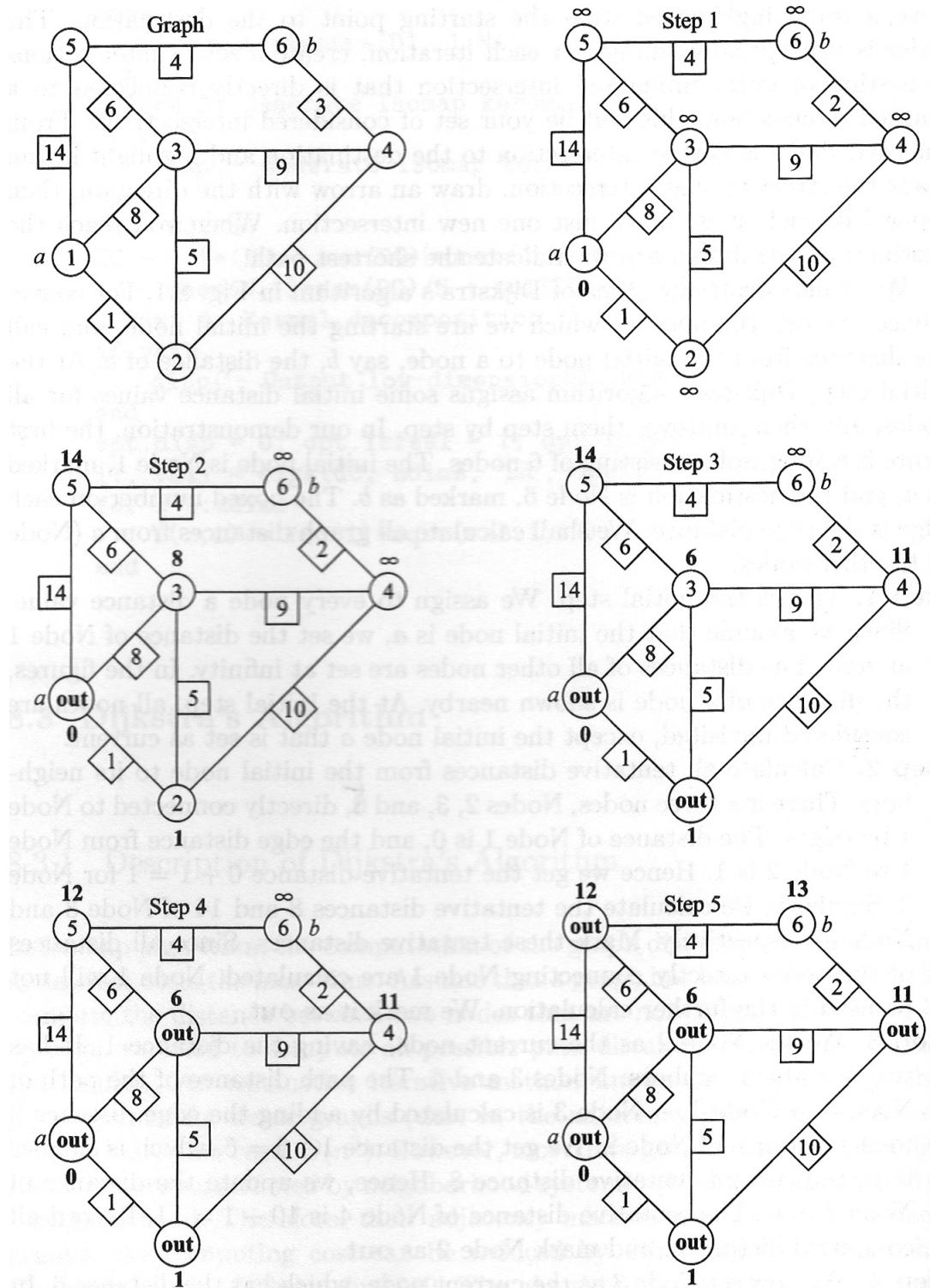


图 5.4: Dijkstra's algorithm runtime. The figure illustrates the Dijkstra's algorithm runtime for a subgraph with 6 nodes. The graph distances from Node 1 to other nodes is computed using the algorithms step by step.

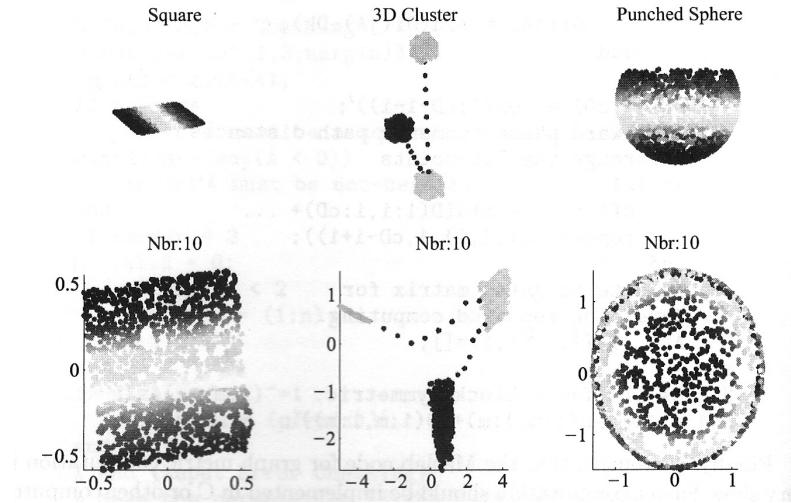


图 5.5: Isomap is performance on three surfaces. On the top line, the left is a square in space, the middle is a 3D-cluster, and the right is a punched sphere. The data set for each surface contains 1000 random samples. Each data graph is constructed using 10-neighborhood, and the graph distance are computed by Dijkstra's algorithm. Their 2D DR data sets are displayed on the bottom line correspondingly. It is known that Isomap successfully reduces the dimension of nonlinear surface data sets.

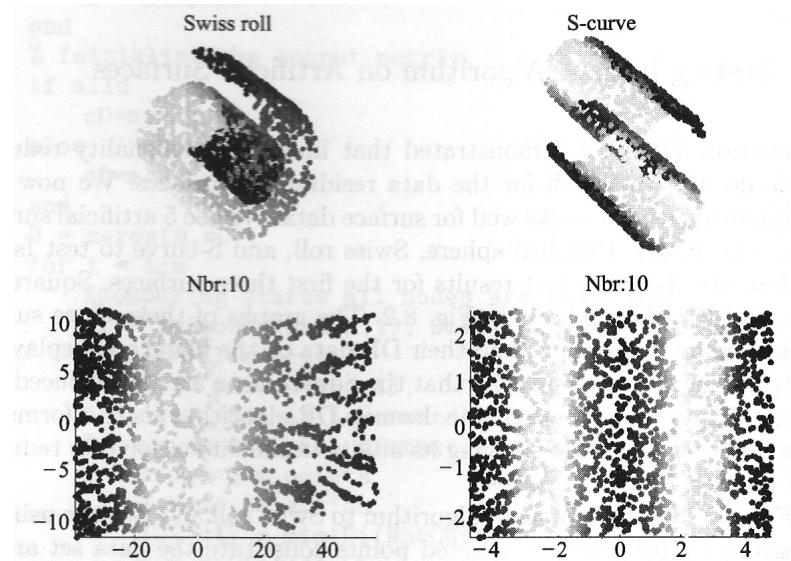


图 5.6: Iso map is performed on Swiss and S-curve. These two are expandable surfaces. The original data are shown on the top line where the left is Swiss roll and the right is S-curve. Their 2D DR data are displayed on the bottom line correspondingly. The results show that Iso map algorithm reduces the dimension of these two surface data sets very well.

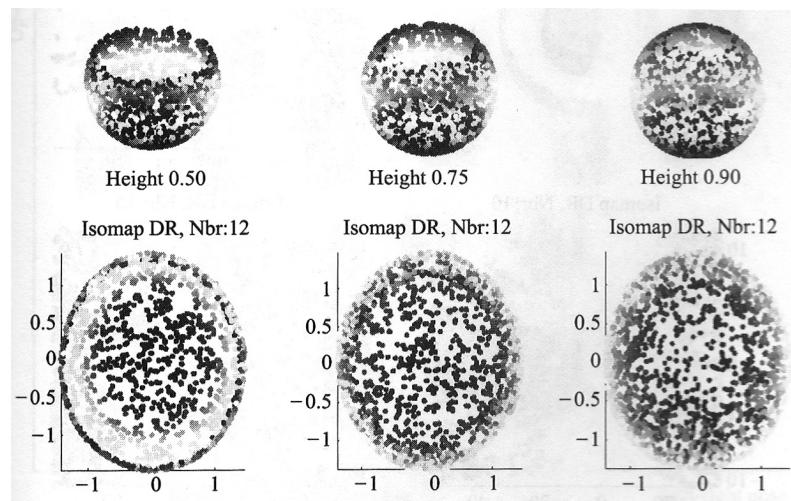


图 5.7: Isomap is performed on the punched spheres with various heights. On the top line, we present three punched sphere with heights of 0.5, 0.75, and 0.9, respectively. The diameter of the sphere is 1. On the bottom line are their DR data sets correspondingly. The results show that the DR data sets do not preserve the local geometry near the boundary when the punched holes become smaller. It is reasonable that when the heights increase, the graph distances between the points near the punched boundary do not well approximate the geodesic distances, so that large errors occur.

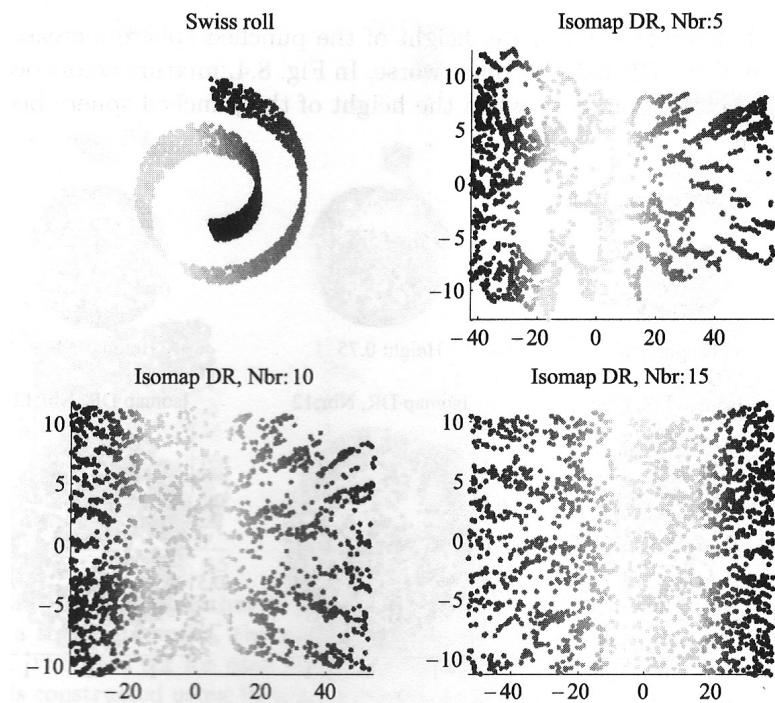


图 5.8: Isomap embeddings of Swiss roll of various sizes of neighborhood. Three different sizes of  $k$ -neighborhood,  $k = 5, 10, 15$ , are used in this experiment. The results show that Isomap is insensitive to the neighborhood sizes, a wide range of neighborhood sizes can be chosen in the algorithm for producing the similar DR results. However, when the size of neighborhood is too small, say  $k = 5$ , in this experiment, although the low-dimensional data have a little distortion, the geometry of the original data is still well preserved in most of the area.

### 5.2.6.2 Isomap Algorithms in Visual Perception

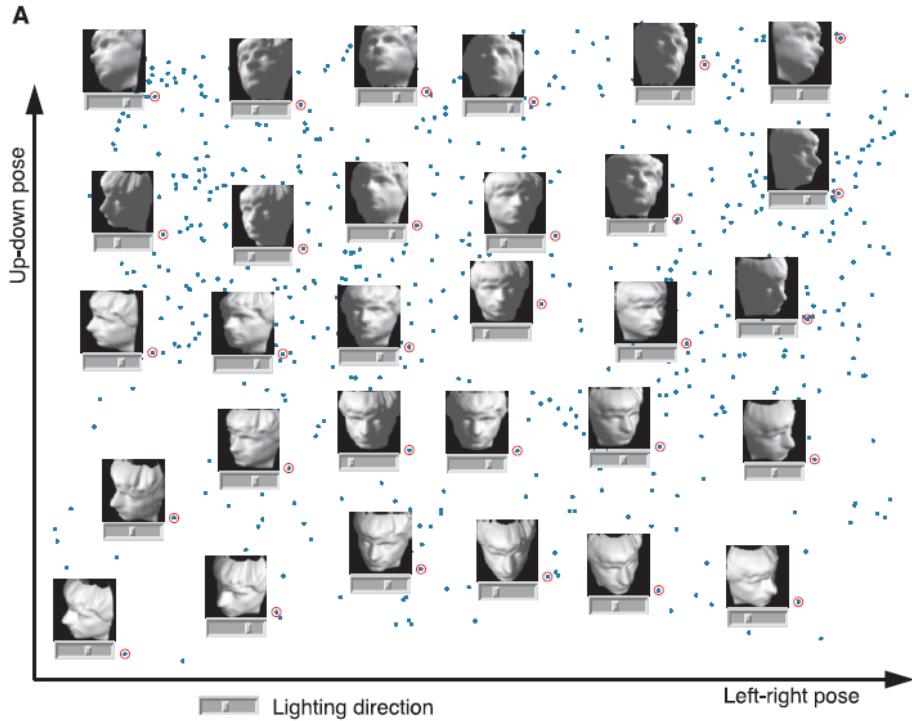


图 5.9: A canonical dimensionality reduction problem from visual perception. The input consists of a sequence of 4096-dimensional vectors, representing the brightness values of  $64 \times 64$  pixel images of a face rendered with different poses and lighting directions. Applied to  $N = 698$  raw images, Isomap ( $K = 6$ ) learns a three-dimensional embedding of the data's intrinsic geometric structure. A two-dimensional projection is shown, with a sample of the original input images (red circles) superimposed on all the data points (blue) and horizontal sliders (under the images) representing the third dimension. Each coordinate axis of the embedding correlates highly with one degree of freedom underlying the original data: leftright pose ( $x$  axis,  $R = 0.99$ ), up-down pose ( $y$  axis,  $R = 0.90$ ), and lighting direction (slider position,  $R = 0.92$ ). The input-space distances  $d_X(i, j)$  given to Isomap were Euclidean distances between the 4096-dimensional image vectors.

### 5.2.6.3 Conclusion

Isomap is a computationally stable algorithm. Various experiments show that most high-dimensional data lie on convex manifolds. Hence, Isomap method usually well models data and it is widely adopted in high-dimensional data analysis. The method is also insensitive to the neighborhood sizes used in the data graph construction, to the

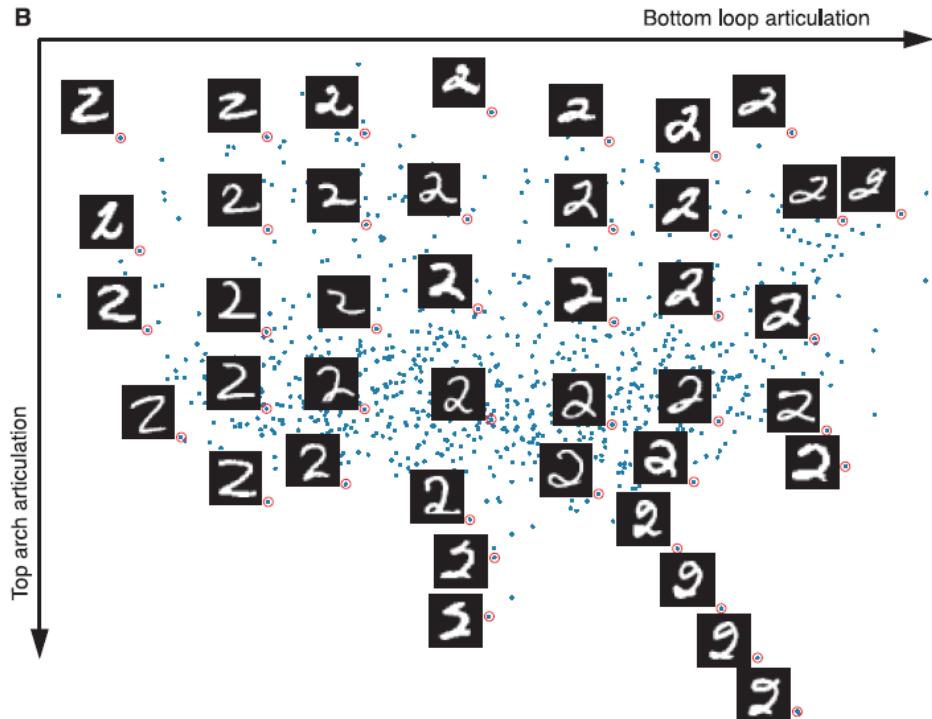


图 5.10: Isomap applied to  $N = 1000$  handwritten “2”s from the MNIST database. The two most significant dimensions in the Isomap embedding, shown here, articulate the major features of the “2”: bottom loop ( $x$  axis) and top arch ( $y$  axis). Input-space distances  $d_X(i, j)$  were measured by tangent distance, a metric designed to capture the invariances relevant in handwriting recognition. Here we used  $\varepsilon$ -Isomap (with  $\varepsilon = 4.2$ ) because we did not expect a constant dimensionality to hold over the whole data set; consistent with this, Isomap finds several tendrils projecting from the higher dimensional mass of data and representing successive exaggerations of an extra stroke or ornament in the digit.

smoothness of the data underlying manifolds, and to the noise on the data. Hence, it is a promising DR method. However, estimating geodesic distances requires significantly intensive computation. Since the Isomap kernel is dense, the Isomap algorithm is computationally expensive. Besides, if the data set does not lie on a convex manifold (for instance, the manifold has some “holes” or “cracks”), the Isomap method may not preserve the data geometry very well near the holes and cracks, producing large embedding errors there.

**练习94.** Use one person’s face images in Data for MATLAB hackers (<http://www.cs.nyu.edu/~roweis/data.html>) as the data for dimensionality reduction with Isomap, where the target dimension is 2, and see whether it is better than CMDS, e.g., whether the face images are arranged better in some order.

### 5.3 Locally Linear Embedding (LLE)

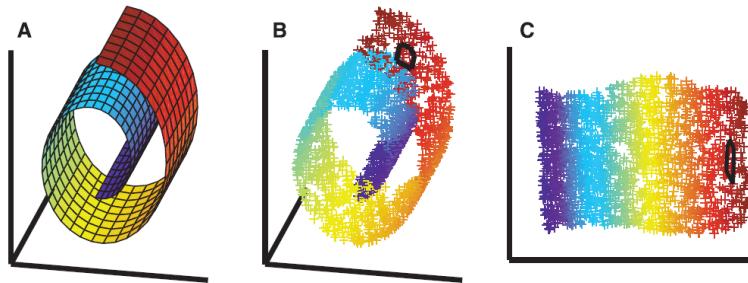


图 5.11: The problem of nonlinear dimensionality reduction, as illustrated for three-dimensional data (B) sampled from a two-dimensional manifold (A). An unsupervised learning algorithm must discover the global internal coordinates of the manifold without signals that explicitly indicate how the data should be embedded in two dimensions. The color coding illustrates the neighborhood preserving mapping discovered by LLE; black outlines in (B) and (C) show the neighborhood of a single point. Unlike LLE, projections of the data by principal component analysis (PCA) or classical MDS map faraway data points to nearby points in the plane, failing to identify the underlying structure of the manifold. Note that mixture models for local dimensionality reduction, which cluster the data and perform PCA within each cluster, do not address the problem considered here: namely, how to map high-dimensional data into a single global coordinate system of lower dimensionality.

LLE is based on simple geometric intuitions: if a data set is sampled from a smooth manifold, then neighbors of each point remain nearby and are similarly co-located in the

low-dimensional space. In LLE, each point in the data set is linearly embedded into a locally linear patch of the manifold. Then low-dimensional data is constructed such that the locally linear relations of the original data are preserved.

### 5.3.1 Description of LLE

The method of LLE was introduced by Roweis and Saul [118]. It is based on simple geometric intuitions. Assume that the data set  $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subset \mathbb{R}^D$  is well sampled from a smooth underlying manifold and a neighborhood structure is defined on the data set, such that each neighborhood lies on or is close to a locally linear patch of the manifold. In LLE, the local geometry of each of these patches is characterized by linear coefficients that approximately reconstruct each data point from its neighbors. Then LLE computes a low-dimensional embedding that is optimized to preserve the local configurations of the data.

### 5.3.2 Barycentric Coordinates

In LLE, a low-dimensional embedding is constructed to preserve the local geometry of data in a neighborhood, using the information within the local area. LEE realizes the local linear embedding using barycentric coordinate of each neighborhood. Barycentric coordinates are widely used in physics, computer graphics, and other areas. Assume that a set  $\mathcal{P} = \{\mathbf{p}_1, \dots, \mathbf{p}_{m+1}\} \subset \mathbb{R}^m$  generates a simplex in  $\mathbb{R}^m$  with the vertices  $\mathbf{p}_1, \dots, \mathbf{p}_{m+1}$ . Then each point  $\mathbf{p}$  inside the simplex  $\mathcal{P}$  can be uniquely represented by the weights (or coefficients)  $\mathbf{w} = (w_1, \dots, w_{m+1})^T$  as:

$$\mathbf{p} = \mathbf{P}\mathbf{w},$$

under the constraint  $\mathbf{1}^T \mathbf{w} = 1$ , where  $\mathbf{P} = (\mathbf{p}_1, \dots, \mathbf{p}_{m+1})$ . The coefficients  $w_i$  are called the barycentric coordinates of  $\mathbf{p}$  (with respect to the simplex  $\mathcal{P}$ ).

The advantage of barycentric coordinates is evident. The coordinates are “purely” geometric: they are independent of the physical locations and the dimension of the vertices. More precisely, barycentric coordinates are invariant under rigid motion and isometric embedding. If we consider each barycentric coordinate  $w_i$  as a function of  $\mathbf{p}$ :  $w_i = w_i(\mathbf{p})$ , then  $w_i(\mathbf{p})$  has the following properties:

1. **Linearity.** It is a linear function of  $\mathbf{p}$ :

$$w_i(\lambda\mathbf{p} + (1 - \lambda)\mathbf{q}) = \lambda w_i(\mathbf{p}) + (1 - \lambda)w_i(\mathbf{q}).$$

2. **Positivity.** If  $\mathbf{p}$  is strictly inside the simplex, then  $w_i(\mathbf{p}) > 0$ .

3. **Lagrange interpolating property.** For each vertex  $\mathbf{p}_j$ ,  $w_i(\mathbf{p}_j) = \delta_{ij}$ .

Since the barycentric coordinate  $w_i$  favors  $\mathbf{p}$  that is close to  $\mathbf{p}_i$ , it naturally describes the local similarities of data. Due to these properties, under the barycentric coordinates each linear function  $F$  on  $\mathbf{p}$  has the representation

$$F(\mathbf{p}) = \sum_{i=1}^{m+1} w_i(\mathbf{p}) f_i,$$

where  $f_i = F(\mathbf{p}_i)$ .

Therefore, if  $\phi$  is a full-rank linear transformation, which maps the set  $\mathcal{P}$  onto the set  $\mathcal{Q} = \{\mathbf{q}_i = \phi(\mathbf{p}_i)\}$ , then

$$\mathbf{q} \triangleq \phi(\mathbf{p}) = \sum_{i=1}^{m+1} w_i \mathbf{q}_i,$$

i.e., the barycentric coordinates are preserved under linear transformation.

We may generalize the barycentric coordinates to any convex polytope in  $\mathbb{R}^m$ . Assume that  $\mathcal{P} = \{\mathbf{p}_1, \dots, \mathbf{p}_{s+1}\} \subset \mathbb{R}^m$  is a convex polytope in  $\mathbb{R}^m$ . Then each point  $\mathbf{p}$  inside the polytope can be represented as

$$\mathbf{p} = \mathbf{P}\mathbf{w},$$

with the constraint  $\mathbf{1}^T \mathbf{w} = 1$  and  $\mathbf{w} \geq 0$ . In this case the coordinates may not be unique. Many methods, for example, Hermite mean value interpolation, can be used to compute the generalized barycentric coordinates.

### 5.3.3 LLE Method

We first focus on the construction of LLE DR kernels. Let  $\mathbf{X}_{D \times n} = (\mathbf{x}_1, \dots, \mathbf{x}_n)$  be the data matrix of a given data set  $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subset \mathbb{R}^D$ , which is sampled from a  $d$ -dimensional manifold  $M \subset \mathbb{R}^D$ . Assume that the neighborhood of each point  $\mathcal{X}$  is well defined by using either  $k$  nearest points or an  $\varepsilon$ -ball, even a hybrid one. Then the neighborhood structure defines a graph  $G = (\mathcal{X}, \mathbf{A})$  on  $\mathcal{X}$ , where  $\mathbf{A}$  is the adjacency matrix. Then the index set of the neighborhood of  $\mathbf{x}_i$ , denoted by  $N(i)$ , is given by the  $i$ th row of  $\mathbf{A}$  such that

$$N(i) = \{j | \mathbf{A}_{i,j} \neq 0\},$$

and the neighborhood of  $\mathbf{x}_i$ , denoted by  $O_i$ , is the set

$$O(i) = \{\mathbf{x}_i \in \mathcal{X} | j \in N(i)\}.$$

Next, we construct an  $n \times n$  weight matrix  $\mathbf{W}$  on the graph to define the similarity between the data points. Since a point  $\mathbf{x}_j \notin O(i)$  is considered not similar to  $\mathbf{x}_i$ , we set the weight  $w_{i,j} = 0$  if  $\mathbf{A}_{i,j} = 0$ . To define  $w_{i,j}$  for each neighborhood  $\{\mathbf{x}_j\}_{j \in N(i)}$ , where  $\mathbf{x}_j$  is near  $\mathbf{x}_i$  on  $M$ . The geometric structure of  $O(i)$  then can be approximated by its projection on the tangent space  $T_{\mathbf{x}_i}$ . Let  $f$  denote the orthogonal projection from  $M$  to  $T_{\mathbf{x}_i}$  and write  $\tilde{\mathbf{x}} = f(\mathbf{x})$ . The set  $\{\tilde{\mathbf{x}}_j - \tilde{\mathbf{x}}_i | j \in N(i)\}$  spans a subspace of the tangent space of  $T_{\tilde{\mathbf{x}}_i}$ , and there is a weight set

$$\left\{ w_{i,j} \in \mathbb{R} | j \in N(i), \sum_{j \in N(i)} w_{i,j} = 1 \right\},$$

such that  $\sum_{j \in N(i)} w_{i,j} \tilde{\mathbf{x}}_j = \tilde{\mathbf{x}}_i$ . Note that some weights may be negative if  $\tilde{\mathbf{x}}_i$  is not in the polytope of  $\{\tilde{\mathbf{x}}_j\}_{j \in N(i)}$ . We denote the graph degree at  $\mathbf{x}_i$  by  $d_i = |N(i)|$  and denote the  $i$ th **row**<sup>1</sup> of the weight matrix  $\mathbf{W}$  by  $\mathbf{w}_i^T = (w_{i,1}, \dots, w_{i,n})$ , which is sparse and has at most  $d_i$  non-zero entries. For convenience, we define the  $d_i$  dimensional sub-vector of  $\mathbf{w}_i^T$  by

$$(\mathbf{w}_i^{sub})^T = (w_{i,j_1}, \dots, w_{i,j_{d_i}}), \quad j_s \in N(i). \quad (5.27)$$

The set of sub-vectors  $\{\mathbf{w}_i^{sub}\}_{i=1}^n$  together with the neighbor-index set  $\{N(i)\}_{i=1}^n$  uniquely determines the weight matrix  $\mathbf{W}$ . To avoid involving in the tangent projection  $f$  in the weight computation, LLE specifies the choice of the sub-vectors by minimizing the error

$$\mathbf{w}_i^{sub} = \underset{\mathbf{a} \in \mathbb{R}^{d_i}}{\operatorname{argmin}} \left\| \mathbf{x}_i - \sum_{j \in N(i)} a_j \mathbf{x}_j \right\|, \quad s.t. \quad \sum_{j=1}^{d_i} a_j = 1, \quad (5.28)$$

where  $\mathbf{a} = (a_1, \dots, a_{d_i})^T$ . It is clear that the weight  $w_{i,j}$  summarizes the contribution of  $\mathbf{x}_j$  to the reconstruction of  $\mathbf{x}_i$ . As Equation (5.28) shows, to compute the weights, we minimize the cost function in Equation (5.28) subject to two constraints:

1. Sparseness constraint. Each data point  $\mathbf{x}_i$  is reconstructed only from its neighbors  $O(i)$ , enforcing  $w_{i,j} = 0$  if  $\mathbf{x}_j \notin O(i)$ .
2. Invariance constraint. The rows of the weight matrix sum to one:

$$\sum_{j=1}^n w_{i,j} = 1. \quad (5.29)$$

---

<sup>1</sup>Note that in the class I put  $\mathbf{w}_i$  as the columns of  $\mathbf{W}$ .

The invariance constraint is important to preserve the local geometry of the data. It indicates that 1 is an eigenvalue of  $\mathbf{W}$  and the eigenvector  $\mathbf{1} = (1, \dots, 1)^T$  achieves 1. The sum

$$\sum_{j \in N(i)} w_{i,j} \mathbf{x}_j$$

is called the locally linear embedding (LLE) of  $\mathbf{x}_i$ .

**备注95.** Some entries in the weight matrix may be negative. If nonnegative entries of  $\mathbf{W}$  are required, we may set each negative entry at zero and then re-normalize the row vectors of  $\mathbf{W}$  by the sum rule  $\sum_{j=1}^n w_{i,j} = 1$ . When  $\mathbf{W}$  is a non-negative matrix, 1 is the largest eigenvalue of  $\mathbf{W}$ . However, in applications, forcing the weights to become nonnegative does not always produce better results. Sometimes, it distorts the local geometric structure of data.

Finally, the DR data set  $\mathcal{Y} = \{\mathbf{y}_1, \dots, \mathbf{y}_n\} \subset \mathbb{R}^d$  of LLE method is computed by optimally preserving the local linear embedding. That is,  $\mathcal{Y}$  minimizes the locally linear embedding error:

$$\mathcal{Y} = \underset{\mathbf{y} \in \mathbb{R}^d}{\operatorname{argmin}} \sum_{i=1}^n \left\| \mathbf{y}_i - \sum_{j \in N(i)} w_{i,j} \mathbf{y}_j \right\|^2 \quad (5.30)$$

under the constraints

$$\mathbf{Y}\mathbf{1} = \mathbf{0} \quad \text{and} \quad \mathbf{Y}\mathbf{Y}^T = \mathbf{I}_d. \quad (5.31)$$

The first constraint is to remove the translation ambiguity by making the mean of embedded points zero. The second one is to remove the rotation ambiguity.

To solve the minimization problem (5.30) under the constraints (5.31), we construct the LLE DR kernel

$$\mathbf{K} = (\mathbf{I} - \mathbf{W})^T (\mathbf{I} - \mathbf{W}), \quad (5.32)$$

which is positive semi-definite, having the 0-eigenvector  $\mathbf{1}$ .

Let  $\mathbf{u}_1, \dots, \mathbf{u}_d$  denote the eigenvectors corresponding to the 2nd and up to the  $(d+1)$ st smallest eigenvalues of  $\mathbf{K}$ , respectively. Write

$$\mathbf{U}_d = [\mathbf{u}_1 \cdots \mathbf{u}_d].$$

Then  $\mathbf{Y} = \mathbf{U}_d^T$  is the DR data matrix. We put the justification below.

### 5.3.4 Reduction of the DR Data to LLE Model

We see how to find the DR data set  $\mathcal{Y}$  and what constraints it must satisfy. Note that the number of neighbors of each point of  $\mathcal{X}$  is greater than  $d$ . Hence, the graph  $G = (\mathcal{X}, \mathbf{A})$  has no isolate points. We also assume that  $G$  is connected so that  $\mathbf{0}$  is a simple singular value of the Laplacian  $\mathbf{L} = \mathbf{I} - \mathbf{W}$  and  $\frac{1}{\sqrt{n}}\mathbf{1} \in \mathbb{R}^n$  is the 0-eigenvector. Let  $\mathcal{Y} = \{\mathbf{y}_1, \dots, \mathbf{y}_n\} \subset \mathbb{R}^d$  be the DR data set of  $\mathcal{X}$  that is obtained as the solution of

$$\mathcal{Y} = \operatorname{argmin}_{\mathcal{Y}} \Phi(\mathcal{Y}), \quad \Phi(\mathcal{Y}) = \sum_{i=1}^n \left\| \mathbf{y}_i - \sum_{j \in N(i)} w_{i,j} \mathbf{y}_j \right\|^2. \quad (5.33)$$

Write  $\mathbf{Y} = (\mathbf{y}_1, \dots, \mathbf{y}_n) \in \mathcal{M}_{d,n}$ . Firstly, since the solution of the minimization problem (5.33) is invariant under the shift, we may ask the centroid of  $\mathcal{Y}$  to be at the origin:  $\sum_{j=1}^n \mathbf{y}_j = \mathbf{0}$ , which is equivalent to the constraint

$$\mathbf{Y}\mathbf{1} = \mathbf{0}. \quad (5.34)$$

Secondly, since  $\mathbf{y}_i, 1 \leq i \leq n$ , can be considered as the manifold coordinates of  $\mathbf{x}_i \in \mathcal{X}$ , we may ask all row vectors of  $\mathbf{Y}$  to form an orthonormal system. It leads to the second constraint:

$$\mathbf{Y}\mathbf{Y}^T = \mathbf{I}_d. \quad (5.35)$$

We now solve the minimization problem under the constraints (5.34) and (5.35). Recall that

$$\begin{aligned} \Phi(\mathcal{Y}) &= \sum_{i=1}^n \operatorname{tr} ((\mathbf{Y}\mathbf{e}_i - \mathbf{Y}\mathbf{w}_i)(\mathbf{Y}\mathbf{e}_i - \mathbf{Y}\mathbf{w}_i)^T) \\ &= \operatorname{tr} \left[ \mathbf{Y} \left( \sum_{i=1}^n (\mathbf{e}_i - \mathbf{w}_i)(\mathbf{e}_i - \mathbf{w}_i)^T \right) \mathbf{Y}^T \right] \\ &= \operatorname{tr} [\mathbf{Y}(\mathbf{I} - \mathbf{W}^T)(\mathbf{I} - \mathbf{W})\mathbf{Y}^T] \\ &\triangleq \operatorname{tr} (\mathbf{Y}\mathbf{K}\mathbf{Y}^T), \end{aligned} \quad (5.36)$$

where  $\mathbf{K} = (\mathbf{I} - \mathbf{W}^T)(\mathbf{I} - \mathbf{W})$ . So the DR kernel is  $\mathbf{K}$ .

Let the spectral decomposition of  $\mathbf{K}$  be given by

$$\mathbf{K} = \mathbf{U}\Lambda\mathbf{U}^T,$$

where  $\Lambda = \operatorname{diag}(\lambda_0, \lambda_1, \dots, \lambda_{n-1})$  with all eigenvalues arranged descended:

$$0 = \lambda_0 \leq \lambda_1 \leq \dots \leq \lambda_{n-1}.$$

We have

$$\Phi(\mathbf{Y}) = \operatorname{tr}(\mathbf{Y}\mathbf{K}\mathbf{Y}^T) = \operatorname{tr}(\mathbf{Y}\mathbf{U}\Lambda(\mathbf{Y}\mathbf{U})^T).$$

Let  $\mathbf{U}_d = (\mathbf{u}_1, \dots, \mathbf{u}_d)$  be the submatrix of  $\mathbf{U}$ , consisting of the eigenvectors corresponding to the 2nd to the  $(d+1)$ st smallest eigenvalues of  $\mathbf{K}$ . Since  $\mathbf{K}\mathbf{1} = \mathbf{0}$ , the vector set  $\{\mathbf{u}_1, \dots, \mathbf{u}_d\}$  satisfies the constraints (5.34) and (5.35). Then the minimum of  $\Phi(\mathbf{Y})$  is attained by  $\mathbf{Y} = \mathbf{U}_d^T$ .

### 5.3.5 LLE Algorithm

An LLE algorithm consists of the following four steps.

1. **Neighborhood definition.** The neighborhood of each point can be either  $k$ -neighborhood or  $\varepsilon$ -neighborhood. The number of points in a neighborhood is chosen to be greater than  $d$ . We denote by  $G = (\mathcal{X}, \mathbf{A})$  the constructed graph, where  $\mathbf{A}$  is the adjacency matrix. Recall that if  $\varepsilon$ -neighborhood is applied, then  $\mathbf{A}$  is symmetric, else if  $k$ -neighborhood is applied, then  $\mathbf{A}$  may be asymmetric. Because the value  $\varepsilon$  in  $\varepsilon$ -neighborhood definition depends on the dimension and the scale of the data points,  $\varepsilon$ -neighborhood is not often used in neighborhood construction for LLE.
2. **Weight matrix construction.** A weight matrix  $\mathbf{W} = (w_{i,j})$  is generated on the graph  $G = (\mathcal{X}, \mathbf{A})$  in the following way. Set  $w_{i,j} = 0$ , if  $\mathbf{A}_{i,j} = 0$ , and compute  $w_{i,j}$  by (5.28), if  $\mathbf{A}_{i,j} = 1$ . The equation (5.28) is solved by the least square method as follows.

Define

$$g(\mathbf{a}) = \frac{1}{2} \left\| \mathbf{x}_i - \sum_{j \in N(i)} a_j \mathbf{x}_j \right\|^2,$$

and the Lagrangian function is

$$L(\mathbf{a}, \lambda) = g(\mathbf{a}) + \lambda(\mathbf{1}^T \mathbf{a} - 1).$$

Differentiate  $L$  w.r.t.  $\mathbf{a}$  gives

$$\tilde{\mathbf{X}}^T (\tilde{\mathbf{X}} \mathbf{a} - \mathbf{x}_i) + \lambda \mathbf{1} = \mathbf{0},$$

where  $\tilde{\mathbf{X}}$  consists of  $\{\mathbf{x}_j\}_{j \in N(i)}$ . So

$$\mathbf{a} = (\tilde{\mathbf{X}}^T \tilde{\mathbf{X}})^{-1} (\tilde{\mathbf{X}}^T \mathbf{x}_i - \lambda \mathbf{1}).$$

The  $\lambda$  in the above is determined by making  $\mathbf{a}$  fulfil the constraint  $\mathbf{1}^T \mathbf{a} = 1$ :

$$1 = \mathbf{1}^T \mathbf{a} = \mathbf{1}^T (\tilde{\mathbf{X}}^T \tilde{\mathbf{X}})^{-1} (\tilde{\mathbf{X}}^T \mathbf{x}_i - \lambda \mathbf{1}).$$

So

$$\lambda = \frac{\mathbf{1}^T(\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T\mathbf{x}_i - 1}{\mathbf{1}^T(\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\mathbf{1}}.$$

Let  $\mathbf{G}_i$  be the (shifted) Gram matrix of  $O(i)$ :

$$\mathbf{G}_i = (\langle \mathbf{x}_j - \mathbf{x}_i, \mathbf{x}_k - \mathbf{x}_i \rangle)_{j,k \in N(i)}.$$

We solve the equation

$$\mathbf{G}_i \mathbf{w}_i = \begin{cases} \mathbf{1}, & \text{rank}(\mathbf{G}_i) = d_i, \\ \mathbf{0}, & \text{rank}(\mathbf{G}_i) < d_i, \end{cases}$$

and then normalize the solution by  $\sum_{j \in N(i)} w_{i,j} = 1$ .

**3. LLE kernel construction.** Write  $\mathbf{L} = \mathbf{I} - \mathbf{W}$ , where  $\mathbf{I}$  is the  $n \times n$  identity matrix.

The LLE kernel is given by

$$\mathbf{K} = \mathbf{L}^T \mathbf{L} = (\mathbf{I} - \mathbf{W})^T (\mathbf{I} - \mathbf{W}).$$

**4. Eigen decomposition of LLE kernel.** Let the spectral decomposition of  $\mathbf{K}$  be given by

$$\mathbf{K} = \mathbf{U} \Lambda \mathbf{U}^T,$$

where  $\Lambda = \text{diag}(\lambda_0, \lambda_1, \dots, \lambda_{n-1})$  with

$$0 = \lambda_0 \leq \lambda_1 \leq \dots \leq \lambda_{n-1}.$$

Assume that the graph  $G = (\mathcal{X}, \mathbf{A})$  is connected, then  $\lambda_1 > 0$ . Let  $\mathbf{u}_1, \dots, \mathbf{u}_d$  be the 2nd to the  $(d+1)$ st columns of  $\mathbf{U}$ . The matrix  $\mathbf{Y} = (\mathbf{u}_1, \dots, \mathbf{u}_d)^T$  is the DR data matrix.

**练习96.** Deduce the solution to (5.28) by the method of Lagrange multiplier and compare with the above solution. Using the data in Exercise 97 to check whether they are identical.

### 5.3.6 Experiments and Applications of LLE

#### 5.3.6.1 Experiments on Artificial Surfaces

In Fig. 5.12, LLE algorithm is applied to reducing the 3-dimensional data of 4 artificial surfaces, 3D-cluster, Punched sphere, Swiss roll, and S-curve, to 2-dimensioanl data, where 10 nearest points are used to construct the neighborhood of a point. The

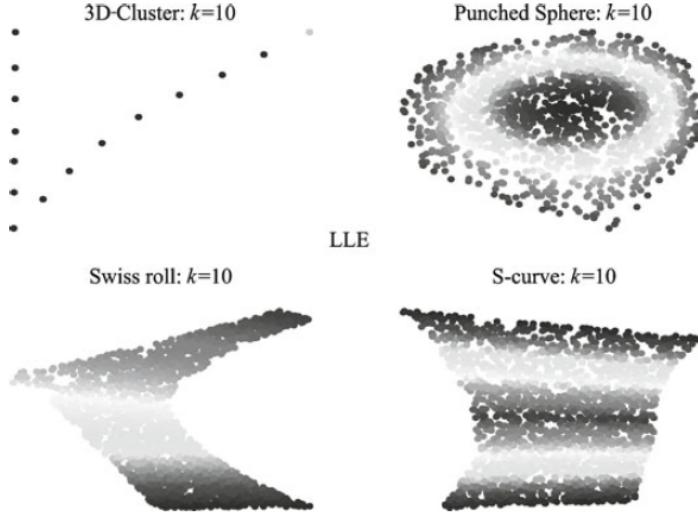


图 5.12: LLE algorithm is tested on 4 surfaces: 3D-cluster, Punched sphere, Swiss roll, and S-curve. 1000 points are randomly sampled on each surface. All of them are reduced to 2-dimensional data (on the plane). The figure displays the DR results of these surfaces.

experiment shows that LLE works well for these surfaces, except 3D-cluster. Note that the data of 3D-cluster lacks the surface structure on the line segments. Because LLE is not an isometric mapping, it usually does not produce valid dimensionality reduction for the data such as 3D-cluster.

Noise on data impacts dimensionality reduction processing. In Fig. 5.13 and Fig. 5.14, we apply the LLE algorithm to noise data, with the noise standard deviation 0.2 and 0.4 respectively. When strong noise exists, the DR data may not preserve the data geometry.

The results of LLE are stable over a range of neighborhood sizes. The stable range of the neighborhood sizes depends on the features of the data, such as the sampling density and the manifold geometry. Several criteria must be kept when choosing the number of neighbors of a data point. First, the algorithm can only be expected to recover embeddings whose dimensionality  $d$  is strictly less than the number of neighbors  $k$ . It is observed that some margin between  $d$  and  $k$  is generally necessary to obtain a topology preserving embedding. However, finding the exact relation between  $k$  and  $d$  is very difficult. It depends on several facts, such as the data geometry and the distribution of samples. Note also that LLE algorithm is based on the assumption that a data point and its nearest neighbors can be modeled as locally linear. For curved data sets, choosing  $k$  too large will in general violate this assumption. As we mentioned above, when  $k >$

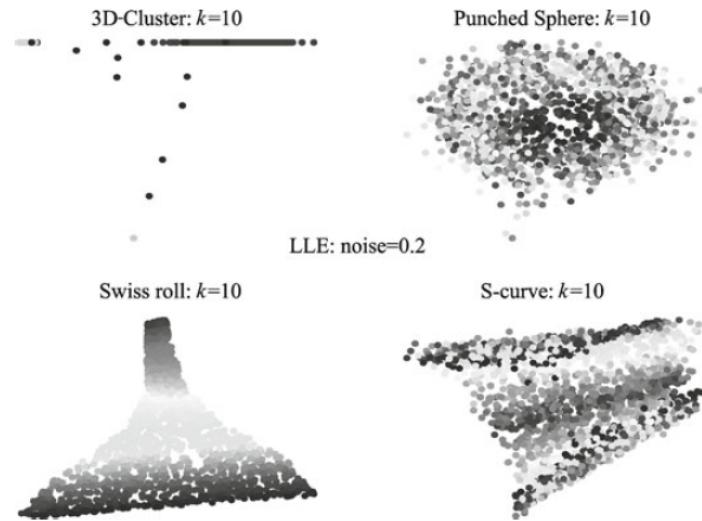


图 5.13: LLE algorithm is applied on 4 surfaces with the noise standard deviation 0.2. All of these surfaces are sampled in the same way as in Fig. 5.12.

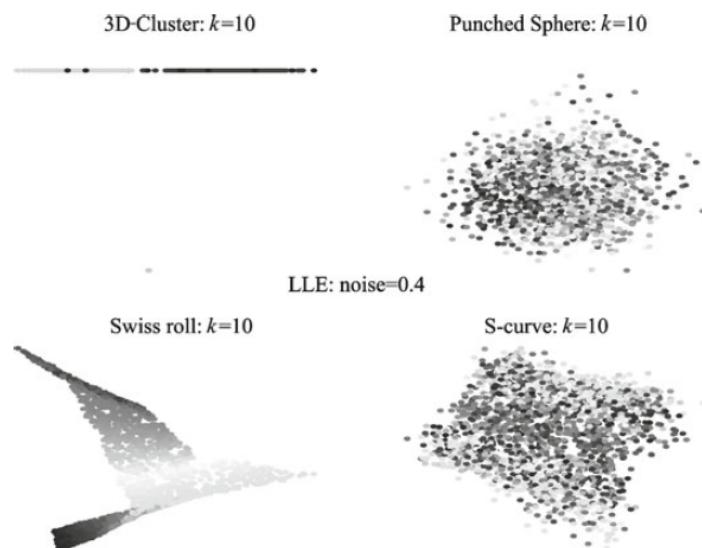


图 5.14: LLE algorithm is applied on 4 surfaces with the noise standard deviation 0.4. All of these surfaces are sampled in the same way as in Fig. 5.13.

$d$ , sometimes the weights are no longer uniquely defined. In this case, some further regularization must be added to break the degeneracy. Finally, the noise will also destroy the local linearity.

Figure 5.15 shows a range of embeddings discovered by the algorithm, all on the same data set, using different numbers of nearest neighbors. The results are stable over a wide range of values but do break down as  $k$  becomes too small or too large. The shapes of surfaces also impact the results. In Fig. 5.16, LLE algorithm is applied to Swiss rolls with lengths 21, 51, and 81, respectively. The number of nearest points,  $k = 10$ , is chosen for the LLE algorithm. The result is not very satisfactory when the length of the roll is too long or too short.

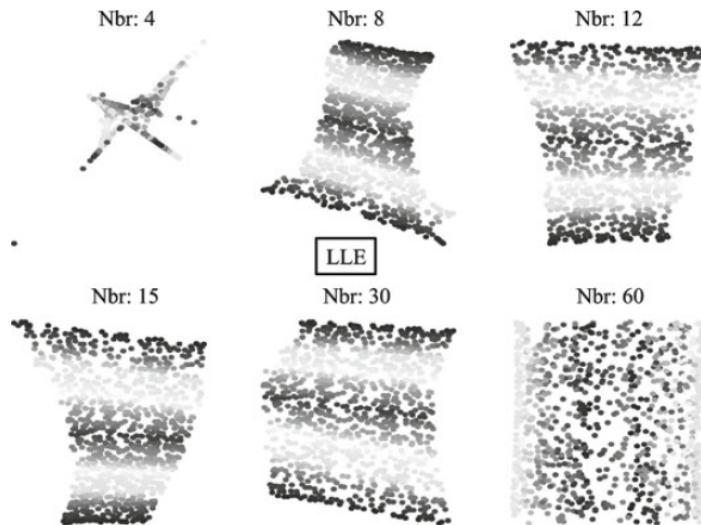


图 5.15: Effect of neighborhood size on LLE. Embeddings of S-curve are computed for different numbers of nearest neighbors, 4, 8, 12, 15, 30, 60, respectively. Reliable embeddings from 3 dimension to 2 dimension are obtained over a wide range of values. If the number is too small (4 on top left) or too large (60 on bottom right), LLE does not succeed in preserving data geometry.

**Conclusions.** Since the kernel of LLE is sparse, the LLE algorithm is fast. The method provides a fairly good linearization of the data on manifolds. However, the algorithm is occasionally computationally unstable. In the neighborhood definition,  $k$ -neighborhood is very common since it does not depend on the data scale and dimension. However, to determine an optimal  $k$  for a given data set is not easy. Small  $k$  usually leads to unstable computation and poor separation, while large  $k$  often causes misclassification. Fortunately, LLE algorithm adopts a wide range of the neighborhood sizes, so that the

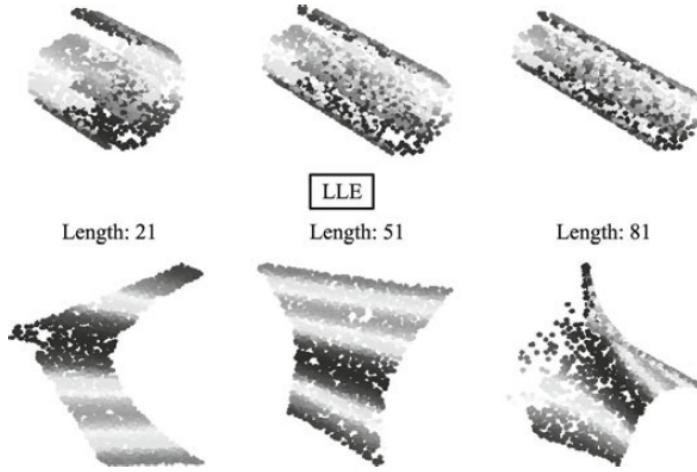


图 5.16: Effect of surface shape on LLE. Embeddings of Swiss-Roll are computed for different choices of the lengths of the roll, 21, 51, and 81, respectively. It is shown that the dimensionality reduction for the Swiss Roll with length 51 is better than the other two.

neighborhood size is not a very sensitive factor for LLE. If  $\varepsilon$ -neighborhood is applied, then it is hard to determine the suitable  $\varepsilon$  for a given data set since it is heavily dependent on the data scale and local geometry. Particularly, when the extrinsic data dimension is high, because of curse of high dimension,  $\varepsilon$ -neighborhood becomes unpractical method for defining data similarity.

### 5.3.6.2 Applications of LLE

**LLE in Image Ordering.** LLE method can be applied in image ordering when the parameters that determine images are nonlinearly related. Saul and Roweis applied LLE to images of lips used in the animation of talking heads. This data set contained  $N = 15960$  color (RGB) images of lips at  $144 \times 152$  resolution ( $D = 65664$ ). Figure 5.17 shows the first two components of these images discovered by LLE with  $K = 24$  nearest neighbors. These components appear to capture highly nonlinear degrees of freedom associated with opening the mouth, pursing the lips, and clenching the teeth. Figure 5.18 shows how one particular neighborhood of lip images is embedded in this two-dimensional space. Dimensionality reduction of these images is useful for faster and more efficient animation. Particularly, the low-dimensional outputs of LLE can be used to index the original collection of high-dimensional images. Fast and accurate indexing is an essential component of example based video synthesis from a large library of stored frames. The data set of lip images is the largest data set to which LLE has been applied

by Saul and Roweis. LLE scales relatively well to large data sets because it generates sparse intermediate results and eigen-problems.

**LLE in Super-resolution.** [29, 145].

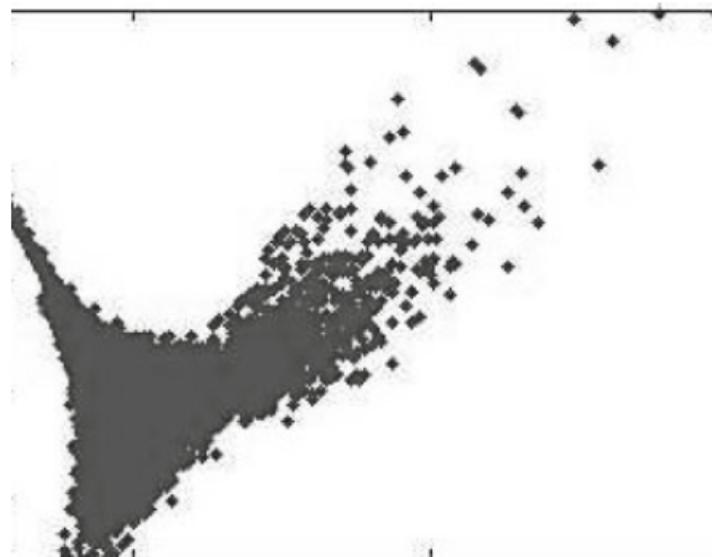


图 5.17: High resolution ( $D = 65664$ ) images of lips, mapped into the embedding space discovered by the first two coordinates of LLE, using  $K = 24$  nearest neighbors. Representative lips are shown at different points in the space. The inset shows the first two LLE coordinates for the entire data set ( $N = 15960$ ) without any corresponding images.

**练习97.** Use one person's face images in Data for MATLAB hackers (<http://www.cs.nyu.edu/~roweis/data.html>) as the data for dimensionality reduction with LLE, where the target dimension is 2, and see whether it is better than CMDS and ISOMAP, e.g., whether the face images are arranged better in some order. Different neighborhood sizes and distance measures are encouraged to be tried.

## 5.4 Local Tangent Space Alignment (LTSA)

In this chapter, a dimensionality reduction embedding method, called local tangent space alignment (LTSA), is introduced. The method is based on the same geometric intuitions as LLE: If a data set is sampled from a smooth manifold, then the neighbors of each point remain nearby and similarly co-located in the low dimensional space. LTSA uses a different approach to the embedded space compared with LLE. In LLE, each point in the data set is linearly embedded into a locally linear patch of the manifold. Then low-dimensional data are constructed so that the locally linear relations of the



图 5.18: A typical neighborhood of  $K = 24$  lip images mapped into the embedding space described by the first two coordinates of LLE. The figure shows a typical neighborhood in the overall space of lip images in Fig. 5.17.

original data are preserved. In LTSA, a locally linear patch is constructed by applying PCA on the neighbors. The patch then can be considered as an approximation of the tangent space at the point. Since the tangent place provides a coordinate representation of the neighbors, the coordinates give a low-dimensional representation of the patch. An alignment technique is introduced in LTSA to align the local representation to a global one.

### 5.4.1 Description of Local Tangent Space Alignment (Taken from [133])

#### 5.4.1.1 Tangent Coordinates and Manifold Coordinates

To describe LTSA, we first review some basis knowledge in manifold calculus, which was introduced in Chapter 3. Let  $M \subset \mathbb{R}^D$  be a  $d$ -dimensional manifold and  $f : U \subset \mathbb{R}^d \rightarrow M$  be a parameterization of  $M$  such that, for each  $\mathbf{x} \in M$ ,

$$\mathbf{x} = f(\mathbf{y}), \quad \mathbf{y} \in U.$$

Let  $h = f^{-1}$  be the coordinate mapping on  $M$  so that  $\mathbf{y} = h(\mathbf{x})$ ,  $\mathbf{x} \in M$ , provides the manifold coordinates of  $\mathbf{x}$ . Assume that a given data set  $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subset \mathbb{R}^D$  lies on the manifold  $M$  and  $\mathbf{y}_i = h(\mathbf{x}_i)$ ,  $1 \leq i \leq n$ . Then the coordinate set  $\mathcal{Y} = \{\mathbf{y}_1, \dots, \mathbf{y}_n\} \subset \mathbb{R}^d$  is a dimensionality reduction of  $\mathcal{X}$ .

The LTSA method finds the DR data set  $\mathcal{Y}$  as follows. It is known that for each  $\mathbf{p} \in M$ , the local geometry of its neighborhood  $O_{\mathbf{p}}$  can be learned from the local tangent place  $T_{\mathbf{p}}M$ . Let  $Q_{\mathbf{p}}$  be the orthogonal projection from  $O_{\mathbf{p}}$  to the tangent place  $T_{\mathbf{p}}M$ . Then we have the approximation

$$Q_{\mathbf{p}}(\mathbf{x} - \mathbf{p}) \approx \mathbf{x} - \mathbf{p}, \quad \forall \mathbf{x} \in O_{\mathbf{p}}. \quad (5.37)$$

Let  $\mathcal{U} = \{\mathbf{u}_1, \dots, \mathbf{u}_d\}$  be an orthonormal basis on  $T_{\mathbf{p}}M$ . Then, we have

$$Q_{\mathbf{p}}(\mathbf{x} - \mathbf{p}) = \sum_{i=1}^d \tau_i \mathbf{u}_i, \quad \tau = (\tau_1, \dots, \tau_d)^T \in \mathbb{R}^d.$$

By the approximation (5.37), we also have

$$\mathbf{x} - \mathbf{p} \approx \sum_{i=1}^d \tau_i \mathbf{u}_i, \quad (5.38)$$

which gives a (local) tangent coordinate representation of  $\mathbf{x}$ . On the other hand, each point  $\mathbf{x} \in M$  has the (manifold) coordinate representation  $\mathbf{x} = f(\mathbf{y})$ ,  $\mathbf{y} \in \mathbb{R}^d$ . Let  $\mathbf{q} = h(\mathbf{p})$  and  $df(\mathbf{q})$  be the derivative of  $f$  at  $q$ . Applying Taylor's expansion, we have

$$\begin{aligned} \mathbf{x} - \mathbf{p} &= df(\mathbf{q})(\mathbf{y} - \mathbf{q}) + O(d_2^2(\mathbf{y}, \mathbf{q})), \\ &\approx df(\mathbf{q})(\mathbf{y} - \mathbf{q}), \end{aligned} \quad (5.39)$$

where  $df(\mathbf{q})$  is an invertible linear transformation. Equation (5.39) provides a (global) manifold coordinate representation of  $\mathbf{x}$ .

Equations (5.38) and (5.39) show that the manifold coordinate  $\mathbf{y}$  of the vector  $\mathbf{x} \in \mathcal{X}$  can be obtained by aligning the local coordinate  $\tau$  of  $\mathbf{x}$  (via the linear transform  $df(\mathbf{q})$ ). The LTSA method realizes dimensionality reduction by finding the local coordinates for the given data, then aligning them with the manifold coordinates. The necessity of the alignment of local tangent coordinates is illustrated in Fig. 5.19.

#### 5.4.1.2 Local Coordinate Representation

In this subsection, we discuss the local coordinate representations of the points in the data set  $\mathcal{X}$ . Assume that on the data set  $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subset M \subset \mathbb{R}^D$  a neighborhood system is well defined and the system generates a graph  $G = (\mathcal{X}, \mathbf{A})$  on  $\mathcal{X}$ . We denote by  $O(i)$  the neighborhood of  $\mathbf{x}_i$  and by  $N(i) = \{i_1, \dots, i_k\}$  the index set of  $O(i)$ . The set  $N(i)$  also corresponds to the non-zero entries in the  $i$ th row of  $\mathbf{A}$ . Let  $k = |N(i)|$ . We define  $\bar{\mathbf{x}} \triangleq \frac{1}{k} \sum_{j \in N(i)} \mathbf{x}_j$ , which is the geometric center of the point set  $O(i)$ . Although  $\bar{\mathbf{x}}$  may not reside on  $M$ , it is very close to  $M$ . For simplicity, we assume that  $\bar{\mathbf{x}} \in M$

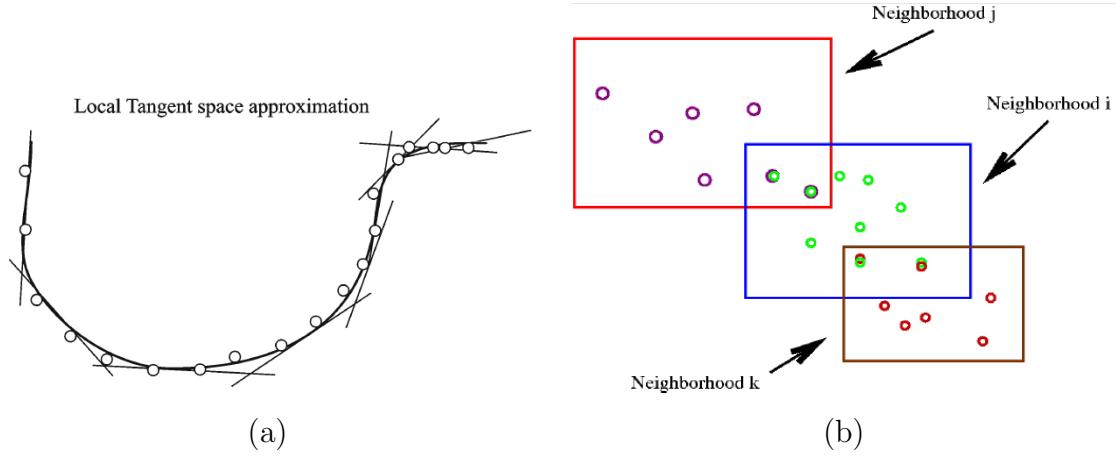


图 5.19: (a) At each point, tangent coordinates are used to represent the local geometry of the data. (b) These coordinates need to align with a global representation.

(otherwise, use its nearest point in  $M$  instead). Denote by  $T_i$  the tangent space of  $M$  at  $\bar{\mathbf{x}}$  and by  $H_i$  the tangent hyperplane  $H_i = \bar{\mathbf{x}} + T_i$ . Since the neighborhood system on  $\mathcal{X}$  is well defined, all points in  $O(i)$  are very close to their projections on the hyperplane  $H_i$ . Let  $F : O(i) \rightarrow H_i$  be the orthogonal projection such that

$$F(\mathbf{x}_j) = \bar{\mathbf{x}} + \mathbf{p}_j, \quad j \in N(i), \mathbf{p}_j \in T_i. \quad (5.40)$$

Due to the properties of linear approximation of  $T_i$ ,

$$\frac{1}{k} \sum_{j \in N(i)} \mathbf{p}_j = \frac{1}{k} \sum_{j \in N(i)} F(\mathbf{x}_j) - \bar{\mathbf{x}} \approx \mathbf{0}.$$

Hence, the vector set  $\{\mathbf{p}_j\}_{j \in N(i)}$  is centered, so that

$$\mathbf{P}_i = \mathbf{P}_i \mathbf{H},$$

where  $\mathbf{H}$  is the  $k \times k$  centralizing matrix and  $\mathbf{P}_i = (\mathbf{p}_{i_1}, \dots, \mathbf{p}_{i_k})$ , where  $i_j \in N(i)$ ,  $j = 1, \dots, k$ . Assume that  $k > d$  and the vector set  $\{\mathbf{p}_j\}_{j \in N(i)}$  spans  $T_i$ . Then we can construct an orthonormal basis of  $T_i$  using the singular value decomposition of the matrix  $\mathbf{P}_i = (\mathbf{p}_{i_1}, \dots, \mathbf{p}_{i_k})$ :

$$\mathbf{P}_i = \mathbf{U} \Sigma \mathbf{V}^T, \quad (5.41)$$

where the column vectors of  $\mathbf{U} = (\mathbf{u}_1, \dots, \mathbf{u}_d) \in \mathcal{O}_{D,d}$  form an orthonormal basis of  $T_i$ . By (5.41),

$$\Theta_i \triangleq \Sigma \mathbf{V}^T = (\tau_{s,j})_{s,j=1}^{d,k}$$

is the local coordinate matrix of  $\mathbf{P}_i$  and

$$\mathbf{p}_j = \sum_{s=1}^d \tau_{s,j} \mathbf{u}_s, \quad j \in N(i). \quad (5.42)$$

In order to compute the local coordinates  $\tau_{s,j}$ , we replace  $\{\mathbf{p}_j\}_{j \in N(i)}$  by its approximation  $\mathcal{X}_i \triangleq \{\mathbf{x}_j - \bar{\mathbf{x}}\}_{j \in N(i)}$ . Write  $\mathbf{X}_i = (\mathbf{x}_{i_1}, \dots, \mathbf{x}_{i_k})$  and let the singular value decomposition of the centered data matrix  $\mathbf{X}_i \mathbf{H}$  be

$$\mathbf{X}_i \mathbf{H} = \mathbf{U}_D \Sigma_D \mathbf{V}_D^T, \quad (5.43)$$

where  $\mathbf{U}_D = (\mathbf{u}_1, \dots, \mathbf{u}_D)$  is an orthonormal matrix,  $\Sigma_D$  is a  $D \times D$  diagonal matrix consisting of all singular values of  $\mathbf{X}_i \mathbf{H}$ , and  $\mathbf{V}_D = (\mathbf{v}_1, \dots, \mathbf{v}_D) \in \mathcal{O}_{k,D}$ . Write  $\mathbf{U}_d = (\mathbf{u}_1, \dots, \mathbf{u}_d)$ ,  $\mathbf{V}_d = (\mathbf{v}_1, \dots, \mathbf{v}_d)$ , and  $\Sigma_d = \text{diag}(\sigma_1, \dots, \sigma_d)$ . By (5.40),  $\mathbf{U}_d \Sigma_d \mathbf{V}_d^T$  is (approximately) equal to  $\mathbf{P}_i$  in (5.41). For convenience, we shall delete the subscript  $d$  in  $\mathbf{U}_d \Sigma_d \mathbf{V}_d^T$  and identify it with (5.41). (5.43) leads to

$$\mathbf{X}_i \mathbf{H} = \mathbf{U} \Sigma \mathbf{V}^T (= \mathbf{U} \Theta_i). \quad (5.44)$$

We now discuss the relation between the local coordinate representation of the set  $\mathcal{X}_i$  and its manifold representation. Applying (5.39) to the set  $\mathcal{X}_i$  with the setting  $\mathbf{p} = \bar{\mathbf{x}}$ , we have

$$\mathbf{x}_j - \bar{\mathbf{x}} = df(\bar{\mathbf{y}})(\mathbf{y}_j - \bar{\mathbf{y}}), \quad (5.45)$$

where  $\bar{\mathbf{y}} = h(\bar{\mathbf{x}})$ . If the approximation is accurate enough, we have  $\bar{\mathbf{y}} = \frac{1}{k} \sum_{j \in N(i)} \mathbf{y}_j$ . Setting  $\mathbf{Y}_i = (\mathbf{y}_{i_1}, \dots, \mathbf{y}_{i_k})$ , we can rewrite (5.45) in the matrix form

$$\mathbf{X}_i \mathbf{H} = df(\bar{\mathbf{y}}) \mathbf{Y}_i \mathbf{H}. \quad (5.46)$$

Recall that  $df(\bar{\mathbf{y}})$  is invertible and  $(df(\bar{\mathbf{y}}))^{-1} = dh(\bar{\mathbf{x}})$ . Hence, we have

$$\mathbf{Y}_i \mathbf{H} = dh(\bar{\mathbf{x}}) \mathbf{X}_i \mathbf{H} = \mathbf{L} \Sigma \mathbf{V}^T, \quad \mathbf{L} \triangleq dh(\bar{\mathbf{x}}) \mathbf{U}. \quad (5.47)$$

So

$$\mathbf{Y}_i \mathbf{H} (\mathbf{I} - \mathbf{V} \mathbf{V}^T) = \mathbf{L} \Sigma \mathbf{V}^T (\mathbf{I} - \mathbf{V} \mathbf{V}^T) = \mathbf{0}. \quad (5.48)$$

#### 5.4.1.3 Global Alignment

To make the global alignment for each local tangent approximation, we first place some constraints on the dimensionality reduction data matrix  $\mathbf{Y}$ . We require that  $\mathbf{Y}$  has zero mean and  $\mathbf{Y} \mathbf{Y}^T = \mathbf{I}$ .

The key of the global alignment is to convert the local relation (5.48) to a global one. Let  $\mathbf{V}$  be the matrix in (5.44). We write  $\mathbf{W}_i = \mathbf{H} (\mathbf{I} - \mathbf{V} \mathbf{V}^T)$ . By (5.48),

$$\mathbf{Y}_i \mathbf{W}_i = \mathbf{0}.$$

We now extend the  $d \times d$  matrix  $\mathbf{W}_i$  to an  $n \times n$  matrix  $\mathbf{W}^i$ , which is defined by

$$\mathbf{W}^i(j, k) = \begin{cases} \mathbf{W}_i(s, l), & \text{if } j = i_s, k = i_l \in N(i), \\ 0, & \text{otherwise.} \end{cases} \quad (5.49)$$

In the similar way, we extend the matrix  $\mathbf{Y}_i$  to a  $k \times n$  matrix  $\mathbf{Y}^i$  such that  $\mathbf{Y}_i$  is the submatrix of  $\mathbf{Y}^i$  with the column indices of  $N(i)$ , and other columns of  $\mathbf{Y}^i$  are zero. The equation (5.48) immediately yields

$$\mathbf{Y}^i \mathbf{W}^i = \mathbf{0}. \quad (5.50)$$

Because non-vanished columns of  $\mathbf{W}^i$  must have the indices in  $N(i)$ , we have

$$\mathbf{Y} \mathbf{W}^i = \mathbf{0}. \quad (5.51)$$

Furthermore, by  $\mathbf{1}^T \mathbf{H} = \mathbf{0}$ , we also have

$$\mathbf{1}^T \mathbf{W}^i = \mathbf{0}. \quad (5.52)$$

Then the data matrix  $\mathbf{Y}$  satisfies (5.51) for all  $i = 1, 2, \dots, n$ . So

$$\mathbf{Y} \mathbf{K} \mathbf{Y}^T = \mathbf{0}, \quad (5.53)$$

and

$$\frac{1}{\sqrt{n}} \mathbf{1}^T \mathbf{K} = \mathbf{0}, \quad (5.54)$$

where

$$\mathbf{K} = \sum_{i=1}^n \mathbf{W}^i \quad (5.55)$$

is the LTSA kernel. Note that (5.51) was obtained by approximation. It may not hold exactly due to possible errors of the approximation and the data deviation. So we may turn to

$$\mathbf{Y} = \underset{\mathbf{Y}}{\operatorname{argmin}} \operatorname{tr}(\mathbf{Y} \mathbf{K} \mathbf{Y}^T), \quad s.t. \quad \mathbf{Y} \mathbf{Y}^T = \mathbf{I}, \mathbf{Y} \mathbf{1} = \mathbf{0}. \quad (5.56)$$

(Going from (5.53) to (5.56): From (5.51) we have that  $\mathbf{Y} \mathbf{K} = \mathbf{0}$ . So  $\mathbf{Y} \mathbf{K}^{1/2} = \mathbf{0}$ . If this does not hold, we want to minimize  $\|\mathbf{Y} \mathbf{K}^{1/2}\|_F^2 = \operatorname{tr}(\mathbf{Y} \mathbf{K} \mathbf{Y}^T)$  instead.)

We have to make sure that  $\mathbf{K}$  is positive semi-definite. To prove this, we consider the matrix  $\mathbf{W}^i$ . It is clear that  $\mathbf{W}^i$  is positive semidefinite if and only if  $\mathbf{W}_i$  is positive semi-definite. By (5.52) and  $\mathbf{P}_i = \mathbf{P}_i \mathbf{H}$ , we have

$$\mathbf{U} \Sigma \mathbf{V}^T = \mathbf{U} \Sigma \mathbf{V}^T \mathbf{H},$$

which yields  $\mathbf{V}^T = \mathbf{V}^T \mathbf{H}$  and therefore,

$$\mathbf{W}_i = \mathbf{H}(\mathbf{I} - \mathbf{V}\mathbf{V}^T) = \mathbf{H} - \mathbf{H}\mathbf{V}\mathbf{V}^T\mathbf{H} = \mathbf{H}\mathbf{I}\mathbf{H} - \mathbf{H}\mathbf{V}\mathbf{V}^T\mathbf{H} = \mathbf{H}(\mathbf{I} - \mathbf{V}\mathbf{V}^T)\mathbf{H}.$$

Since  $\mathbf{V}\mathbf{V}^T$  is a projection matrix,  $\|\mathbf{V}\mathbf{V}^T\| \leq 1$ . Therefore,  $\mathbf{W}_i$  is positive semi-definite. Since  $\mathbf{K}$  is the sum of positive semidefinite matrices,  $\mathbf{K} = \sum_{i=1}^n \mathbf{W}_i$ , it is also positive semidefinite too.

Thus by (5.56), the columns of the matrix  $\mathbf{Y}^T$  are the eigenvectors of  $\mathbf{K}$  corresponding to the 2nd- $(d+1)$ st smallest eigenvalues.

#### 5.4.2 LTSA Algorithm

According to the discussion of the previous section, an LTSA dimensionality reduction algorithm consists of the following steps.

1. **Neighborhood definition.** The neighborhood of each point can be either  $k$ -neighborhood or  $\varepsilon$ -neighborhood. The number of points in a neighborhood is chosen to be greater than  $d$ . We denote by  $G = (\mathcal{X}, \mathbf{A})$  the data graph. Because the value  $\varepsilon$  in  $\varepsilon$ -neighborhood definition depends on the dimension and the scale of the data,  $k$ -neighborhood is more often used in neighborhood definition in LTSA algorithm.
2. **Local coordinate relation computation.** Let  $\mathbf{X}_i = (\mathbf{x}_{i_1}, \dots, \mathbf{x}_{i_k})$  be the matrix whose columns are the  $k$  neighbors of  $\mathbf{x}_i$ . We centralize the data by subtracting their mean:  $\hat{\mathbf{X}}_i = (\mathbf{x}_{i_1} - \bar{\mathbf{x}}, \dots, \mathbf{x}_{i_k} - \bar{\mathbf{x}})$ . The local coordinates (in  $\mathbb{R}^d$ ) of  $\hat{\mathbf{X}}_i$  is found by PCA: Let the  $d$ -rank best approximation of  $\hat{\mathbf{X}}_i$  be  $\sum_{j=1}^d \sigma_j \mathbf{u}_j \mathbf{v}_j^T$ . Write  $\mathbf{V} = (\mathbf{v}_1, \dots, \mathbf{v}_d)$  and set  $\mathbf{G}_i = \left( \frac{1}{\sqrt{k}} \mathbf{1}, \mathbf{V}_i \right) \in \mathcal{O}_{k,d+1}$ . Then  $\mathbf{W}_i = \mathbf{I} - \mathbf{G}_i \mathbf{G}_i^T$ .  
(Checking  $\mathbf{W}_i = \mathbf{I} - \mathbf{G}_i \mathbf{G}_i^T$ :

$$\mathbf{W}_i = (\mathbf{I} - \mathbf{1}\mathbf{1}^T/k)(\mathbf{I} - \mathbf{V}\mathbf{V}^T) = \mathbf{I} - \mathbf{V}\mathbf{V}^T - \mathbf{1}\mathbf{1}^T/k + \mathbf{1}(\mathbf{1}^T \mathbf{V}) \mathbf{V}^T/k.$$

From  $\mathbf{V}^T = \mathbf{V}^T \mathbf{H} = \mathbf{V}^T (\mathbf{I} - \mathbf{1}\mathbf{1}^T/k)$ , we have

$$\mathbf{0} = \mathbf{V}^T \mathbf{1}\mathbf{1}^T \Rightarrow \mathbf{V}^T \mathbf{1} = \mathbf{0}.$$

So

$$\mathbf{W}_i = \mathbf{I} - \mathbf{V}\mathbf{V}^T - \mathbf{1}\mathbf{1}^T/k = \mathbf{I} - \mathbf{G}_i \mathbf{G}_i^T.$$

3. **LTSA kernel construction via global alignment.** The LTSA kernel  $\mathbf{K}$  is the alignment matrix of all local matrices  $\mathbf{W}_i$ . Initialize  $\mathbf{K}$  by a zero matrix. Let  $N(i)$  be the index set of the neighborhood of  $\mathbf{x}_i$  and  $\mathbf{K}(N(i), N(i))$  be the submatrix of  $\mathbf{K}$  containing the rows and columns of the indices  $N(i)$ . Then update  $\mathbf{K}$  by setting  $\mathbf{K}(N(i), N(i)) = \mathbf{K}(N(i), N(i)) + \mathbf{W}_i$ , for  $i = 1, 2, \dots, n$ , successively.
4. **Eigen decomposition of DR kernel.** Let the spectral decomposition of  $\mathbf{K}$  be given by

$$\mathbf{K} = \mathbf{U}\Lambda\mathbf{U}^T,$$

where  $\Lambda = \text{diag}(\lambda_0, \dots, \lambda_{n-1})$  with

$$0 = \lambda_0 \leq \lambda_1 \leq \dots \leq \lambda_{n-1}.$$

Assume that the graph  $G = (\mathcal{X}, \mathbf{A})$  is connected, then  $\lambda_1 > 0$ . Let  $\mathbf{Y} = (\mathbf{u}_1, \dots, \mathbf{u}_d)^T$  be the transpose of the eigenvector matrix corresponding to the 2nd- $(d+1)$ st smallest eigenvalues of  $\mathbf{K}$ . Then  $\mathbf{Y}$  is the DR data set. This step is the same as in LLE algorithm.

### 5.4.3 The Original Description on LTSA (Taken from [160])

#### 5.4.3.1 Manifold Learning and Dimension Reduction

We assume that a  $d$ -dimensional manifold  $\mathcal{F}$  embedded in an  $m$ -dimensional space ( $d < m$ ) can be represented by a function

$$f : C \subset \mathbb{R}^d \rightarrow \mathbb{R}^m,$$

where  $C$  is a compact subset of  $\mathbb{R}^d$  with open interior. We are given a set of data points  $\mathbf{x}_1, \dots, \mathbf{x}_N$ , where  $\mathbf{x}_i \in \mathbb{R}^m$  are sampled possibly with noise from the manifold, i.e.,

$$\mathbf{x}_i = f(\tau_i) + \varepsilon_i, \quad i = 1, \dots, N,$$

where  $\varepsilon_i$  represents noise. By dimension reduction we mean the estimation of the unknown lower dimensional feature vectors  $\tau_i$ 's from the  $\mathbf{x}_i$ 's, i.e., the  $\mathbf{x}_i$ 's which are data points in  $\mathbb{R}^m$  is (nonlinearly) projected to  $\tau_i$ 's which are points in  $\mathbb{R}^d$ , with  $d < m$  we realize the objective of dimensionality reduction of the data points. By manifold learning we mean the reconstruction of  $f$  from the  $\mathbf{x}_i$ 's, i.e., for an arbitrary test point  $\tau \in C \subset \mathbb{R}^d$ , we can provide an estimate of  $f(\tau)$ . These two problems are inter-related, and the solution of one leads to the solution of the other. In some situations, dimension reduction can be the means to an end by itself, and it is not necessary to learn the

manifold. In this paper, however, we promote the notion that both problems are really the two sides of the same coin, and the best approach is not to consider each in isolation. Before we tackle the algorithmic details, we first want to point out that the key difficulty in manifold learning and nonlinear dimension reduction from a sample of data points is that the data points are unorganized, i.e., no adjacency relationship among them are known beforehand. Otherwise, the learning problem becomes the well-researched nonlinear regression problem. To ease discussion, in what follows we will call the space where the data points live the input space, and the space into which the data points are projected the feature space.

To illustrate the concepts and problems we have introduced, we consider the example of linear manifold learning and linear dimension reduction. We assume that the set of data points are sampled from a  $d$ -dimensional affine subspace, i.e.,

$$\mathbf{x}_i = \mathbf{c} + \mathbf{U}\tau_i + \varepsilon_i, \quad i = 1, \dots, N,$$

where  $\mathbf{c} \in \mathbb{R}^m$ ,  $\tau_i \in \mathbb{R}^d$  and  $\varepsilon_i \in \mathbb{R}^m$  represents noise.  $\mathbf{U} \in \mathbb{R}^{m \times d}$  is a matrix consisting of an orthonormal basis of the affine subspace. Let

$$\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N], \quad \mathbf{T} = [\tau_1, \dots, \tau_N], \quad \mathbf{E} = [\varepsilon_1, \dots, \varepsilon_N].$$

Then in matrix form, the data-generation model can be written as

$$\mathbf{X} = \mathbf{c}\mathbf{1}^T + \mathbf{U}\mathbf{T} + \mathbf{E}.$$

The problem of linear manifold learning amounts to seeking  $\mathbf{c}$ ,  $\mathbf{U}$  and  $\mathbf{T}$  to minimize the reconstruction error  $\mathbf{E}$ , i.e.,

$$\min_{\mathbf{c}, \mathbf{U}, \mathbf{T}} \|\mathbf{E}\|_F = \|\mathbf{X} - (\mathbf{c}\mathbf{1}^T + \mathbf{U}\mathbf{T})\|_F.$$

This problem can be readily solved by the singular value decomposition (SVD) based upon the following two observations.

1. The norm of the error matrix  $\mathbf{E}$  can be reduced by removing the mean of the columns of  $\mathbf{E}$  from each column of  $\mathbf{E}$ , and hence one can assume that the optimal  $\mathbf{E}$  has zero mean. This requirement can be fulfilled if  $\mathbf{c}$  is chosen as the mean of  $\mathbf{X}$ , i.e.,  $\mathbf{c} = \mathbf{X}\mathbf{1}/N \equiv \bar{\mathbf{x}}$ .

Proof. To show that the optimal  $\mathbf{c}$  is as above, we have to prove that  $\mathbf{U}\mathbf{T}\mathbf{1} = \mathbf{0}$  because given  $\mathbf{U}\mathbf{T}$ , the optimal  $\mathbf{c} = (\mathbf{X} - \mathbf{U}\mathbf{T})\mathbf{1}/N$ . Indeed, for simplicity we may assume that  $\mathbf{Q}\Sigma\mathbf{V}^T$  is the skinny SVD of  $\mathbf{X}\mathbf{H}$ . Then

$$\mathbf{Q}\Sigma\mathbf{V}^T\mathbf{1} = \mathbf{X}\mathbf{H}\mathbf{1} = \mathbf{0}.$$

So  $\mathbf{V}^T \mathbf{1} = \mathbf{0}$ . Hence  $\mathbf{V}_d^T \mathbf{1} = \mathbf{0}$ , which gives

$$\mathbf{U}\mathbf{T}\mathbf{1} = \mathbf{Q}_d \Sigma_d \mathbf{V}_d^T \mathbf{1} = \mathbf{0}.$$

2. The low-rank matrix  $\mathbf{UT}$  is the optimal rank- $d$  approximation to the centered data matrix  $\mathbf{X} - \bar{\mathbf{x}}\mathbf{1}^T$ . Hence the the optimal solution is given by the SVD of  $\mathbf{X} - \bar{\mathbf{x}}\mathbf{1}^T$ ,

$$\mathbf{X} - \bar{\mathbf{x}}\mathbf{1}^T = \mathbf{Q}\Sigma\mathbf{V}^T, \quad \mathbf{P} \in \mathbb{R}^{m \times m}, \Sigma \in \mathbb{R}^{m \times N}, \mathbf{V} \in \mathbb{R}^{N \times N},$$

i.e.,  $\mathbf{UT} = \mathbf{Q}_d \Sigma_d \mathbf{V}_d^T$ , where  $\Sigma_d = \text{diag}(\sigma_1, \dots, \sigma_d)$  with the  $d$  largest singular values of  $\mathbf{X} - \bar{\mathbf{x}}\mathbf{1}^T$ ,  $\mathbf{Q}_d$  and  $\mathbf{V}_d$  are the matrices of the corresponding left and right singular vectors, respectively. The optimal  $\mathbf{U}^*$  is then given by  $\mathbf{Q}_d$  and the learned linear manifold is represented by the linear function

$$f(\tau) = \bar{\mathbf{x}} + \mathbf{U}^*\tau.$$

In this model, the coordinate matrix  $\mathbf{T}$  corresponding to the data matrix  $\mathbf{X}$  is given by

$$\mathbf{T} = (\mathbf{U}^*)^T (\mathbf{X} - \bar{\mathbf{x}}\mathbf{1}^T) = \text{diag}(\sigma_1, \dots, \sigma_d) \mathbf{V}_d^T.$$

Ideally, the dimension  $d$  of the learned linear manifold should be chosen such that  $\sigma_{d+1} \ll \sigma_d$ .

The function  $f$  is not unique in the sense that it can be reparametrized, i.e., the coordinate can be replaced by  $\tilde{\tau}$  with a global affine transformation  $\tau = \mathbf{P}\tilde{\tau}$ , if we change the basis matrix  $\mathbf{U}^*$  to  $\mathbf{U}^*\mathbf{P}$ . What we are interested in with respect to dimension reduction is the low-dimensional representation of the linear manifold in the feature space. Therefore, without loss of generality, we can assume that the feature vectors are uniformly distributed. For a given data set, this amounts to assuming that the coordinate matrix  $\mathbf{T}$  is orthonormal in row, i.e.,  $\mathbf{T}\mathbf{T}^T = \mathbf{I}_d$ . Hence we we can take  $\mathbf{T} = \mathbf{V}_d^T$  and the linear function is now the following

$$f(\tau) = \bar{\mathbf{x}} + \mathbf{U}^* \text{diag}(\sigma_1, \dots, \sigma_d) \tau.$$

For the linear case we just discussed, the problem of dimension reduction is solved by computing the right singular vectors  $\mathbf{V}_d$ , and this can be done without the help of the linear function  $f$ . Similarly, the construction of the linear function  $f$  is done by computing  $\mathbf{U}^*$  which are just the  $d$  largest left singular vectors of  $\mathbf{X} - \bar{\mathbf{x}}\mathbf{1}^T$ .

The case for nonlinear manifolds is more complicated. In general, the global nonlinear structure will have to come from local linear analysis and alignment. The type of

local geometric information we use is the tangent space at a given point which is constructed from a neighborhood of the given point. The local tangent space provides a low-dimensional linear approximation of the local geometric structure of the nonlinear manifold. What we want to preserve are the local coordinates of the data points in the neighborhood with respect to the tangent space. Those local tangent coordinates will be aligned in the low dimensional space by different local affine transformations to obtain a global coordinate system. In the next subsection we will discuss the local tangent space and global alignment which will then be applied to data points sampled with noise.

#### 5.4.3.2 Local Tangent Space and Its Global Alignment

We assume that  $\mathcal{F}$  is a  $d$ -dimensional manifold in an  $m$ -dimensional space with unknown generating function  $f(\tau)$ ,  $\tau \in \mathbb{R}^d$ , and we are given a data set consists of  $N$   $m$ -dimensional vectors  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N]$ ,  $\mathbf{x}_i \in \mathbb{R}^m$  generated from the following noise-free model,

$$\mathbf{x}_i = f(\tau_i), \quad i = 1, \dots, N,$$

where  $\tau_i \in \mathbb{R}^d$  with  $d < m$ . The objective as we mentioned before for nonlinear dimension reduction is to reconstruct  $\tau_i$ 's from the corresponding function values  $f(\tau_i)$ 's without explicitly constructing  $f$ . Assume that the function  $f$  is smooth enough, using first-order Taylor expansion at a fixed  $\tau$ , we have

$$f(\bar{\tau}) \approx f(\tau) + J_f(\tau) \cdot (\bar{\tau} - \tau), \quad (5.57)$$

where  $J_f(\tau) \in \mathbb{R}^{m \times d}$  is the Jacobi matrix of  $f$  at  $\tau$ .

The tangent space  $\mathcal{T}_\tau$  of  $f$  at  $\tau$  is spanned by the  $d$  column vectors of  $J_f(\tau)$  and is therefore of dimension at most  $d$ , i.e.,  $\mathcal{T}_\tau = \text{span}(J_f(\tau))$ . The vector  $\bar{\tau} - \tau$  gives the coordinate of  $f(\bar{\tau})$  in the affine subspace  $f(\tau) + \mathcal{T}_\tau$ . Without knowing the function  $f$ , we can not explicitly compute the Jacobi matrix  $J_f(\tau)$ . However, if we know  $\mathcal{T}_\tau$  in terms of  $\mathbf{Q}_\tau$ , a matrix forming an orthonormal basis of  $\mathcal{T}_\tau$ , we can write

$$J_f(\bar{\tau} - \tau) = \mathbf{Q}_\tau \theta_\tau^*.$$

Furthermore,

$$\theta_\tau^* = \mathbf{Q}_\tau^T J_f(\tau) (\tau - \bar{\tau}) \equiv \mathbf{P}_\tau (\bar{\tau} - \tau).$$

The mapping from  $\tau$  to  $\theta_\tau^*$  represents a local affine transformation. This affine transformation is unknown because we do not know the function  $f$ . The vector  $\theta_\tau^*$ , however, has an approximate  $\theta_\tau$  that orthogonally projects  $f(\bar{\tau}) - f(\tau)$  onto  $\mathcal{T}_\tau$ ,

$$\theta_\tau \equiv \mathbf{Q}_\tau^T (f(\bar{\tau}) - f(\tau)) \approx \theta_\tau^*, \quad (5.58)$$

provided  $\mathbf{Q}_\tau$  is known at each  $\tau$ . So the global coordinate  $\tau$  satisfies

$$\int d\tau \int_{\Omega(\tau)} \|\mathbf{P}_\tau(\bar{\tau} - \tau) - \theta_\tau\|^2 d\bar{\tau} \approx 0.$$

Here  $\Omega(\tau)$  defines the neighborhood of  $\tau$ . Therefore, a natural way to approximate the global coordinate is to find a global coordinate  $\tau$  and a local affine transformation  $\mathbf{P}_\tau$  that minimize the error function

$$\int d\tau \int_{\Omega(\tau)} \|\mathbf{P}_\tau(\bar{\tau} - \tau) - \theta_\tau\|^2 d\bar{\tau}. \quad (5.59)$$

This represents a nonlinear alignment approach for the dimension reduction problem.

On the other hand, a linear alignment approach can be devised as follows. If  $J_f(\tau)$  is of full column rank, the matrix  $\mathbf{P}_\tau$  should be non-singular and

$$\bar{\tau} - \tau \approx \mathbf{P}_\tau^{-1} \theta_\tau \equiv \mathbf{L}_\tau \theta_\tau.$$

The above equation shows that the affine transformation  $\mathbf{L}_\tau$  should align this local coordinate with the global coordinate  $\bar{\tau} - \tau$  for  $f(\tau)$ . Naturally we should seek to find a global coordinate  $\tau$  and a local affine transformation  $\mathbf{L}_\tau$  to minimize

$$\int d\tau \int_{\Omega(\tau)} \|\bar{\tau} - \tau - \mathbf{L}_\tau \theta_\tau\|^2 d\bar{\tau}. \quad (5.60)$$

The above amounts to matching the local geometry in the feature space. Notice that  $\theta_\tau$  is defined by the “known” function value and the “unknown” orthogonal basis matrix  $\mathbf{Q}_\tau$  of the tangent space. It turns out, however,  $\mathbf{Q}_\tau$  can be approximately determined by certain function values. We will discuss this approach in the next subsection. Clearly, this linear approach is more readily applicable than (5.59). Obviously, If the manifold  $\mathcal{F}$  is not regular, i.e., the Jacobi matrix  $J_f$  is not of full column rank at some points  $\tau \in C$ , then the two minimization problems (5.60) and (5.59) may lead to quite different solutions. As is discussed in the linear case, the low-dimensional feature vector  $\tau$  is not uniquely determined by the manifold  $\mathcal{F}$ . We can reparametrize  $\mathcal{F}$  using  $f(g(\tau))$  where  $g(\cdot)$  is a smooth 1-to-1 onto mapping of  $C$  to itself. The parameterization of  $\mathcal{F}$  can be fixed by requiring that  $\tau$  has a uniform distribution over  $C$ . This will come up as a normalization issue in the next subsection.

#### 5.4.3.3 Feature Extraction through Alignment

Now we consider how to construct the global coordinates and local affine transformation when we are given a data set  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N]$  sampled with noise from an underlying nonlinear manifold,

$$\mathbf{x}_i = f(\tau_i) + \varepsilon_i, \quad i = 1, \dots, N,$$

where  $\tau \in \mathbb{R}^d$ ,  $\mathbf{x}_i \in \mathbb{R}^m$  with  $d < m$ . For each  $\mathbf{x}_i$ , let  $\mathbf{X}_i = [\mathbf{x}_{i_1}, \dots, \mathbf{x}_{i_k}]$  be a matrix consisting of its  $k$ -nearest neighbors including  $\mathbf{x}_i$ , say in terms of the Euclidean distance. Consider computing the best  $d$ -dimensional affine subspace approximation for the data points in  $\mathbf{X}_i$ ,

$$\min_{\mathbf{x}, \mathbf{Q}, \Theta} \sum_{j=1}^k \|\mathbf{x}_{i_j} - (\mathbf{x} + \mathbf{Q}\theta_j)\|_2^2 = \min_{\mathbf{x}, \mathbf{Q}, \Theta} \|\mathbf{X}_i - (\mathbf{x}\mathbf{1}^T + \mathbf{Q}\Theta)\|_F^2,$$

where  $\mathbf{Q}$  is of  $d$  columns and is orthonormal, and  $\Theta = [\theta_1, \dots, \theta_k]$ . As is discussed in subsection 5.4.3.1, the optimal  $\mathbf{x}$  is given by  $\bar{\mathbf{x}}_i$ , the mean of all the  $\mathbf{x}_{i_j}$ 's and the optimal  $\mathbf{Q}$  is given by  $\mathbf{Q}_i$ , the  $d$  left singular vectors of  $\mathbf{X}_i(\mathbf{I} - \mathbf{1}\mathbf{1}^T/k)$  corresponding to its  $d$  largest singular values, and  $\Theta$  is given by  $\Theta_i$  defined as

$$\Theta_i = \mathbf{Q}_i^T \mathbf{X}_i (\mathbf{I} - \mathbf{1}\mathbf{1}^T/k) = [\theta_1^{(i)}, \dots, \theta_k^{(i)}], \quad \theta_j^{(i)} = \mathbf{Q}_i^T (\mathbf{x}_{i_j} - \bar{\mathbf{x}}_i). \quad (5.61)$$

Therefore we have

$$\mathbf{x}_{i_j} = \bar{\mathbf{x}}_i + \mathbf{Q}_i \theta_j^{(i)} + \xi_j^{(i)}, \quad (5.62)$$

where  $\xi_j^{(i)} = (\mathbf{I} - \mathbf{Q}_i \mathbf{Q}_i^T)(\mathbf{x}_{i_j} - \bar{\mathbf{x}}_i)$  denotes the reconstruction error.

(This step is to solve for  $\Theta_i$ .  $\mathbf{Q}_i$  is not used in the sequel.)

We now consider constructing the global coordinates  $\tau_i$ ,  $i = 1, \dots, N$ , in the low-dimensional feature space based on the local coordinates  $\theta_j^{(i)}$  which represents the local geometry. Specifically, we want  $\tau_{i_j}$  to satisfy the following set of equations, i.e., the global coordinates should respect the local geometry determined by the  $\theta_j^{(i)}$ ,

$$\tau_{i_j} = \bar{\tau}_i + \mathbf{L}_i \theta_j^{(i)} + \varepsilon_j^{(i)}, \quad j = 1, \dots, k, i = 1, \dots, N, \quad (5.63)$$

where  $\bar{\tau}_i$  is the mean of  $\tau_{i_j}$ ,  $j = 1, \dots, k$ . In matrix form,

$$\mathbf{T}_i = \frac{1}{k} \mathbf{T}_i \mathbf{1} \mathbf{1}^T + \mathbf{L}_i \Theta_i + \mathbf{E}_i,$$

where  $\mathbf{T}_i = [\tau_{i_1}, \dots, \tau_{i_k}]$  and  $\mathbf{E}_i = [\varepsilon_1^{(i)}, \dots, \varepsilon_k^{(i)}]$  is the local reconstruction error matrix, and we write

$$\mathbf{E}_i = \mathbf{T}_i (\mathbf{I} - \mathbf{1} \mathbf{1}^T/k) - \mathbf{L}_i \Theta_i. \quad (5.64)$$

To preserve as much of the local geometry in the low-dimensional feature space, we seek to find  $\tau_i$  and the local affine transformations  $\mathbf{L}_i$  to minimize the reconstruction errors  $\varepsilon_j^{(i)}$ , i.e.,

$$\min_{\mathbf{T}, \{\mathbf{L}_i\}} \sum_i \|\mathbf{E}_i\|_F^2 = \sum_i \|\mathbf{T}_i (\mathbf{I} - \mathbf{1} \mathbf{1}^T/k) - \mathbf{L}_i \Theta_i\|_F^2. \quad (5.65)$$

(This step approximates (5.60) in a discrete way.)

Obviously, the optimal alignment matrix  $\mathbf{L}_i$  that minimizes the local reconstruction error  $\|\mathbf{E}_i\|_F$  for a fixed  $\mathbf{T}_i$ , is given by

$$\mathbf{L}_i = \mathbf{T}_i(\mathbf{I} - \mathbf{1}\mathbf{1}^T/k)\Theta_i^+$$

and therefore

$$\mathbf{E}_i = \mathbf{T}_i(\mathbf{I} - \mathbf{1}\mathbf{1}^T/k)(\mathbf{I} - \Theta_i^+\Theta_i),$$

where  $\Theta_i^+$  is the Moore-Penrose generalized inverse of  $\Theta_i$ . Let  $\mathbf{T} = [\tau_1, \dots, \tau_N]$  and  $\mathbf{S}_i$  be the 0-1 selection matrix such that  $\mathbf{T}\mathbf{S}_i = \mathbf{T}_i$ . We then need to find  $\mathbf{T}$  to minimize the overall reconstruction error

$$\min_{\mathbf{T}} \sum_i \|\mathbf{E}_i\|_F^2 = \|\mathbf{T}\mathbf{S}\mathbf{W}\|_F^2,$$

where  $\mathbf{S} = [\mathbf{S}_1, \dots, \mathbf{S}_N]$  and  $\mathbf{W} = \text{diag}(\mathbf{W}_1, \dots, \mathbf{W}_N)$  with

$$\mathbf{W}_i = (\mathbf{I} - \mathbf{1}\mathbf{1}^T/k)(\mathbf{I} - \Theta_i^+\Theta_i). \quad (5.66)$$

(This step aligns the local tangent spaces.)

To uniquely determine  $\mathbf{T}$ , we will impose the constraints  $\mathbf{T}\mathbf{T}^T = \mathbf{I}_d$ . It turns out that the vector  $\mathbf{1}$  of all ones is an eigenvector of

$$\mathbf{B} = \mathbf{S}\mathbf{W}\mathbf{W}^T\mathbf{S}^T \quad (5.67)$$

corresponding to a zero eigenvalue, therefore, the optimal  $\mathbf{T}$  is given by the  $d$  eigenvectors of the matrix  $\mathbf{B}$ , corresponding to the 2nd to  $(d+1)$ -st smallest eigenvalues of  $\mathbf{B}$ .

#### 5.4.3.4 Constructing Principal Manifolds

Once the global coordinates  $\tau_i$  are computed for each of the data points  $\mathbf{x}_i$ , we can apply some non-parametric regression methods such as local polynomial regression to  $\{(\tau_i, \mathbf{x}_i)\}_{i=1}^N$  to construct the principal manifold underlying the set of points  $\mathbf{x}_i$ . Here each of the component functions  $f_j(\tau)$  can be constructed separately.

In general, when the low-dimensional coordinates  $\tau_i$  are available, we can construct a mapping from the  $\tau$ -space (feature space) to the  $\mathbf{x}$ -space (input space) as follows.

1. For each fixed  $\tau$ , let  $\tau_i$  be the nearest neighbor (i.e.,  $\|\tau - \tau_i\| \leq \|\tau - \tau_j\|$ , for  $j \neq i$ ). Define

$$\theta = \mathbf{L}_i^{-1}(\tau - \bar{\tau}_i),$$

where  $\bar{\tau}_i$  be the mean of the feature vectors in a neighbor to which  $\tau_i$  belongs.

2. Back in the input space, we define

$$\mathbf{x} = \bar{\mathbf{x}}_i + \mathbf{Q}_i\theta.$$

#### 5.4.4 Test LTSA on Artificial Surfaces

In Fig. 5.20, LTSA algorithm is applied to reduce the data of 4 artificial surfaces, 3D-cluster, Punched sphere, Swiss roll, and S-curve to 2-dimensional data, where 10 nearest points are used to construct the neighborhood. The experiment shows that LTSA works well for these surfaces, except 3D-cluster. Note that the data set of 3D-cluster lacks the surface structure on the line segments. Compared to LLE, LTSA does the better job on Swiss roll.

Noise on data impacts the dimensionality reduction results very much. Because LTSA adopts the local approximation using tangent space, it is more sensitive to noise. When we apply LTSA algorithm to noise data with the standard deviation 0.2 and 0.4, we cannot obtain the valid DR data for these surfaces. When we reduce the noise level to 0.1, the DR data are valid for Swiss roll and S-curve only. In Fig. 5.21, the two valid DR results are displayed.

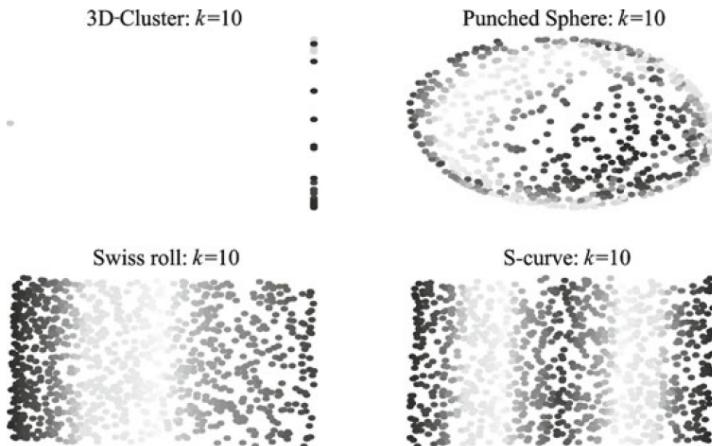


图 5.20: LTSA algorithm is tested on 4 surfaces: 3D-cluster, Punched sphere, Swiss roll, and S-curve. 1000 points are randomly sampled on each surface and 10-neighborhood is chosen to construct the data graphs. All of these surfaces are reduced to 2-dimensional data (on the plane). The figure displays the DR results of these surfaces. The original 3-dimensional graphs are not displayed.

LTSA algorithm is relatively sensitive to the neighborhood sizes, only stable over a relatively narrow range. The range depends on various features of the data, such as the sampling density and the manifold geometry. The experiment shows that when we perform LTSA for the above 4 surfaces, it is unstable if the neighborhood size is less than 7. It is also shown that when the size is larger than 12, the DR data distorts the local geometry of the original data. The results are displayed in Fig. 5.22.

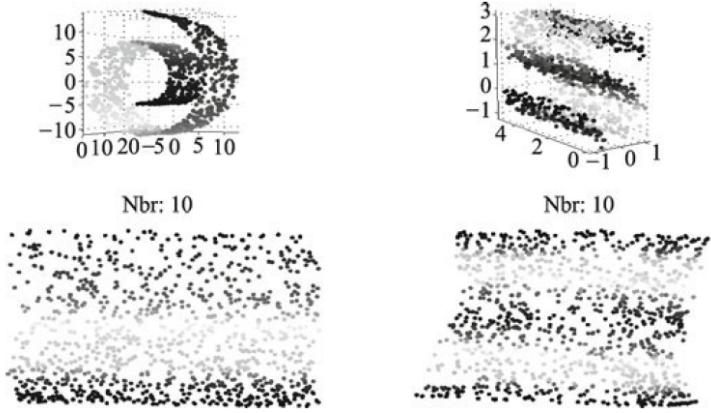


图 5.21: LTSA algorithm is applied on 2 surfaces with the noise standard deviation 0.1. The algorithm is relatively sensitive to noise. When the noise level is high, we cannot obtain valid DR results. Even though the noise standard deviation is already reduced to 0.1, the algorithm can only produce the valid results for Swiss roll and S-curve. The settings for the test are the same as above: 1000 data points are randomly sampled on each surface and 10-neighborhood is used for the construction of data graphs. Top left: Swiss roll. Top right: S-curve. Their corresponding DR data sets are displayed on the bottom line.

The shape of the surface also impacts the results of LTSA dimensionality reduction. In Fig. 5.23, LTSA algorithm is applied to Swiss rolls with the lengths 21, 51, and 81, respectively. The number of nearest points  $K = 10$  is chosen for the experiment. The result is not very satisfied when the length of the roll is too long or too short. In Fig. 5.24 LTSA algorithm is applied to Swiss rolls with the lengths 21, 51, and 81, respectively. The results show that when the length is 81, the DR result is not valid. This phenomenon can be explained as follows. When the length of the roll is 81 while the sample points keep the same number of 1000 as for the short roll of length 21, the distances between the points in a neighborhood for the long roll are four times of those for the short roll in average. Therefore, the graph neighborhood of the long roll may not reside on the manifold neighborhood so that the DR result cannot preserve the geometric structure of the manifold. In Fig. 5.25, the different lengths of S-curve, 5, 10, 30, are chosen for the test with the same parameter settings as in Fig. 11.2. By the same argument we have made for the Swiss rolls in Fig. 5.24, the distortion of the DR data for the long S-curve is perhaps due to the neighborhood inconsistence.

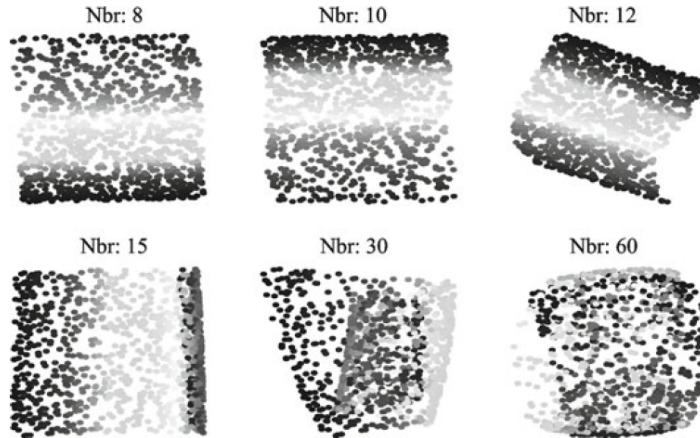


图 5.22: Effect of neighborhood size on LTSA. LTSA is quite sensitive to the neighborhood size. In this figure, LTSA is applied to Swiss roll for different choices of the number of nearest neighbors  $K$ , 8, 10, 12, 15, 30, 60, respectively. When  $K$  is chosen to be less than 7, the algorithm becomes unstable and no DR result is produced. Unlike LLE, the reliable LTSA embedding from 3 dimension to 2 dimension can only be obtained in a narrow range of the sizes. If  $K$  is too small (less than 7), then no valid results can be produced. If  $K$  is a little larger, say, larger than 14, LTSA does not succeed in preserving data geometry.

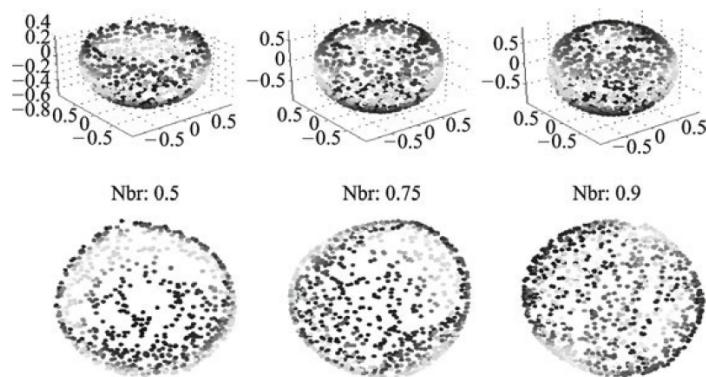


图 5.23: LTSA algorithm is performed on the punched spheres with various heights. On the top line, we present three punched spheres with heights 0.5, 0.75, and 0.9 respectively. The diameter of the sphere is 1. On the bottom line are their DR data sets correspondingly. The results show that the DR data sets do not preserve the local geometry near the boundary when the punched holes become smaller.

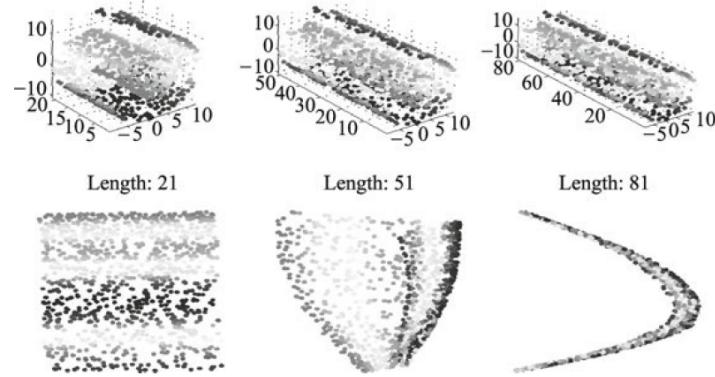


图 5.24: Effect of surface size on LTSA. LTSA embeddings of Swiss-Roll are computed for different choices of the lengths of the roll, 21, 51, 81, respectively. It is shown that the dimensionality reduction for the Swiss roll with length 51 is greater than other two.

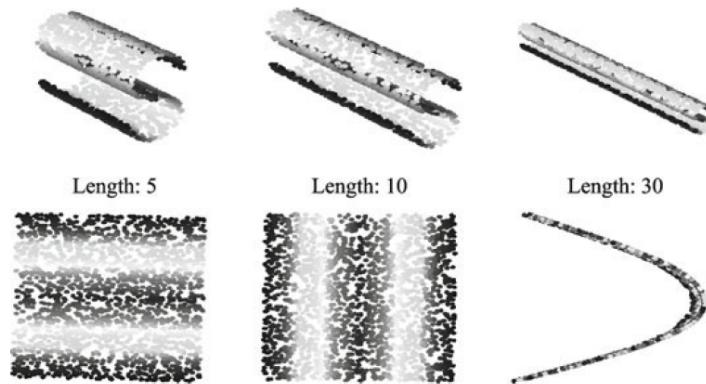


图 5.25: The figure illustrates again that when the graph neighborhood is inconsistent with manifold, the DR data may distort the geometric structure so that it becomes invalid. In this test, LTSA embeddings of the S-curves are computed for different choices of the lengths, 5, 10, and 30, respectively. The surface data are sampled in the same way as in Fig. 5.20. The results show that the dimensionality reduction becomes worse as the length increases, due to the neighborhood inconsistency.

### 5.4.5 Laplacian PCA (LPCA)

To better understand the technique of global alignment, here I introduce my ICCV2007 paper “Laplacian PCA and Its Applications” [162].

#### 5.4.5.1 Local LPCA

For non-Gaussian or manifold-valued data, we usually deal with it from local patches because non-Gaussian data can be viewed locally Gaussian and a curved manifold can be locally viewed Euclidean. Particularly, Gaussian distribution is the theoretical base of many statistical operations, and tangent spaces and tangent coordinates are the fundamental descriptors of a manifold. So, we begin with local LPCA.

Specifically, let  $\alpha_i$  denote the local scatter on the  $i$ -th neighborhood  $\mathcal{S}_i^y$ . It is defined as

$$\alpha_i = \sum_{k=0}^K w_{ik} \|\mathbf{y}_{ik} - \bar{\mathbf{y}}_i\|^2, \quad (5.68)$$

where  $w_{ik}$  is the related weight and  $\bar{\mathbf{y}}_i$  is the geometric centroid of  $\mathcal{S}_i^y$ , i.e.,  $\bar{\mathbf{y}}_i = (\mathbf{Y}_i \mathbf{W}_i \mathbf{1}) / (\mathbf{1}^T \mathbf{W}_i \mathbf{1})$ , where  $\mathbf{W}_i = \text{diag}(w_{i0}, \dots, w_{iK})$ .  $w_{ik}$  could be chosen as  $\exp(-\|\mathbf{x}_i - \mathbf{x}_k\|/\sigma)$ , but other possibilities exist. The distance between  $\mathbf{y}_{ik}$  and  $\bar{\mathbf{y}}_i$  are measured by the  $\ell_2$  norm. Rewriting (5.68) yields

$$\alpha_i = \sum_{k=0}^K w_{ik} \text{tr}((\mathbf{y}_{ik} - \bar{\mathbf{y}}_i)(\mathbf{y}_{ik} - \bar{\mathbf{y}}_i)^T) \quad (5.69)$$

$$= \text{tr}(\mathbf{Y}_i \mathbf{W}_i \mathbf{Y}_i^T) - \text{tr}(\mathbf{Y}_i \mathbf{W}_i \mathbf{1} \mathbf{1}^T \mathbf{W}_i \mathbf{Y}_i^T) / (\mathbf{1}^T \mathbf{W}_i \mathbf{1}). \quad (5.70)$$

Thus we obtain

$$\alpha_i = \text{tr}(\mathbf{Y}_i \mathbf{L}_i \mathbf{Y}_i^T), \quad (5.71)$$

where

$$\mathbf{L}_i = \mathbf{W}_i - \mathbf{W}_i \mathbf{1} \mathbf{1}^T \mathbf{W}_i^T / (\mathbf{1}^T \mathbf{W}_i \mathbf{1}) \quad (5.72)$$

is called the local Laplacian scatter matrix. We want to find low-dimensional embeddings  $\mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_n]$  such that

$$\sum_{i=1}^n \alpha_i \quad (5.73)$$

is maximized.

#### 5.4.5.2 Alignment of Local Geometry

If we aim at deriving the global  $\mathbf{Y}$ , then global analysis can be performed on the alignment of localities. For the traditional approach, the Gaussian mixing model (GMM),

along with the EM scheme, is usually applied to fulfill this task (probabilistic PCA for instance). For spectral methods however, there has a simple approach. Here, we present a unified framework of alignment for spectral methods, by which the optimal solution in closed form can be obtained by eigen-analysis.

For each  $\mathcal{S}_i^y$ , we have  $\mathcal{S}_i^y \subset \mathcal{S}^y$ , meaning that  $\{\mathbf{y}_{i_0}, \mathbf{y}_{i_1}, \dots, \mathbf{y}_{i_K}\}$  are always selected from  $\{\mathbf{y}_1, \dots, \mathbf{y}_n\}$ . What is more, the selection labels are known from the process of nearest neighbors searching. Thus, we can write  $\mathbf{Y}_i = \mathbf{YS}_i$ , where  $\mathbf{S}_i$  is the  $n \times (K + 1)$  binary selection matrix associated with  $\mathcal{S}_i^y$ . Let  $I_i = \{i_0, i_1, \dots, i_K\}$  denote the label set. It is not hard to know that the structure of  $\mathbf{S}_i$  can be expressed by

$$(\mathbf{S}_i)_{pq} = \begin{cases} 1, & \text{if } p = i_{q-1}, \quad i_{q-1} \in I_i, q = 1, \dots, K + 1, \\ 0, & \text{otherwise,} \end{cases} \quad (5.74)$$

meaning that  $(\mathbf{S}_i)_{pq} = 1$  if the  $q$ -th vector in  $\mathbf{Y}_i$  is the  $p$ -th vector in  $\mathbf{Y}$ . Then rewriting (5.71) gives

$$\alpha_i = \text{tr}(\mathbf{YS}_i \mathbf{L}_i \mathbf{S}_i^T \mathbf{Y}^T). \quad (5.75)$$

So (5.73) becomes

$$\max_{\mathbf{Y}} \text{tr}(\mathbf{YL} \mathbf{Y}^T), \quad (5.76)$$

where  $\mathbf{L} = \sum_{i=1}^n \mathbf{S}_i \mathbf{L}_i \mathbf{S}_i^T$  is called the global Laplacian scatter matrix. The expression of  $\mathbf{L}$  implies that, initialized by a zero matrix of the same size,  $\mathbf{L}$  can be obtained by the update  $\mathbf{L}(I_i, I_i) \leftarrow \mathbf{L}(I_i, I_i) + \mathbf{L}_i, i = 1, \dots, n$ .

Problem (5.76) has to be solved by preventing  $\mathbf{Y}$  being a zero matrix, e.g., under the constraint

$$\mathbf{YY}^T = \mathbf{I}.$$

#### 5.4.5.3 LPCA

For LPCA, our goal is to derive a global projection matrix  $\mathbf{U}$ . To this end, we need to plug  $\mathbf{Y} = \mathbf{U}^T(\mathbf{XH})$  in (5.76) to derive the expression of the global scatter when the global Laplacian scatter matrix is ready. Thus we obtain the following maximization problem

$$\max_{\mathbf{U}} \text{tr}(\mathbf{U}^T \mathbf{XHLHX}^T \mathbf{U}), \quad s.t. \quad \mathbf{U}^T \mathbf{U} = \mathbf{I}. \quad (5.77)$$

$\mathbf{U}$  can be achieved by the eigen-decomposition of  $\mathbf{XHLHX}^T$ .

#### 5.4.6 Connection between LLE and LTSA

Please see [161].

### 5.4.7 Conclusion

Similar to LLE, the LTSA kernel is sparse (wrong!), and hence LTSA is a fast algorithm. Because LTSA uses tangent spaces to approximate the data, it usually produces satisfactory DR results for the data sampled on smooth and connected manifolds. For non-smooth surfaces, such as 3D-cluster, it cannot produce the valid DR result. Furthermore, since LTSA is quite sensitive to the smoothness, when the data are contaminated with noise, the algorithm becomes unstable. The algorithm is also relatively sensitive to the neighborhood size. Hence, when LTSA algorithm is adopted, the neighborhood size has to be carefully chosen.

**练习98.** Use one person's face images in Data for MATLAB hackers (<http://www.cs.nyu.edu/~roweis/data.html>) as the data for dimensionality reduction with LTSA, where the target dimension is 2, and see whether it is better than CMDs, ISOMAP, and LLE, e.g., whether the face images are arranged better in some order. Different neighborhood sizes and distance measures are encouraged to be tried.

Use the above data to check whether the method described in section 5.4.2 is the same as that in section 5.4.3.3, i.e.,  $\mathbf{Y} = \mathbf{T}$ ?

## 5.5 Laplacian Eigenmaps (LE)

Laplacian eigenmaps (LE) method is based on the idea of manifold unsupervised learning. Let  $\mathcal{X}$  be the observed high-dimensional data, which reside on a low-dimentional manifold  $M$ . Let  $h$  be the coordinate mapping on  $M$  so that  $\mathcal{Y} = h(\mathcal{X})$  is a DR of  $\mathcal{X}$ . Each component of the coordinate mapping  $h$  is a linear function on  $M$ . Hence, all components of  $h$  nearly reside on the numerically null space of the Laplace-Beltrami operator on  $M$ . In LE method, a Laplace-Beltrami operator is constructed on the data graph. Then the DR data set is derived from the eigenvectors of the operator corresponding to several smallest eigenvalues.

A better explanation of the motivation of LE: Suppose  $h^i(x)$  be the  $i$ -th coordinating mapping on  $M$ . We need  $h^i(x)$  to be smooth so that nearby points on  $M$  are mapped to nearby points in  $\mathbb{R}^d$ . So we need to find the functions that minimize  $\int_M \|\nabla f\|^2$ .

### 5.5.1 Description of the Laplacian Eigenmap Method

The Laplacian eigenmaps method (LE) was introduced by Belkin and Niyogi, using the idea of the manifold learning. While the idea of Laplacian eigenmaps method is similar to that of LLE, its DR kernel is derived from the Laplace-Beltrami operator on

the manifold, where the data set resides. To describe the method, we adopt the same data model as before. Let  $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subset \mathbb{R}^D$  be the data set, which resides on an underlying  $d$ -dimensional manifold  $M \subset \mathbb{R}^D$ . Let  $g$  be a parametrization of  $M$  and  $h = g^{-1} = (h^1, \dots, h^d)^T$  be the coordinate mapping on  $M$  such that for each  $\mathbf{x} \in M$ ,  $\mathbf{y} = h(\mathbf{x}) \in \mathbb{R}^d$  provides  $d$  coordinates for  $\mathbf{x}$ . Then the data set

$$\mathcal{Y} = \{\mathbf{y}_1, \dots, \mathbf{y}_n\}, \quad \mathbf{y}_i = h(\mathbf{x}_i), 1 \leq i \leq n \quad (5.78)$$

is the DR set. To find  $\mathcal{Y}$ , we have to learn the coordinate mapping from the observed data set  $\mathcal{X}$ , assuming that a neighborhood system is given on  $\mathcal{X}$ . Let  $G = (\mathcal{X}, \mathbf{A})$  be the graph on  $\mathcal{X}$ , which defines the neighborhood system. We assume that the graph  $G$  is undirected, i.e.,  $\mathbf{A}$  is symmetric, and the neighborhood system on  $\mathcal{X}$  is consistent with the geometry on  $M$ . Let  $O_i$  be the graph neighborhood of  $\mathbf{x}_i$ . Since the neighborhood is well defined, each point  $\mathbf{x}_j \in O_i$  is near  $\mathbf{x}_i$  on  $M$ .

The idea of Laplacian eigenmaps method can be simply explained as follows. Let  $C^2(M)$  be the space of twice differentiable functions on  $M$ , and  $\mathcal{L}$  be the Laplace-Beltrami operator on  $C^2(M)$ , which is defined by (see Section 3.1.6):

$$\mathcal{L}(u) = -\text{div}(\text{grad}(u)), \quad u \in C^2(M).$$

Let  $e$  denote the constant function on  $M$  with  $e(\mathbf{x}) = 1$ ,  $\mathbf{x} \in M$ . Then  $e$  is in the null space of the Laplace-Beltrami operator  $\mathcal{L}$ . Therefore, it can be obtained as the solution of the equation

$$\mathcal{L}(f) = 0. \quad (5.79)$$

Since each coordinate function  $h^l$  is linear on  $M$ , the totality of  $h^l$ ,  $1 \leq l \leq d$ , can be approximately represented by the linear combinations of achieving the  $d$  smallest positive eigenvalues. They are called feature functions. In LE method, the DR data are derived from these feature functions instead of the coordinate ones.

In general, data sets may be contaminated by noise, and there are many other facts which may disturb the solutions. Hence, the equation model (5.79) needs to be slightly modified. The relation (see Section 3.1.6):

$$\int_M f \mathcal{L}(f) = \int_M \|\nabla f\|^2 \quad (5.80)$$

inspires us to modify the equation model (5.79) to the following minimization model

$$h = \operatorname{argmin}_{f \in \mathcal{F}} \int_M f \mathcal{L}(f) \quad (5.81)$$

under certain constraints. LE method adopts the model (5.81) to find the DR set  $\mathcal{Y}$ .

### 5.5.2 Approximation of Laplace-Beltrami Operator

The key step of LE method is to construct the Laplace-Beltrami operator  $\mathcal{L}$ . By (5.80), the Laplace-Beltrami operator  $\mathcal{L}$  is positive and self-adjoint. Applying the exponential formula to positive and self-adjoint operators, we can construct a single-parametric semi-group

$$\mathcal{A}_t = \exp(-t\mathcal{L}), \quad (5.82)$$

where  $\mathcal{A}_t$  is called the *Neumann heat diffusion operator* on  $M$  having the following integration expression:

$$\mathcal{A}_t(f)(\mathbf{x}) = \int_M G_t(\mathbf{x}, \mathbf{y}) f(\mathbf{y}), \quad f \in C(M). \quad (5.83)$$

In (5.83), the integral kernel  $G_t$  is known as the Neumann heat kernel, which approximates the Gaussian:

$$G_t(\mathbf{x}, \mathbf{y}) = (4\pi t)^{-d/2} \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{4t}\right) (\phi(\mathbf{x}, \mathbf{y}) + O(1)), \quad (5.84)$$

where  $\phi$  is a smooth function with  $\phi(\mathbf{x}, \mathbf{x}) = 1$  and  $(4\pi t)^{d/2}$  is the normalization factor computed from

$$(4\pi t)^{d/2} = \int_M \exp\left(-\frac{\|\mathbf{z}\|^2}{4t}\right). \quad (5.85)$$

Hence, we have

$$G_t(\mathbf{x}, \mathbf{y}) \approx (4\pi t)^{-d/2} \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{4t}\right). \quad (5.86)$$

By the exponential formula (5.82),

$$\mathcal{L} = \lim_{t \rightarrow 0^+} \frac{I - \mathcal{A}_t}{t}. \quad (5.87)$$

When  $t$  is close to 0, (5.87) yields the following approximation:

$$\mathcal{L}(f)(\mathbf{x}) \approx \frac{1}{t} \left( f(\mathbf{x}) - (4\pi t)^{-d/2} \int_M \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{4t}\right) f(\mathbf{y}) \right). \quad (5.88)$$

### 5.5.3 Discrete form of Laplace-Beltrami Operator

To construct the LE DR kernel, we need to discretize the expression of the continuous Laplace-Beltrami operator in (5.88). A function  $f$  on  $\mathcal{X}$  can be represented as a vector  $\mathbf{f} = (f(\mathbf{x}_1), \dots, f(\mathbf{x}_n))^T$ . Intuitively, we might discretize (5.88) to

$$\mathcal{L}(f)(\mathbf{x}_i) \approx \frac{1}{t} \left( f(\mathbf{x}_i) - (4\pi t)^{-d/2} \sum_{\mathbf{x}_j \in O_i} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{4t}\right) f(\mathbf{x}_j) \right). \quad (5.89)$$

Note that the normalization factor  $(4\pi t)^{d/2}$  should make

$$(4\pi t)^{-d/2} \sum_{\mathbf{x}_j \in O_i} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{4t}\right) = 1. \quad (5.90)$$

Unfortunately, it is not true in the discrete case. To ensure that the above sum is properly normalized to have the sum 1, we have to replace  $(4\pi t)^{d/2}$  by

$$d_i = \sum_{\mathbf{x}_j \in O_i} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{4t}\right). \quad (5.91)$$

However, the replacement above destroys the self-adjoint property of  $\mathcal{L}$ . Hence, to break away from the dilemma, as discussed in Section 3.2, we first define the (non-normalized) discrete Laplacian  $\mathbf{L}$  by

$$\mathbf{L}(f)(\mathbf{x}_i) = \frac{1}{t} \left( d_i f(\mathbf{x}_i) - \sum_{\mathbf{x}_j \in O_i} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{4t}\right) f(\mathbf{x}_j) \right), \quad (5.92)$$

which can be shortly rewritten as

$$t\mathbf{L}(f)(\mathbf{x}_i) = d_i f(\mathbf{x}_i) - \sum_{j=1}^n w_{i,j} f(\mathbf{x}_j),$$

with

$$w_{i,j} = \begin{cases} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{4t}\right), & \text{if } \mathbf{x}_j \in O_i, \\ 0, & \text{otherwise.} \end{cases} \quad (5.93)$$

Let

$$\mathbf{D} = \text{diag}(d_1, \dots, d_n), \quad \mathbf{W} = (w_{i,j})_{i,j=1}^n. \quad (5.94)$$

Then the matrix form of the non-normalized discrete Laplacian  $\mathbf{L}$  is

$$\mathbf{L}(\mathbf{f}) = (\mathbf{D} - \mathbf{W})\mathbf{f}, \quad (5.95)$$

where  $t$  is removed from the equation since it does not impact the solution of the problem. In Section 3.2, we already proved that Laplacian  $\mathbf{L}$  is positive and self-adjoint. It is also known that the discrete Laplace-Beltrami operator  $\mathcal{L}$  can be derived from  $\mathbf{L}$  by the formula

$$\mathcal{L} = \mathbf{D}^{-1/2} \mathbf{L} \mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2} \mathbf{W} \mathbf{D}^{-1/2}. \quad (5.96)$$

#### 5.5.4 Minimization Model for DR Data Set

We return to the discussion on the minimization problem for the DR data set  $\mathcal{Y} = h(\mathcal{X})$ . Write  $\mathbf{Y} = (\mathbf{y}_1, \dots, \mathbf{y}_n) \subset \mathbb{R}^d$  and recall that the  $i$ th row of  $\mathbf{Y}$ , denoted by  $\mathbf{Y}_i^T$ , is the  $i$ th feature function on the set  $\mathcal{X}$ :

$$\mathbf{Y}_i = (h^i(\mathbf{x}_1), \dots, h^i(\mathbf{x}_n))^T. \quad (5.97)$$

As discussed before,  $\mathbf{Y}$  is required to be centered and have orthonormal rows:

$$\mathbf{Y}\mathbf{1} = \mathbf{0}, \quad \mathbf{Y}\mathbf{Y}^T = \mathbf{I}. \quad (5.98)$$

By (5.81), we set  $\mathbf{Y}$  to be the solution to the following discretized minimization problem:

$$\mathbf{Y} = \operatorname{argmin} \operatorname{tr}(\mathbf{Y}\mathcal{L}\mathbf{Y}^T), \quad (5.99)$$

subject to the constraints (5.98).

The problem (5.99) can be solved by the eigen decomposition of  $\mathcal{L}$ . We assume that the graph  $G = (\mathcal{X}, \mathbf{A})$  is connected. Hence, 0 is the simple eigenvalue of  $\mathcal{L}$  and  $\mathbf{D}^{1/2}\mathbf{1}$  is the 0-eigenvector. Let all eigenvalues of  $\mathcal{L}$  be arranged in the order of

$$0 = \lambda_0 < \lambda_1 \leq \lambda_2 \cdots \leq \lambda_{n-1}.$$

Then  $\mathbf{Y}_i$  is the eigenvector of  $\mathcal{L}$  corresponding to  $\lambda_i$ ,  $1 \leq i \leq d$ .

Recall that the eigenvalue problem

$$\mathcal{L}\mathbf{a} = \lambda\mathbf{a} \quad (5.100)$$

is equivalent to the generalized eigenvalue problem

$$\mathbf{L}\mathbf{b} = \lambda\mathbf{D}\mathbf{b} \quad (5.101)$$

with the relation  $\mathbf{a} = \mathbf{D}^{1/2}\mathbf{b}$ . Since  $\mathbf{Y}_i$  is the solution of (5.100) so that

$$\mathcal{L}\mathbf{Y}_i = \lambda_i\mathbf{Y}_i, \quad 1 \leq i \leq d, \quad (5.102)$$

we conclude that  $\mathbf{B}_i = \mathbf{D}^{-1/2}\mathbf{Y}_i$  is the solution of (5.101):

$$\mathbf{L}\mathbf{B}_i = \lambda_i\mathbf{D}\mathbf{B}_i. \quad (5.103)$$

Hence, alternatively, we can first solve (5.101) to obtain

$$\mathbf{B} = (\mathbf{B}_1, \dots, \mathbf{B}_d)^T, \quad (5.104)$$

and then obtain the DR data matrix by

$$\mathbf{Y} = \mathbf{B}\mathbf{D}^{1/2}.$$

**备注99.** If we ignore the data dilation caused by  $\mathbf{D}^{1/2}$ , we can identify  $\mathbf{Y}$  with  $\mathbf{B}$ . Hence, in LE DR algorithm, we sometimes set  $\mathbf{B}$  as the output. A reason to use Equation (5.101) to replace (5.100) is that the computation of the kernel  $\mathbf{L}$  saves the normalization step, that may accelerate the computing speed. Therefore,  $\mathbf{L}$  is often called LE kernel or graph Laplacian.

### 5.5.5 Construction of General LE Kernels

As shown in the previous subsection, the DR data set  $\mathbf{B}$  is the solution of the minimization problem

$$\mathbf{B} = \operatorname{argmin} \operatorname{tr}(\mathbf{B}(\mathbf{D} - \mathbf{W})\mathbf{B}^T), \quad s.t. \quad \mathbf{B}\mathbf{D}\mathbf{B}^T = \mathbf{I}, \mathbf{B}\mathbf{D}\mathbf{1} = \mathbf{0}, \quad (5.105)$$

where

$$\operatorname{tr}(\mathbf{B}(\mathbf{D} - \mathbf{W})\mathbf{B}^T) = \frac{1}{2} \sum_{i,j=1}^n w_{i,j} \|\mathbf{B}_i - \mathbf{B}_j\|^2, \quad (5.106)$$

and  $w_{i,j}$  is defined by (5.93). The objective function above with the choice of weights  $w_{i,j}$  incurs a heavy penalty if neighboring points  $\mathbf{x}_i$  and  $\mathbf{x}_j$  are far apart from each other. Therefore, minimizing it is an attempt to ensure that if  $\mathbf{x}_i$  and  $\mathbf{x}_j$  are close, then  $\mathbf{B}_i$  and  $\mathbf{B}_j$  are close too. By the interpretation above, the weight matrix  $\mathbf{W}$  in the minimization problem (5.105) is not necessarily derived from the Gaussian. Generally, let  $g > 0$  be a decreasing function on  $[0, \infty)$ . Then the weight matrix defined by

$$w_{i,j} = \begin{cases} g(\|\mathbf{x}_i - \mathbf{x}_j\|^2), & \text{if } \mathbf{x}_j \in O_i, \\ 0, & \text{otherwise.} \end{cases} \quad (5.107)$$

can be used in the model (5.99). When a general function  $g$  is applied in the construction of  $\mathbf{W}$ , we shall call (5.105) a generalized LE model. This generalization widens the contents of LE method. As an important extension, we may directly adopt the adjacency matrix  $\mathbf{A}$  of the data graph as the weight matrix  $\mathbf{W}$  in (5.105).

### 5.5.6 Laplacian Eigenmaps Algorithm

LE algorithm consists of the following steps (Comment: Steps 1 and 2 should be merged as computing neighborhoods requires the similarity/distance/weight matrix.).

1. **Neighborhood definition.** This step is similar to other nonlinear DR algorithms. Either  $k$ -neighborhood or  $\varepsilon$ -neighborhood can be used for the construction of graph on the data  $\mathcal{X}$ . In applications,  $k$ -neighborhood is more common than  $\varepsilon$ -neighborhood. To ensure that the graph is undirected, when  $k$ -neighborhood is applied, a symmetric modification is needed for obtaining a symmetric adjacency matrix  $\mathbf{A}$ .

2. **Weighted graph creation.** The LE algorithm can simply adopt the symmetric adjacency  $\mathbf{A}$  as the weight matrix. It can also use the formula (5.93), even the more general formula (5.107), to construct  $\mathbf{W}$ . Using  $\mathbf{A}$  as the weight matrix has the advantage that the constructed weight matrix has no parameter. Otherwise, say, (5.93) is adopted, we have a parameter  $t$  in the determination of the weights. Experiments show that the LE DR algorithm is not very sensitive to  $t$  in general. Hence, it is quite common using parameter-free weight matrix in LE algorithm.
3. **DR kernel construction.** Let the weight matrix be denoted by  $\mathbf{W} = (w_{i,j})$ . Let  $d_i = \sum_j w_{i,j}$  and  $\mathbf{D} = \text{diag}(d_1, \dots, d_n)$ . Then  $\mathbf{L} = \mathbf{D} - \mathbf{W}$  is the LE DR kernel.
4. **Eigen decomposition of DR kernel.** We solve the generalized eigen problem:

$$\mathbf{L}\mathbf{f} = \lambda\mathbf{D}\mathbf{f}. \quad (5.108)$$

Let  $\{\lambda_i\}_{i=0}^{n-1}$ , with  $0 = \lambda_0 \leq \lambda_1 \leq \dots \leq \lambda_{n-1}$ , be the eigenvalues of (5.108), and  $\mathbf{F} = (\mathbf{f}_1, \dots, \mathbf{f}_d)$  be the harmonic eigenvector matrix corresponding to the eigenvalue set  $\{\lambda_1, \dots, \lambda_d\}$ . Then  $\mathbf{Y} = \mathbf{F}^T \mathbf{D}^{1/2}$  is the required DR data matrix. If scaling is not required, we set  $\mathbf{Y} = \mathbf{F}^T$ .

### 5.5.7 Experiments on Artificial Surfaces

In this section, we demonstrate the LE method in the dimensionality reduction for artificial surfaces. For comparing LE method with other nonlinear DR methods, we adopt the same settings on the data for testing the algorithm. In Fig. 5.26, the 3-dimensional data of 4 artificial surfaces, 3D-cluster, Punched sphere, Swiss roll, and S-curve are reduced to 2-dimensional ones by LE algorithm. In the experiment, 1000 points are randomly sampled on each surface and 10 nearest points are used to construct the neighborhood of a point. The experiment shows that LE algorithm works well for S-curve and Punched sphere. It cannot produce a valid DR for 3D-cluster and the DR data is not well scaled for Swiss roll. Note that LE is based on Laplace-Beltrami operator. Hence, it may not perform well in non-smooth area of a surface. But the data of 3D-cluster shows a lack of the local geometry in the area around the line segments. It is reasonable that LE does not produce a valid DR data set for 3D-cluster. The coordinate mapping in LE is not an isometric mapping. Hence, it usually does not well preserve the local Euclidean distances for the data such as Swiss roll.

Noise on data impacts the dimensionality reduction results. In Fig. 5.27 and Fig. 5.28, we apply LE algorithm to reduce the dimension of noisy data, with the noise standard deviation 0.2 and 0.4 respectively. The results show that LE algorithm can tolerate

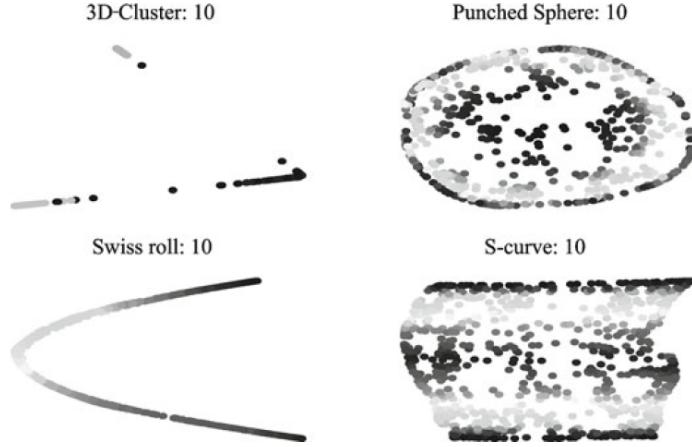


图 5.26: LE algorithm is tested on 4 surfaces: 3D-cluster, Punched sphere, Swiss roll, and S-curve. 1000 points are randomly sampled on each of them. All are reduced to 2-dimensional data (on the plane). The original 3-dimensional graphs of these surfaces are not presented here. The figure displays the DR results for these surfaces. The graph shows that LE algorithm works well for S-curve and Punched sphere.

noise with the deviation in a certain range, producing better DR results than LTSA and LLE. This is perhaps due to the diffusion structure of the LE DR kernel. Recall that the LE DR kernel is derived from a diffusion kernel and a diffusion processing reduces the noise impact. When strong noise exists, the DR data may not preserve the data geometry very well since the data have a large deviation from the underlying manifold.

As we discussed before, the properties of the underlying manifold greatly impact the results of the DR. In Fig. 5.29, the different heights of Punched sphere are chosen for the test. It is not surprise that when the height increases, the DR results have more distortion. Compared with LTSA and LLE, LE has better performance. In Fig. 5.30 LE algorithm is applied to Swiss rolls with different lengths 21, 51, and 81, respectively. The results show that when the length is 81, the DR result is not valid. This phenomenon can be explained as follows. When the length of the roll is 81, the surface area is four times of the roll with the length 21. Since the sample points keep the same number of 1000, the distances between the points in a neighborhood for the long roll is also four times of those for the short roll in average. Therefore, when the length is too large, the graph neighborhood does not truly reside on a neighborhood of the manifold, the DR result cannot preserve the geometric structure of the manifold. In Fig. 5.31, the different lengths of S-curve, 5, 10, 30, are chosen for the test with the same data settings as in Fig. 5.26. By the same argument we have made for the Swiss rolls in Fig. 5.30, the distortion

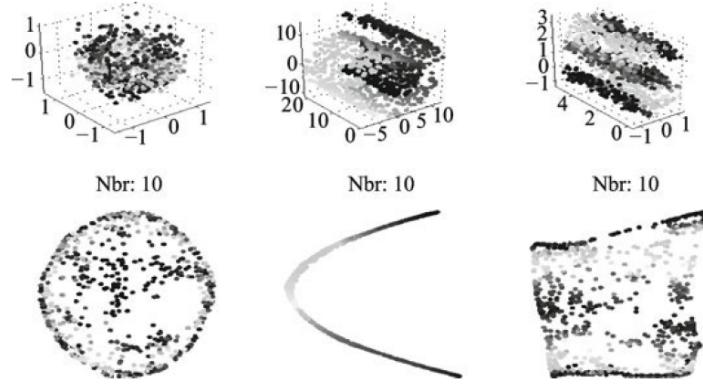


图 5.27: LE algorithm is applied on three surfaces with the noise standard deviation 0.2. All of the surfaces are sampled in the same way as in Fig. 5.26. The figure shows that LE algorithm can tolerate noise in a certain range. At the top line, three noisy data are presented. Top left: Punched sphere, Top middle: Swiss roll. Top right: S-curve. At the bottom line are their corresponding dimensional reduced 2-dimensional data.

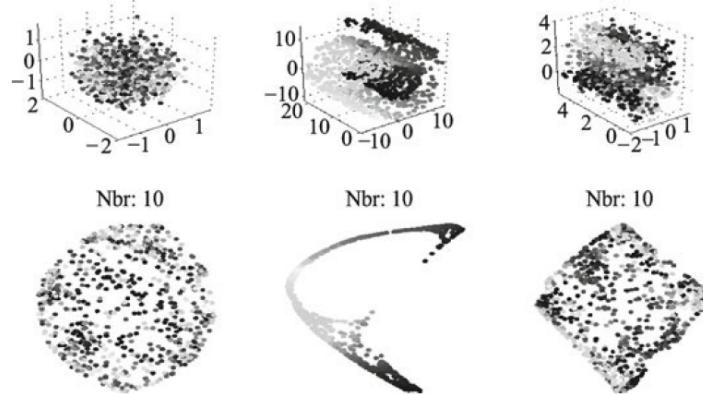


图 5.28: LE algorithm is applied on three surfaces with the noise standard deviation 0.4. All of the surfaces are sampled in the same way as in Fig. 5.26. The figure shows that when the noise deviation is large, the DR results of LE algorithm have distortion at a certain level. At the top line, three noisy data are presented. Top left: Punched sphere, Top middle: Swiss roll. Top right: S-curve. At the bottom line are their corresponding dimensional reduced 2-dimensional data.

of the DR result for the long S-curve is due to the neighborhood inconsistence.

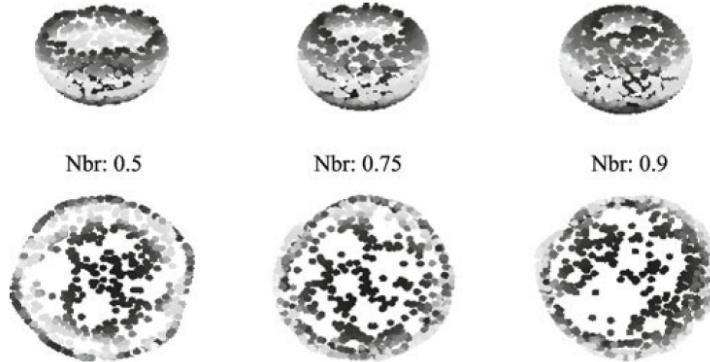


图 5.29: Effect of surface shape on LE (I). Embeddings of the punched spheres are computed for different choices of the heights of the punched sphere, 0.5, 0.75, and 0.9, respectively, where the diameter of the original sphere is unit. It shows that the dimensionality reduction for the punched sphere becomes worse as the height increases.

### 5.5.8 Conclusions

Similar to LLE and LTSA, the LE kernel is sparse. The construction of the LE kernel is extremely simple if it adopts the adjacency matrix as the weight matrix. If the formula (5.93) or (5.107) is applied in the construction of the weight matrix, the computation is still straightforward and economic. Therefore, the algorithm performs very fast. However, LE method does not guarantee an isometric coordinate mapping  $h$  in DR processing. Hence, the reduced data may not preserve the local Euclidean distance. In some applications where local distance reservation is critical, the method may not provide the required DR. When a parametric weight matrix is used, the optimization of the parameter becomes a question although the algorithm is not very sensitive to it in general. Since the LE kernel is derived from a diffusion kernel, it relies on the smoothness of underlying manifolds. When the data resides on a non-smooth manifold, such as 3D-cluster, the method becomes invalid. Taking the advantage that the diffusion processing reduces noise, the LE algorithm tolerates a certain level of noise and may become unstable only if the noise is strong.

**练习100.** Use one person's face images in Data for MATLAB hackers (<http://www.cs.nyu.edu/~roweis/data.html>) as the data for dimensionality reduction with LE, where the target dimension is 2, and see whether it is better than CMDS, ISOMAP, LLE,

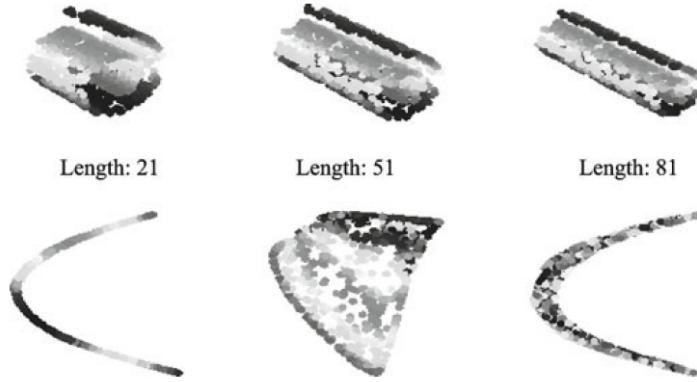


图 5.30: The figure illustrates that when the graph neighborhood is inconsistent with manifold, the DR data may distort the geometric structure of data so that it becomes invalid. Embeddings of the Swiss rolls are computed for different choices of the lengths of the roll, 21, 51, 81, respectively. The surface data are sampled in the same way as in Fig. 5.26. In the figure, the results of the experiment show that the dimensionality reduction for the Swiss roll becomes worse as the length increases. This phenomenon is due to the neighborhood inconsistence.

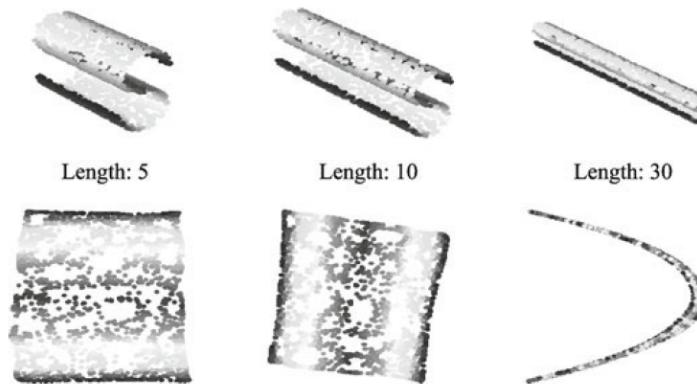


图 5.31: The figure illustrates again that when the graph neighborhood is inconsistent with manifold, the DR data may distort the geometric structure of data so that it becomes invalid. In this test, embeddings of the S-curves are computed for different choices of the lengths, 5, 10, and 30, respectively. The surface data are sampled in the same way as in Fig. 5.26. The results show that the dimensionality reduction becomes worse as the length increases, due to the neighborhood inconsistence.

and LTSA, e.g., whether the face images are arranged better in some order. Different neighborhood sizes and distance measures are encouraged to be tried.

## 5.6 Diffusion Maps

In Laplacian eigenmaps method, the DR data is obtained from the eigen-subspace of the Laplace-Beltrami operator on the underlying manifold where the observed data resides. In Section 5.5, it was pointed out that Laplace-Beltrami operator directly links up with the heat diffusion operator by the exponential formula for positive self-adjoint operators. Therefore, they have the same eigenvector set, and the corresponding eigenvalues are linked by the exponential relation too. The relation indicates that the diffusion kernel on the manifold itself can be used in the construction of DR kernel. The diffusion map method (Dmaps) constructs the DR kernel using the diffusion maps. Then the DR data is computed from several leading eigenvectors of the Dmaps DR kernel.

### 5.6.1 Description of DR Method of Diffusion Maps

The Gaussian-type diffusion kernels were widely used in machine learning and data clusters before 2004. The mathematics of diffusion maps was first studied by Coifman and Lafon. The idea of the Dmaps method is to consider a family of diffusion maps, each of which embeds the data set into a Euclidean space so that the Euclidean distance in the space is equal to the diffusion distance on the data. The various diffusion distances induced by diffusion maps describe the multiscale geometric structures on the data, represented by feature functions. Among them, one truly reveals the targeted features and the corresponding embedding produces the required dimensional reduction of the original data. In this section, we adopt the same data model as before, assuming that the data set  $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subset \mathbb{R}^D$  resides on a  $d$ -dimensional manifold  $M \subset \mathbb{R}^D$ , which has a parameterization  $g$ . Let  $h \approx g^{-1} = (h^1, \dots, h^d)^T$  be the feature mapping on  $M$ . The DR data set is obtained as the feature representation of  $\mathcal{X} : \mathcal{Y} = h(\mathcal{X})$ .

#### 5.6.1.1 Diffusion Operator on Manifold

The DR method of diffusion maps has the similar idea as the method of Laplacian eigenmaps. Hence, we first review the relation between Laplace-Beltrami operator  $\mathcal{L}$  and the Neumann heat diffusion operator  $\mathcal{A}_t$ . Recall that the Laplace-Beltrami operator  $\mathcal{L}$  and the Neumann heat operator  $\mathcal{A}_t$  on  $M$  are linked by the exponential formula

$$\mathcal{A}_t = \exp^{-t\mathcal{L}}. \quad (5.109)$$

Since  $\mathcal{A}_t$  has the integral representation

$$\mathcal{A}_t(f)(\mathbf{x}) = \int_M G_t(\mathbf{x}, \mathbf{y}) f(\mathbf{y}), \quad f \in C(M), \quad (5.110)$$

it can be extended onto a larger function space, say, it can be considered as an operator on  $L^2(M)$ . Let the inner product on  $L^2(M)$  be denoted by  $\langle \cdot, \cdot \rangle$ . The spectral decomposition of  $\mathcal{L}$  leads to

$$\mathcal{L}(f) = \sum_{i=0}^{\infty} \lambda \phi^i \langle \phi^i, f \rangle, \quad (5.111)$$

where

$$0 = \lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_n \rightarrow \infty$$

are eigenvalues of  $\mathcal{L}$  and  $\phi_i$ ,  $0 \leq i < \infty$ , are corresponding eigenfunctions. Unfortunately, the operator  $\mathcal{L}$  is unbounded on  $L^2(M)$ . By (5.111) and (5.109), the spectral decomposition of  $\mathcal{A}_t$  is

$$\mathcal{A}_t(f) = \sum_{i=0}^{\infty} e^{-t\lambda} \phi^i \langle \phi^i, f \rangle, \quad (5.112)$$

which shows that the Neumann heat diffusion operator  $\mathcal{A}_t$  has the same eigenfunction set as the Laplace-Beltrami operator  $\mathcal{L}$ , but  $\mathcal{A}_t$  is a bounded operator on  $L^2(M)$  with the operator norm 1. Besides, all of its eigenvalues are nonnegative so that it is also a positive self-adjoint operator. These important properties make diffusion operators widely used in many applications. Dmaps method uses the eigen decomposition of  $\mathcal{A}_t$  to find the DR data set. In practice, we use the following maximization model. For a certain  $t$ ,

$$h^{i+1} = \operatorname{argmax} \langle f, \mathcal{A}_t(f) \rangle, \quad s.t. \quad \langle h, h^k \rangle = 0, k = 0, \dots, i, \quad (5.113)$$

where  $h^0 = e$  is the constant function.

### 5.6.1.2 Normalization of Diffusion Kernels

The construction of Dmaps kernel is very similar to the construction of LE kernel. Dmaps kernel is derived from the Neumann heat kernel, which can be approximated by Gaussian. Hence, a Dmaps kernel can be constructed by normalizing the Gaussian. In Section 5.5, we already learned that the normalization of Gaussian in the continuous data model is independent of variables  $\mathbf{x}$ , but depends on  $\mathbf{x}$  in the discrete data model. In the normalization of discrete Gaussian we are caught in a dilemma: if we want to keep the sum of each row in the kernel being 1 so that 1 is the 1-eigenvector of the normalized kernel, then we lose the self-adjoint property of the kernel.

In diffusion maps, we construct symmetric Dmaps kernel, replacing the row-sum-one rule by a weaker rule: 1 is the greatest eigenvalue of the kernel. We construct two different

types of Dmaps kernels: Graph-Laplacian type and Laplace-Beltrami type. Let  $G_t$  be a non-normalized Gaussian, say,

$$G_t(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2/t),$$

which generates a bi-linear functional  $\langle g, \mathcal{A}_t(f) \rangle$  on  $L^2(M) \times L^2(M)$  having the form

$$\langle g, \mathcal{A}_t(f) \rangle = \int_M \int_M G_t(\mathbf{x}, \mathbf{y}) f(\mathbf{x}) g(\mathbf{y}).$$

Write

$$S(\mathbf{x}) = \int_M G_t(\mathbf{x}, \mathbf{y}) d\mathbf{y}$$

and normalize  $G_t(\mathbf{x}, \mathbf{y})$  by

$$\tilde{G}_t(\mathbf{x}, \mathbf{y}) = G_t(\mathbf{x}, \mathbf{y}) / S(\mathbf{x}).$$

Then  $\tilde{G}_t(\mathbf{x}, \mathbf{y})$  has row-sum-one property:

$$\int_M \tilde{G}_t(\mathbf{x}, \mathbf{y}) d\mathbf{y} = 1.$$

But  $\tilde{G}_t(\mathbf{x}, \mathbf{y})$  is not symmetric. The kernel  $\tilde{G}_t$  generates the operator

$$\tilde{\mathcal{A}}_t(f) = \int_M \tilde{G}_t(\mathbf{x}, \mathbf{y}) f(\mathbf{y}) d\mathbf{y}.$$

To construct a symmetric kernel, we set

$$\hat{G}_t(\mathbf{x}, \mathbf{y}) = \frac{G_t(\mathbf{x}, \mathbf{y})}{\sqrt{S(\mathbf{x})S(\mathbf{y})}},$$

which generates the operator

$$\hat{\mathcal{A}}_t(f) = \int_M \hat{G}_t(\mathbf{x}, \mathbf{y}) f(\mathbf{y}) d\mathbf{y}.$$

We can see that the operators  $\tilde{\mathcal{A}}_t$  and  $\hat{\mathcal{A}}_t$  have the same eigenvalue set. Furthermore,  $f$  is an eigenvector of  $\tilde{\mathcal{A}}_t$  achieving the eigenvalue  $\lambda$  if and only if  $f/\sqrt{S}$  is the eigenvector of  $\hat{\mathcal{A}}_t$  achieving the same eigenvalue. This conclusion has a simple proof as follows. If we have

$$\tilde{\mathcal{A}}_t(f) = \lambda f,$$

then

$$\begin{aligned} \hat{\mathcal{A}}_t(f/\sqrt{S})(\mathbf{x}) &= \frac{1}{\sqrt{S(\mathbf{x})}} \int_M G_t(\mathbf{x}, \mathbf{y}) \frac{f(\mathbf{y})}{S(\mathbf{y})} d\mathbf{y} \\ &= \frac{1}{\sqrt{S(\mathbf{x})}} \tilde{\mathcal{A}}_t(f)(\mathbf{x}) \\ &= \lambda \left( f/\sqrt{S} \right) (\mathbf{x}), \end{aligned} \tag{5.114}$$

which indicates that  $f/\sqrt{S}$  is the eigenfunction of  $\hat{\mathcal{A}}_t$  achieving  $\lambda$ . The proof of the reverse is similar. Since this normalization is similar to that for graph Laplacian, we call  $\hat{G}_t$  the diffusion kernel of Graph-Laplacian type. Note that  $S$  is the eigenfunction of  $\hat{\mathcal{A}}_t$  achieving 1.

Another type of normalization of  $G_t$  employs the symmetric kernel

$$G_t^{sym}(\mathbf{x}, \mathbf{y}) = \frac{G_t(\mathbf{x}, \mathbf{y})}{S(\mathbf{x})S(\mathbf{y})}.$$

Let

$$S^{sym}(\mathbf{x}) = \int_M G_t^{sym}(\mathbf{x}, \mathbf{y}) d\mathbf{y},$$

which normalizes  $G_t^{sym}$  to the Dmaps kernel

$$\bar{G}_t = \frac{G_t^{sym}(\mathbf{x}, \mathbf{y})}{\sqrt{S^{sym}(\mathbf{x})S^{sym}(\mathbf{y})}}, \quad (5.115)$$

corresponding to the operator

$$\bar{\mathcal{A}}_t(f) = \int_M \bar{G}_t(\mathbf{x}, \mathbf{y}) f(\mathbf{y}) d\mathbf{y}.$$

Because the infinitesimal of  $\bar{\mathcal{A}}_t$  is  $\mathcal{L}$ :

$$\lim_{t \rightarrow 0} \frac{\bar{\mathcal{A}}_t - I}{t} = \mathcal{L},$$

we call  $\bar{\mathcal{A}}_t$  the diffusion kernel of Laplace-Beltrami type. The function  $\sqrt{S^{sym}}$  is the eigenfunction of  $\bar{\mathcal{A}}_t$  achieving 1. The discretization of the kernels  $\hat{G}_t$  can be trivially derived too. We skip it here.

## 5.6.2 Diffusion Maps Algorithm

### 5.6.2.1 Dmaps DR Algorithm Description

Let given data  $\mathcal{X} \subset \mathbb{R}^D$  be the input of a Dmaps algorithm. Assume that the dimension of the DR data is  $d$ . Then the output of the algorithm is the DR data  $\mathcal{Y} \subset \mathbb{R}^d$  in the matrix form. A Dmaps algorithm consists of the following steps.

1. **Data graph construction.** This step is same as other nonlinear DR methods. Either  $k$ -neighborhood or  $\varepsilon$ -neighborhood can be used for defining data graph.
2. **Weight matrix creation.** Assume that a date graph  $(\mathcal{X}, \mathbf{A})$  is defined on the data set  $\mathcal{X}$ , which determines a neighborhood system on  $\mathcal{X}$ . Let  $O_i$  denote the

neighborhood of  $\mathbf{x}_i$  and  $N(i)$  denote the corresponding subscript set, which is read from the  $i$ -th row of  $\mathbf{A}$ . We create the weight matrix  $\mathbf{W} = (w_{ij})_{ij=1}^n$  by setting

$$w_{ij} = \begin{cases} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{t}\right), & j \in N(i) \cup \{i\}, \\ 0, & \text{otherwise,} \end{cases} \quad (5.116)$$

where  $t > 0$  is a parameter. To create a non-parameter weight matrix, we choose

$$t = \frac{1}{\sum_{i=1}^n |N(i)|} \sum_{i=1}^n \sum_{j \in N(i)} \|\mathbf{x}_i - \mathbf{x}_j\|^2.$$

( $\mathbf{W}$  corresponds to  $G_t(\mathbf{x}, \mathbf{y})$  in section 5.6.1.2.)

3. **Diffusion kernel construction.** We normalize the weight matrix to construct the Dmaps kernel  $\mathbf{K}$ . To construct a Graph-Laplacian type kernel, we set  $v_i = (\mathbf{W}^i)^T \mathbf{1}$ , where  $(\mathbf{W}^i)^T$  is the  $i$ -th row of  $\mathbf{W}$ , and define

$$k_{ij} = \frac{w_{ij}}{\sqrt{v_i v_j}}.$$

( $v_i$  and  $k_{ij}$  corresponds to  $S(\mathbf{x})$  and  $\hat{G}_t(\mathbf{x}, \mathbf{y})$  in section 5.6.1.2, respectively.)

To construct a Laplace-Beltrami type kernel, we first compute

$$\tilde{k}_{ij} = \frac{w_{ij}}{v_i v_j}.$$

( $\tilde{k}_{ij}$  corresponds to  $G_t^{sym}(\mathbf{x}, \mathbf{y})$  in section 5.6.1.2.)

Then set  $u_i = (\tilde{\mathbf{K}}^i)^T \mathbf{1}$  and define

$$k_{ij} = \frac{\tilde{k}_{ij}}{\sqrt{u_i u_j}}.$$

( $u_i$  and  $k_{ij}$  corresponds to  $S^{sym}(\mathbf{x})$  and  $\bar{G}_t(\mathbf{x}, \mathbf{y})$  in section 5.6.1.2, respectively.)

4. **DR kernel decomposition.** Let  $\mathbf{z}_0, \mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_d$  be the eigenvectors of  $\mathbf{K}$  achieving the  $(d+1)$  largest eigenvalues  $1 = \lambda_0 > \lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d > 0$ . Scale  $\mathbf{z}_i$  to

$$\tilde{\mathbf{z}}_i = (z_{i1}/z_{01}, z_{i2}/z_{02}, \dots, z_{in}/z_{0n})^T, \quad 1 \leq i \leq d.$$

Then the DR data matrix  $\mathbf{Y} = (\tilde{\mathbf{z}}_1, \tilde{\mathbf{z}}_2, \dots, \tilde{\mathbf{z}}_d)^T$ .

**备注101.** The neighborhood definition for Dmaps can be also realized by thresholding the weights appropriately. If the weight threshold method is used for the definition of neighborhood, Step 1 and Step 2 are merged to the following single step. Let  $\tau$  be a weight threshold and  $g_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{t}\right)$ . Then the weight matrix  $\mathbf{W} = (w_{ij})_{ij=1}^n$  is constructed by

$$w_{ij} = \begin{cases} g_{ij}, & \text{if } g_{ij} \leq \tau, \\ 0, & \text{otherwise.} \end{cases} \quad (5.117)$$

### 5.6.3 The Original Description on Dmaps (Taken from [72])

#### 5.6.3.1 Diffusion Distances

Our goal is to define a distance metric on an arbitrary set that reflects the connectivity of the points within the set. Suppose that one is dealing with a data set in the form of a graph. When identifying clusters or groups of points in this graph, one needs to measure the amount of interaction, as described by the graph structure, between pairs of points. Following this idea, two points should be considered to be close if they are connected by many short paths in the graph. As a consequence, points within regions of high density (defined as groups of nodes with a high degree in the graph), will have a high connectivity. The connectivity is furthermore decided by the strengths of the weights in the graph. Below, we review the diffusion framework and put it into the context of eigenmaps, dimensionality reduction, and Markov random walk learning on graphs.

Let  $G = (V, W)$  be a finite graph with  $n$  nodes, where the weight matrix  $W = \{w(x, y)\}_{x, y \in V}$  satisfies the following conditions:

- symmetry:  $W = W^T$  and
- pointwise positivity:  $w(x, y) \geq 0$  for all  $x, y \in V$ .

The way we define the weights should be completely application-driven, the only requirement being that  $w(x, y)$  should represent the degree of similarity or affinity (as defined by the application) of  $x$  and  $y$ . In particular, we expect  $w(x, x)$  to be a positive number. For instance, if we are dealing with data points on a manifold, we can start with a Gaussian kernel  $w_\sigma = \exp(-\|x - y\|^2/\sigma)$  and then normalize it in order to adjust the influence of geometry versus the distribution of points on the manifold.

The graph  $G$  with weights  $W$  represents our knowledge of the local geometry of the set. Next, we define a Markov random walk on this graph. To this end, we introduce the degree  $d(x)$  of node  $x$  as

$$d(x) = \sum_{z \in V} w(x, z).$$

If one defines  $P$  to be the  $n \times n$  matrix whose entries are given by

$$p_1(x, y) = \frac{w(x, y)}{d(x)},$$

then  $p_1(x, y)$  can be interpreted as the probability of transition from  $x$  to  $y$  in one time step. By construction, this quantity reflects the first-order neighborhood structure of the graph. A new idea introduced in the diffusion maps framework is to capture information on larger neighborhoods by taking powers of the matrix  $P$  or, equivalently, to run the random walk forward in time. If  $P^t$  is the  $t$ -th iterate of  $P$ , then the entry  $p_t(x, y)$  represents the probability of going from  $x$  to  $y$  in  $t$  time steps. Increasing  $t$ , corresponds to propagating the local influence of each node with its neighbors. In other words, the quantity  $P^t$  reflects the intrinsic geometry of the data set defined via the connectivity of the graph in a diffusion process and the time  $t$  of the diffusion plays the role of a scale parameter in the analysis.

If the graph is connected, we have that:

$$\lim_{t \rightarrow +\infty} p_t(x, y) = \phi_0(y),$$

where  $\phi_0$  is the unique stationary distribution

$$\phi_0(x) = \frac{d(x)}{\sum_{z \in V} d(z)}. \quad (5.118)$$

This quantity is proportional to the degree of  $x$  in the graph, which is one measure of the density of points. The Markov chain is furthermore reversible, i.e., it verifies the following detailed balance condition

$$\phi_0(x)p_1(x, y) = \phi_0(y)p_1(y, x).$$

We are mainly concerned with the following idea: For a fixed but finite value  $t > 0$ , we want to define a metric between points of  $V$  which is such that two points  $x$  and  $z$  will be close if the corresponding conditional distributions  $p_t(x, \cdot)$  and  $p_t(z, \cdot)$  are close. The  $L_1$  norm  $\|p_t(x, \cdot) - p_t(z, \cdot)\|$  and the Kullback-Leibler divergence or any other distance can be used to measure the difference between  $p_t(x, \cdot)$  and  $p_t(z, \cdot)$ . However, as shown below, the  $L_2$  metric between the conditional distributions has the advantage that it allows one to relate distances to the spectral properties of the random walk – and thereby, as we will see in the next section, connect Markov random walk learning on graphs with data parameterization via eigenmaps. We will define the “diffusion distance”  $D_t$  between  $x$  and  $y$  as the weighted  $L_2$  distance

$$D_t^2(x, z) = \|p_t(x, \cdot) - p_t(z, \cdot)\|_{1/\phi_0}^2 = \sum_{y \in V} \frac{(p_t(x, y) - p_t(z, y))^2}{\phi_0(y)}, \quad (5.119)$$

where the “weights”  $\frac{1}{\phi_0(y)}$  penalize discrepancies on domains of low density more than those of high density.

This notion of proximity of points in the graph reflects the intrinsic geometry of the set in terms of connectivity of the data points in a diffusion process. The diffusion distance between two points will be small if they are connected by many paths in the graph. This metric is thus a key quantity in the design of inference algorithms that are based on the preponderance of evidences for a given hypothesis. For example, suppose one wants to infer class labels for data points based on a small number of labeled examples. Then, one can easily propagate the label information from a labeled example  $x$  to the new point  $y$  following 1) the shortest path or 2) all paths connecting  $x$  to  $y$ . The second solution (which is employed in the diffusion framework) is usually more appropriate, as it takes into account all “evidences” relating  $x$  to  $y$ . Furthermore, since diffusion-based distances add up the contribution from several paths, they are also (unlike the shortest path) robust to noise; the latter point will be illustrated via an example.

### 5.6.3.2 Dimensionality Reduction and Parameterization of Data by Diffusion Maps

As mentioned, an advantage of the above definition of the diffusion distance is the connection to the spectral theory of the random walk. As is well-known, the transition matrix  $P$  that we have constructed has a set of left and right eigenvectors and a set of eigenvalues  $|\lambda_0| \geq |\lambda_1| \geq \dots \geq |\lambda_{n-1}|$ :

$$\phi_j^T P = \lambda_j \phi_j^T, \text{ and} \quad (5.120)$$

$$P \psi_j = \lambda_j \psi_j, \quad (5.121)$$

where it can be verified that  $\lambda_0 = 1$ ,  $\psi_0 \equiv 1$ , and that

$$\phi_k^T \psi_l = 0, \quad \text{if } k \neq l. \quad (5.122)$$

(This is because  $\phi_k^T P \psi_l = \phi_k^T (P \psi_l) = \lambda_l \phi_k^T \psi_l$  and  $\phi_k^T P \psi_l = (\phi_k^T P) \psi_l = \lambda_k \phi_k^T \psi_l$ . So  $(\lambda_k - \lambda_l) \phi_k^T \psi_l = 0$ . When  $\lambda_k \neq \lambda_l$ ,  $\phi_k^T \psi_l = 0$ ). Note that the eigenvalues are all real because  $P = D^{-1}W$  is similar to  $D^{-1/2}WD^{-1/2}$ .

In fact, left and right eigenvectors are dual and can be regarded as signed measures and test functions, respectively. These two sets of vectors are related according to

$$\psi_l(x) = \frac{\phi_l(x)}{\phi_0(x)} \quad \text{for all } x \in V. \quad (5.123)$$

(From (5.118), without normalization we may write  $\phi_0 = D\mathbf{1}$ . Then

$$\text{diag}(1./\phi_0) = D^{-1}. \quad (5.124)$$

To prove (5.123), by (5.121) we only have to verify

$$D^{-1}W\text{diag}(1./\phi_0)\phi_l = \lambda_l\text{diag}(1./\phi_0)\phi_l. \quad (5.125)$$

By (5.124), the above is simply (5.120).)

For ease of deduction, we normalize the left eigenvectors of  $P$  with respect to  $1/\phi_0$ :

$$\|\phi_l\|_{1/\phi_0}^2 = \sum_x \frac{\phi_l^2(x)}{\phi_0(x)} = 1, \quad (5.126)$$

then the right eigenvectors are also normalized with respect to the weight  $\phi_0$ :

$$\|\psi_l\|_{\phi_0}^2 = \sum_x \psi_l^2(x)\phi_0(x) = 1,$$

thanks to (5.123).

If  $p_t(x, y)$  is the kernel of the  $t$ -th iterate  $P^t$ , we will then have the following biorthogonal spectral decomposition:

$$p_t(x, y) = \sum_{j \geq 0} \lambda_j^t \psi_j(x) \phi_j(y). \quad (5.127)$$

The above identity corresponds to a weighted principal component analysis of  $P^t$ . The first  $k$  terms provide the best rank- $k$  approximation of  $P^t$ , where “best” is defined according to the following weighted metric for matrices:

$$\|A\|^2 = \sum_x \sum_y \phi_0(x)[a(x, y)]^2 \frac{1}{\phi_0(y)}.$$

Here is our main point: If we insert (5.127) into (5.119), we will have that

$$D_t^2(x, z) = \sum_{j=1}^{n-1} \lambda_j^{2t} (\psi_j(x) - \psi_j(z))^2.$$

(

$$\begin{aligned}
 D_t^2(x, z) &= \sum_{y \in V} \frac{1}{\phi_0(y)} (p_t(x, y) - p_t(z, y))^2 \\
 &= \sum_{y \in V} \frac{1}{\phi_0(y)} \left[ \sum_{j \geq 0} \lambda_j^t (\psi_j(x) - \psi_j(z)) \phi_j(y) \right]^2 \\
 &= \sum_{y \in V} \left\{ \frac{1}{\phi_0(y)} \sum_{j \geq 0} \lambda_j^{2t} (\psi_j(x) - \psi_j(z))^2 \phi_j^2(y) \right. \\
 &\quad \left. + \sum_{j \neq k} (\lambda_j \lambda_k)^t (\psi_j(x) - \psi_j(z)) (\psi_k(x) - \psi_k(z)) \phi_j(y) \phi_k(y) \right\} \\
 &= \sum_{j \geq 0} \lambda_j^{2t} (\psi_j(x) - \psi_j(z))^2 \sum_{y \in V} \left( \frac{1}{\phi_0(y)} \phi_j^2(y) \right) \\
 &\quad + \sum_{j \neq k} (\lambda_j \lambda_k)^t (\psi_j(x) - \psi_j(z)) (\psi_k(x) - \psi_k(z)) \left( \sum_{y \in V} \frac{1}{\phi_0(y)} \phi_j(y) \phi_k(y) \right) \\
 &= \sum_{j \geq 0} \lambda_j^{2t} (\psi_j(x) - \psi_j(z))^2 \\
 &\quad + \sum_{j \neq k} (\lambda_j \lambda_k)^t (\psi_j(x) - \psi_j(z)) (\psi_k(x) - \psi_k(z)) \sum_{y \in V} \psi_j(y) \phi_k(y) \\
 &= \sum_{j \geq 0} \lambda_j^{2t} (\psi_j(x) - \psi_j(z))^2,
 \end{aligned} \tag{5.128}$$

where the fourth equality used (5.126), the fifth used (5.123), and the sixth used (5.122).)

Since  $\psi_0 \equiv 1$  is a constant vector, it does not enter in the sum above. Furthermore, because of the decay of the eigenvalues<sup>2</sup>, we only need a few terms in the sum for a certain accuracy. To be precise, let  $q(t)$  be the largest index  $j$  such that  $|\lambda_j|^t > \delta |\lambda_1|^t$ . The diffusion distance can then be approximated to relative precision  $\delta$  using the first  $q(t)$  nontrivial eigenvectors and eigenvalues according to

$$D_t^2(x, z) \approx \sum_{j=1}^{q(t)} \lambda_j^{2t} (\psi_j(x) - \psi_j(z))^2.$$

Now, observe that the identity above can be interpreted as the Euclidean distance in  $\mathbb{R}^{q(t)}$  if we use the right eigenvectors weighted with  $\lambda_j^t$  as coordinates on the data. In other words, this means that, if we introduce the diffusion map

$$\Psi_t : x \mapsto \begin{pmatrix} \lambda_1^t \psi_1(x) \\ \lambda_2^t \psi_2(x) \\ \vdots \\ \lambda_{q(t)}^t \psi_{q(t)}(x) \end{pmatrix}, \tag{5.129}$$

<sup>2</sup>The speed of the decay depends on the graph structure. For example, for the special case of a fully connected graph, the first eigenvalue will be 1 and the remaining eigenvalues will be equal to 0. The other extreme case is a graph that is totally disconnected with all eigenvalues equal to 1.

then clearly,

$$D_t^2(x, z) \approx \sum_{j=1}^{q(t)} \lambda_j^{2t} (\psi_j(x) - \psi_j(z))^2 = \|\Psi_t(x) - \Psi_t(z)\|^2. \quad (5.130)$$

Note that the factors  $\lambda_j^t$  in the definition of  $\Psi_t$  are crucial for this statement to hold.

The mapping  $\Psi_t : V \mapsto \mathbb{R}^{q(t)}$  provides a parameterization of the data set  $V$ , or equivalently, a realization of the graph  $G$  as a cloud of points in a lower-dimensional space  $\mathbb{R}^{q(t)}$ , where the rescaled eigenvectors are the coordinates. The dimensionality reduction and the weighting of the relevant eigenvectors are dictated by both the time  $t$  of the random walk and the spectral fall-off of the eigenvalues.

Equation (5.130) means that  $\Psi_t$  embeds the entire data set in  $\mathbb{R}^{q(t)}$  in such a way that the Euclidean distance is an approximation of the diffusion distance. Our approach is thus different from other eigenmap methods: Our starting point is an explicitly defined distance metric on the data set or graph. This distance is also the quantity we wish to preserve during a nonlinear dimensionality reduction.

#### 5.6.4 Implementation of Dmaps for DR

In this section, we demonstrate the Dmaps method in the dimensionality reduction for artificial surfaces. For comparing with other nonlinear DR methods, we adopt the same data settings as in LLE, LTSA, and LE.

##### 5.6.4.1 Implementation of Dmaps of Graph-Laplacian Type

We first test Dmaps method of Graph-Laplacian type on artificial surfaces. For convenience, we abbreviate Dmaps method of Graph-Laplacian to GL-Dmaps, Dmaps method of Laplace-Beltrami type to LB-Dmaps, and Self-tuning Dmaps to ST-Dmaps.

In Fig. 5.32, the 3-dimensional data of 4 artificial surfaces, 3D-cluster, Punched sphere, Swiss roll, and S-curve are reduced to 2-dimensional ones by GL-Dmaps algorithm. In the experiment, 1000 points are randomly sampled on each surface and 10 nearest points are used to construct the neighborhood of a point. The experiment shows that GL-Dmaps algorithm works well for S-curve, 3D-cluster, and Punched sphere. A certain shape distortion occurs on Swiss roll. Note that all Dmaps methods are based on the diffusion processing. Hence, it is not surprising that it can work well for nonsmooth surface such as 3D-cluster. Dmaps methods do not preserve the local Euclidean distances very well. Hence, the DR result of GL-Dmaps for Swiss roll does not keep the rectangular shape although there is no misclassification on the DR data set.

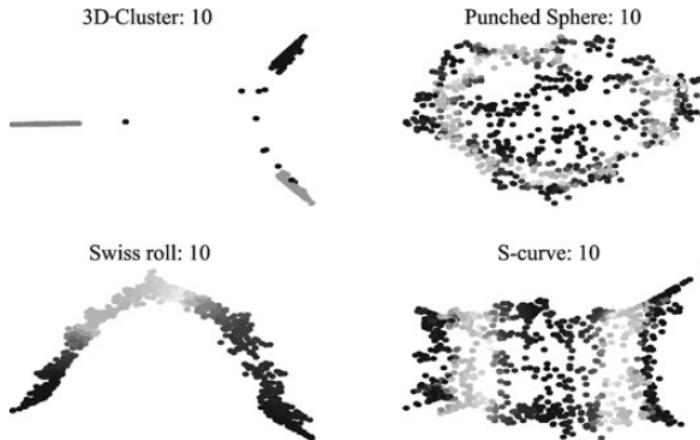


图 5.32: GL-Dmaps algorithm is tested on 4 surfaces: 3D-cluster, Punched sphere, Swiss roll, and S-curve. 1000 points are randomly sampled on each of them. All are reduced to 2-dimensional data (on the plane). The original 3-dimensional graphs of these surfaces are not presented here. The figure displays the DR results for these surfaces. The graph shows that GL-Dmaps algorithm works well for them.

In Fig. 5.33 and Fig. 5.34, we apply GL-Dmaps algorithm to reduce the dimension of noisy data, with the noise standard deviation 0.2 and 0.4 respectively. The results show that GL-Dmaps algorithm is insensitive to noise. This is due to the diffusion processing in Dmaps methods, which reduces the noise on the data.

In Fig. 5.35, the different heights of Punched sphere are chosen for the test. It shows the results similar to those obtained by other methods: when the height increases, the DR results have more distortions. Compared to others, GL-Dmaps method is relatively insensitive to the height change.

In Fig. 5.36 GL-Dmaps algorithm is applied to the lengths of S-curve, 5, 10, and 30, with the same data settings as in Fig. 5.32. It is shown that when the length of the S-curve surface is too long, the distortion of the DR set occurs mainly due to the neighborhood inconsistence. Dmaps algorithms are insensitive to the neighborhood sizes. The algorithms remain stable over a wide range of neighborhood sizes. Figure 5.37 displays the results of GL-Dmaps for the DR of S-curve data, showing that the algorithm is very stable. In the last experiment, the impact of the parameter  $t$  (see (5.116)) on the Dmaps DR kernel construction is tested. The results show that the algorithm is insensitive to the parameter, as shown in Fig. 5.38.

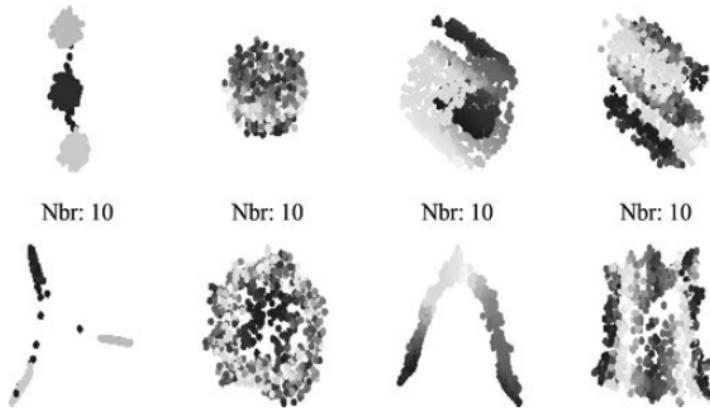


图 5.33: GL-Dmaps algorithm is applied on 4 surfaces, with the noise standard deviation 0.2. All of the surfaces are sampled in the same way as in Fig. 5.32. The figure shows that GL-Dmaps algorithm is insensitive to noise. At the top line, four noisy data are presented. Top left: 3D-cluster, Top second: Punched sphere, Top third, Swiss roll, Top right: S-curve. At the bottom line are their corresponding dimensional reduced 2-dimensional data.

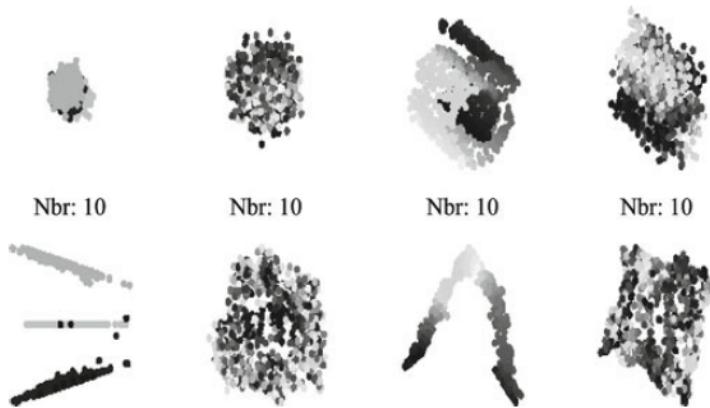


图 5.34: GL-Dmaps algorithm is applied on four surfaces as in Fig. 5.33, with the noise standard deviation 0.4. The surfaces are sampled in the same way as in Fig. 5.32. The figure shows that when the noise deviation is large, the DR results of GL-Dmaps algorithm have certain distortions. At the top line, four noisy data are presented. Top left: 3D-cluster, Top second: Punched sphere, Top third: Swiss roll, Top right: S-curve. At the bottom line are their corresponding dimensional reduced 2-dimensional data.

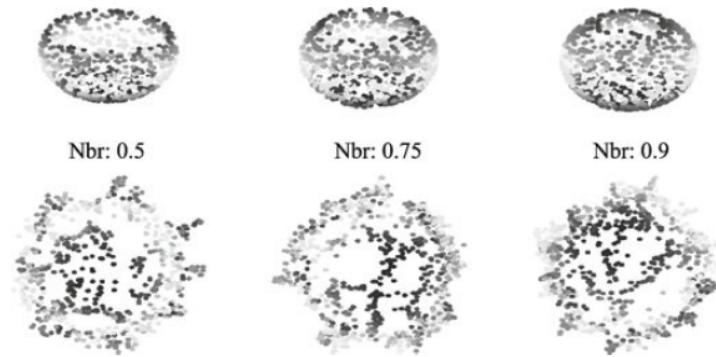


图 5.35: Effect of surface shape on GL-Dmaps. The dimensionality reduction of the punched spheres is produced by GL-Dmaps for heights of the punched sphere, 0.5, 0.75, and 0.9, respectively, where the diameter of the original sphere is unit. It shows that when the height increases, more color mixture areas occur in the dimensionality reduction data.

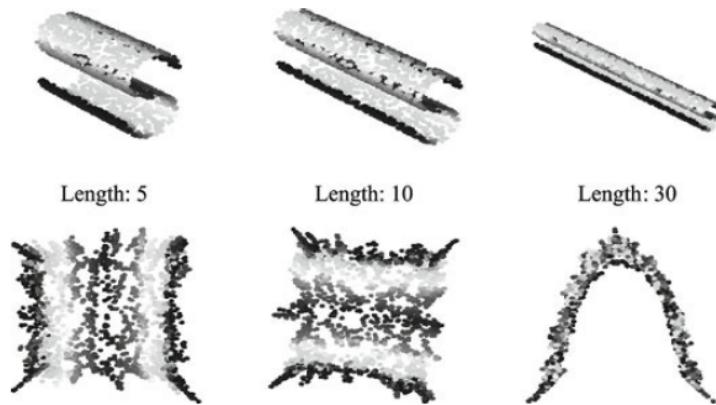


图 5.36: The figure illustrates again that when the graph neighborhood is inconsistent with manifold, the DR data may distort the geometric structure of dat so that it becomes invalid. In this test, embeddings of the S-curves are computed by GL-Dmaps algorithm for different choices of the lengths, 5, 10, and 30, respectively. The surface data are sampled in the same way as in Fig. 5.32. The results show that the dimensionality reduction data are invalid when the length of the S-curve surface is 30. The distortion is due to the neighborhood inconsistence.

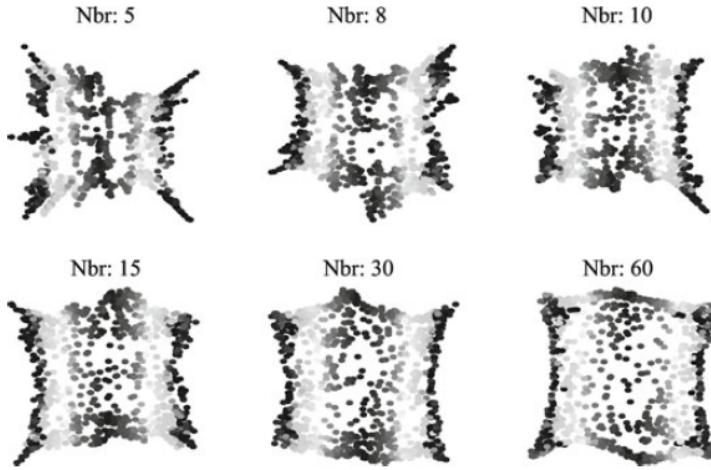


图 5.37: Effect of neighborhood size on GL-Dmaps. The results show that GL-Dmaps algorithm is insensitive to the neighborhood size. In this figure, GL-Dmaps algorithm is applied to S-curve for different choices of the number of nearest neighbors  $k$ , 5, 8, 10, 15, 30, 60, respectively. A reliable GL-Dmaps embedding from 3dimension to 2dimension can be obtained over a wide range of values.

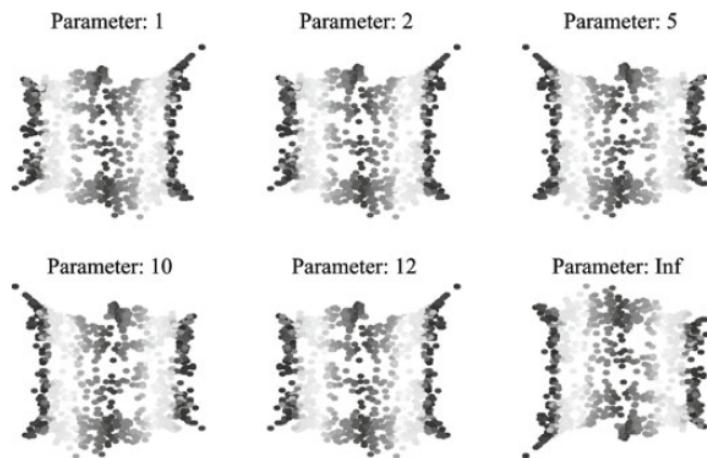


图 5.38: Impact of the parameter  $t$  on GL-Dmaps algorithm. The results show that GL-Dmaps algorithm is insensitive to the value of the parameter  $t$  that is used in the kernel construction. In this figure, GL-Dmaps algorithm is applied to S-curve for different choices of the parameter in Gaussian, say 1, 2, 5, 10, 12, Inf, respectively. A reliable GL-Dmaps embedding from 3-dimension to 2-dimension can be obtained over a wide range of  $t$  values.

### 5.6.4.2 Implementation of Dmaps of Laplace-Beltrami Type

There are no significant differences between LB-Dmaps and GL-Dmaps. In [71], the author pointed out that the GL-Dmaps embedding is sensitive to the density of the points. Particularly, when the density has a steep peak, this embedding tends to map all points around this peak to a single point, creating a corner. Compared with GL-Dmaps, LB-Dmaps is less sensitive to the point density (see Figure 2.5 in [71]).

In Fig. 5.39, the 3-dimensional data of 4 artificial surfaces, 3D-cluster, Punched sphere, Swiss roll, and S-curve, are reduced to 2-dimensional ones by LB-Dmaps algorithm. In the experiment, 1000 points are randomly sampled on each surface and 10 nearest points are used to construct the neighborhood of a point. The experiment shows that LB-Dmaps algorithm produces the results similar to GL-Dmaps, but less sensitive to the density of the points. For example, the DR results for Punched sphere and S-curve of LB-Dmaps contain less shape distortion than GL-Dmaps.

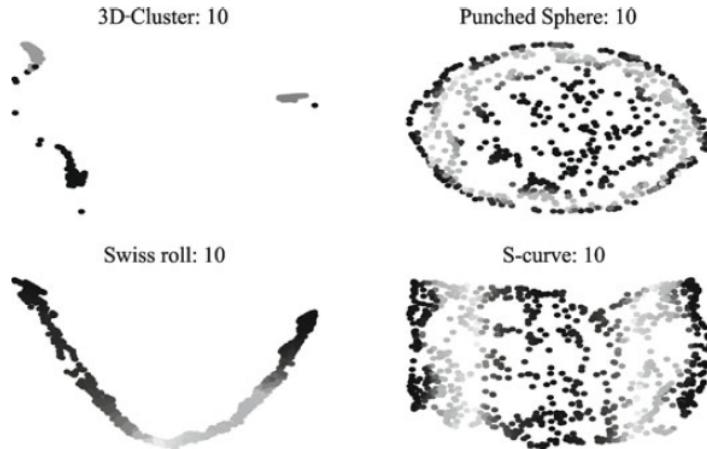


图 5.39: LB-Dmaps algorithm is tested on 4 surfaces: 3D-cluster, Punched sphere, Swiss roll, and S-curve. 1000 points are randomly sampled on each of them. All are reduced to 2-dimensional data (on the plane). The figure displays the DR results for these surfaces. The graphs show that LB-Dmaps algorithm produces less peak points than GL-Dmaps, particularly for the DR data for Punched sphere and Scurve.

In Fig. 5.40, the different heights of Punched sphere are chosen for the test of LB-Dmaps embedding. It shows the results similar to those obtained by GL-Dmaps: LB-Dmaps method is relatively insensitive to the height change. Again we can see that the DR data of LB-Dmaps contain less irregular points than GL-Dmaps. In the last two experiments, the impact of the neighbor size and the parameter  $t$  on the LB-Dmaps embedding are displayed. The results show again that the algorithm is insensitive to

either the neighbor size or the parameter  $t$ , and the DR data set has less irregular points, as shown in Fig. 5.41 and Fig. 5.42, respectively.

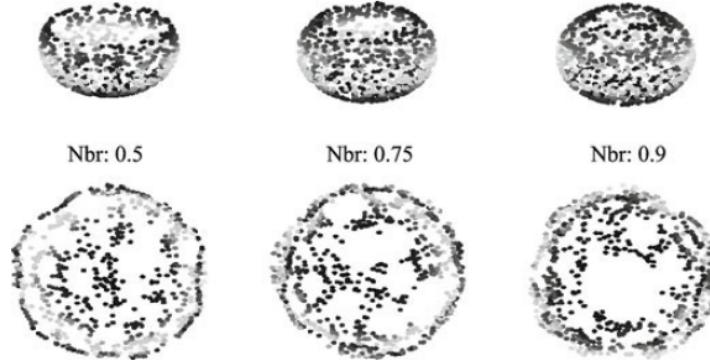


图 5.40: Effect of surface shape on LB-Dmaps. The dimensionality reduction of the punched spheres is produced by LB-Dmaps for heights of the punched sphere, 0.5, 0.75, and 0.9, respectively, where the diameter of the original sphere is unit. It shows that the DR data of LB-Dmaps contain less irregular points than GL-Dmaps.

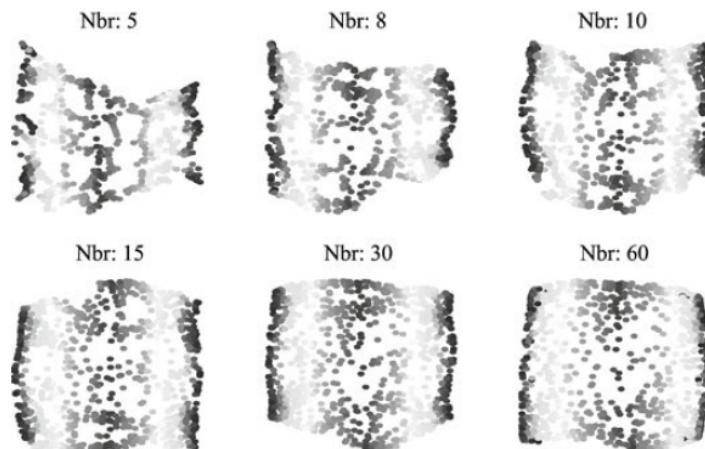


图 5.41: Impact of neighborhood size on LB-Dmaps. The results show that LB-Dmaps algorithm is insensitive to the neighborhood size. It is also shown that the DR data produced by LB-Dmaps have less irregular points than GL-Dmaps.

#### 5.6.4.3 Implementation of Dmaps of Self-turning Type

There are no significant differences between ST-Dmaps and GL-Dmaps. In Fig. 5.43, the 3-dimensional data of 4 artificial surfaces, 3D-cluster, Punched sphere, Swiss roll, and S-curve, are reduced to 2-dimensional ones by ST-Dmaps algorithm. In the experiment,

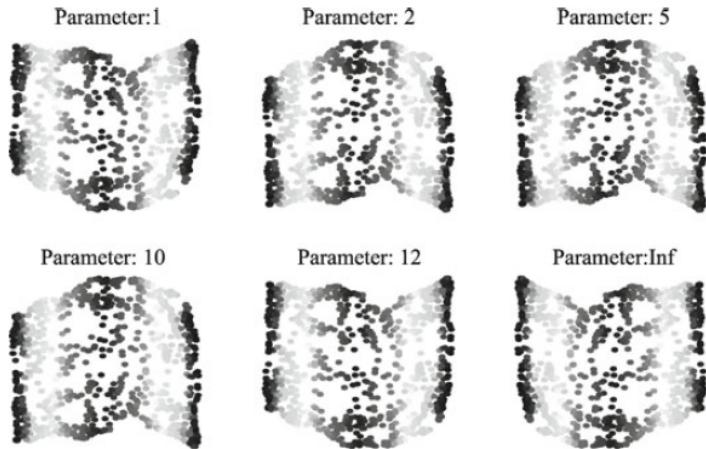


图 5.42: Impact of the parameter  $t$  on LB-Dmaps algorithm. The results show that BL-Dmaps algorithm is insensitive to the value of the parameter  $t$ . It is also shown that the DR data have less irregular points than GL-Dmaps.

1000 points are randomly sampled on each surface and the tuning number  $m$  is set at 10. The experiment shows that ST-Dmaps algorithm produces the results similar to those of GL-Dmaps. If we apply the same settings to other tests used above, the DR results of ST-Dmaps are also very similar to those obtained by GL-Dmaps embedding. Hence, we skip the figures for those tests.

### 5.6.5 Numerical Examples

#### 5.6.5.1 Importance of Learning the Nonlinear Geometry of Data in Clustering

In many applications, real data sets exhibit highly nonlinear structures. In such cases, linear methods such as Principal Components will not be very efficient for representing the data. With the diffusion coordinates, however, it is possible to learn the intrinsic geometry of data set and then project the data points into a nonlinear coordinate space with a diffusion metric. In this diffusion space, one can use classical geometric algorithms (such as separating hyperplane- based methods, k-means algorithms, etc.) for unsupervised as well as supervised learning.

To illustrate this idea, we study the famous Swiss roll. This data set is intrinsically a surface embedded in three dimensions. In this original coordinate system, global extrinsic distances, such as the Euclidean distance, are often meaningless as they do not incorporate any information on the structure or shape of the data set. For instance, if we run the k-means algorithm for clustering with  $k = 4$ , the obtained clusters do not reflect the

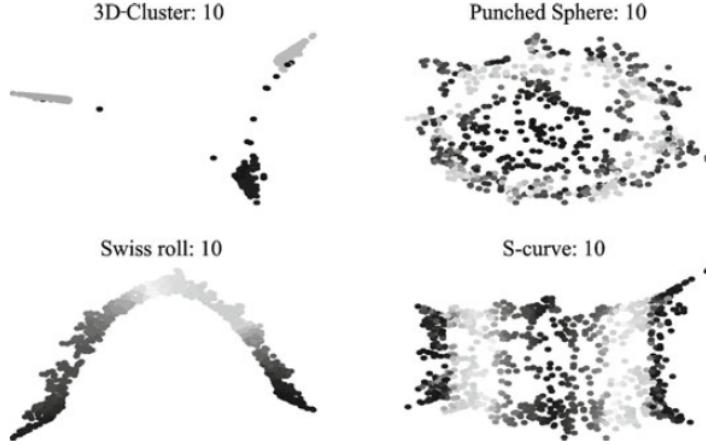


图 5.43: ST-Dmaps algorithm is tested on 4 surfaces: 3D-cluster, Punched sphere, Swiss roll, and S-curve. 1000 points are randomly sampled on each of them. All are reduced to 2-dimensional data (on the plane). The figure displays the DR results for these surfaces. The graphs show that ST-Dmaps algorithm produces results very similar to GL-Dmaps.

natural geometry of the set. As shown in Fig. 5.44, there is some “leakage” between different parts of the spiral due to the convexity of the k-means clusters in the ambient space. As a comparison, we also show in Fig. 5.44 the result of running the k-means algorithm in diffusion space. In the latter case, we obtain meaningful clusters that respect the intrinsic geometry of the data set.

#### 5.6.5.2 Robustness of the Diffusion Distance

One of the most attractive features of the diffusion distance is its robustness to noise and small perturbations of the data. In short, its stability follows from the fact that it reflects the connectivity of the points in the graph. We illustrate this idea by studying the case of data points approximately lying on a spiral in the two-dimensional plane. The goal of this experiment is to show that the diffusion distance is a robust metric on the data, and in order to do so, we compare it to the shortest path (or geodesic) distance that is employed in schemes such as ISOMAP.

We generate 1,000 instances of a noisy spiral in the plane, each corresponding to a different realization of the random noise perturbation (see Fig. 5.45). From each instance, we construct a graph by connecting all pairs of points at a distance less than a given threshold  $\tau$ , which is kept constant over the different realizations of the spiral. The corresponding adjacency matrix  $W$  contains only zeros or ones, depending on the absence or presence of an edge, respectively. In order to measure the robustness of the diffusion

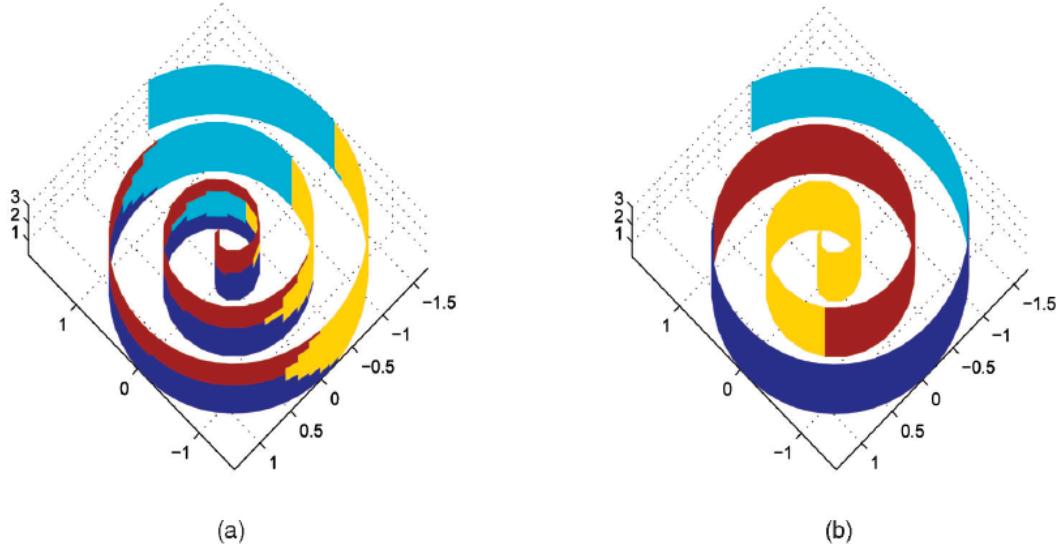


图 5.44: The Swiss roll, and its quantization by k-means ( $k = 4$ ) in the (a) original coordinate system and in the (b) diffusion space.

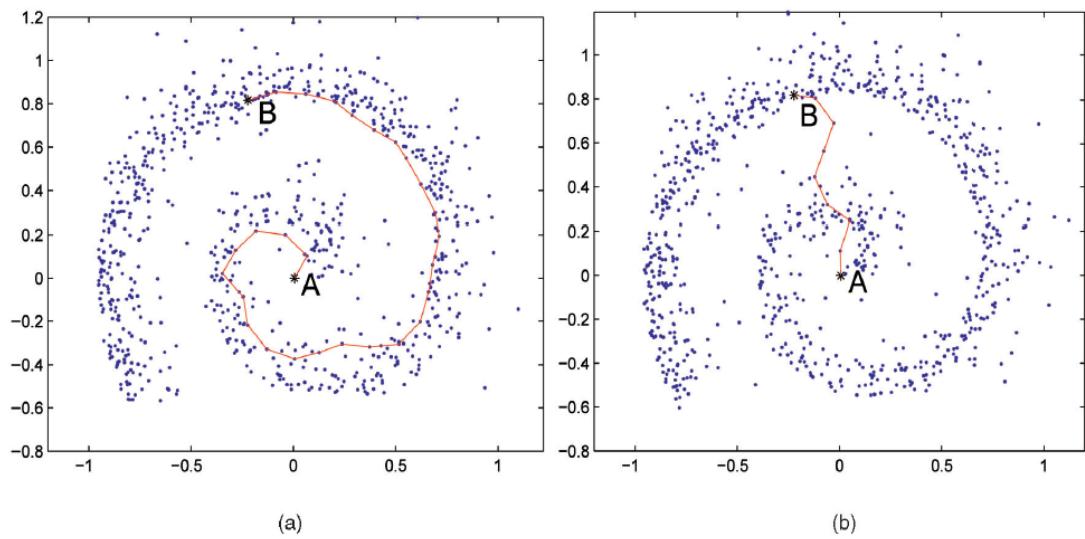


图 5.45: Two realizations of a noisy spiral with points of references  $A$  and  $B$ . Ideally, the shortest path between  $A$  and  $B$  should follow the (a) branch of the spiral. However, some realizations of the noise may give rise to shortcuts, thereby dramatically reducing the length of the (b) shortest path.

distance, we repeatedly compute the diffusion distance between two points of reference  $A$  and  $B$  in all 1,000 noisy spirals. We also compute the geodesic distance between these two points using Dijkstra's algorithm. As shown in Fig. 5.45, depending on the presence of shortcuts arising from points appearing between the branches of the spiral, the geodesic distance (or shortest path length) between  $A$  and  $B$  may vary by large amounts from one realization of the noise to another. The histogram of all geodesic distances measurements between  $A$  and  $B$  over the 1,000 trials is shown on Fig. 5.46. The distribution of the geodesic distance appears poorly localized, as its standard deviation equals 42 percent of its mean. This indicates that the geodesic distance is extremely sensitive to noise and thus unreliable as a measure of distance. The diffusion distance, however, is not sensitive to small random perturbations of the data set because, unlike the geodesic distance, it represents an average quantity. More specifically, it takes into account all paths of length less than or equal to  $t$  that connect  $A$  and  $B$ . As a consequence, shortcuts due to noise will have little weight in the computation, as the number of such paths is much smaller than the number of paths following the shape of the spiral. This is also what our experiment confirms: Fig. 5.46 shows the distribution of the diffusion distances between  $A$  and  $B$  over the random trials. In this experiment,  $t$  was taken to be equal to 600. The corresponding histogram shows a very localized distribution, with a standard deviation equal to only 7 percent of its mean, which translates into robustness and consistency of the diffusion distance.

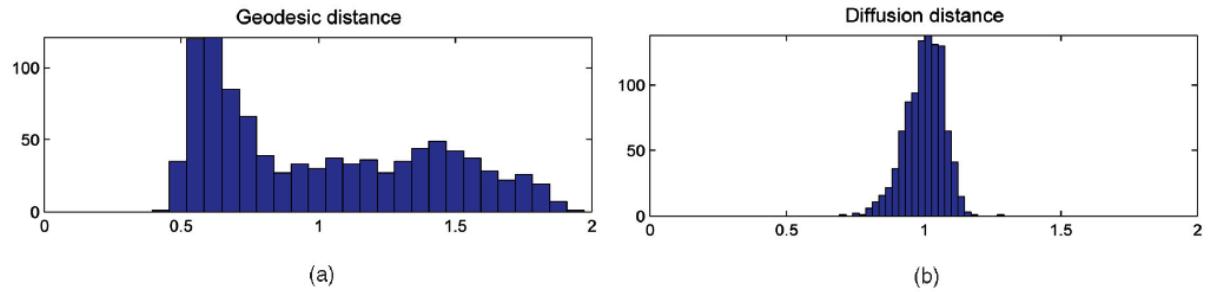


图 5.46: Distribution of the (a) geodesic and (b) diffusion distances. Each distribution was rescaled in order to have a mean equal to 1.

### 5.6.6 Conclusion

The construction of Dmaps kernels is very simple. The neighborhood construction for the diffusion maps method can be obtained from either  $k$ -neighborhood or  $\varepsilon$ -

neighborhood, and even can be obtained by thresholding the weights appropriately. In addition, by taking advantage of the semigroup structure, the fast, stable diffusion wavelet algorithms can be used to accelerate the computation of eigen decomposition of the DR kernel. Since Dmaps algorithms adopt diffusion processing, they are less sensitive to noise. The experiments show that they are also insensitive to the parameter  $t$  in the weight matrix. Dmaps methods do not exactly preserve the local distances. Hence, the shape of the DR data set sometimes are not preserved well.

**练习102.** Use one person's face images in *Data for MATLAB hackers* (<http://www.cs.nyu.edu/~roweis/data.html>) as the data for dimensionality reduction with Graph-Laplacian type Dmaps, where the target dimension is 2, and see whether it is better than CMDS, ISOMAP, LLE, LTSA, and LE, e.g., whether the face images are arranged better in some order. Different neighborhood sizes and distance measures are encouraged to be tried.

## 5.7 Maximum Variance Unfolding (MVU)

Unlike Isomap method that preserves geodesic distances, MVU method learns the data from the similarities, preserving both local distances and angles between the pairs of all neighbors of each point in the data set. Since the method keeps the local maximum variance in dimensionality reduction processing, it is called maximum variance unfolding (MVU). Like multidimensional scaling (MDS), MVU can be applied to the cases that only the local similarities of objects in a set are given. In these cases, MVU tries to find a configuration that preserves the given similarities. Technically, MVU adopts semidefinite programming (SDP) to solve the DR problems. Hence, it is also called semidefinite embedding (SDE). Solving a DR problem using MVU is expensive regarding both memory cost and time cost. To overcome the shortage of high computational cost, landmark MVU (LMVU) is introduced.

### 5.7.1 MVU Method Description

The method of maximum variance unfolding (MVU), introduced by Weinberger and Saul, is also called semidefinite embedding (SDE). It models dimensionality reduction problems using semidefinite programming (SDP). Isomap embedding preserves geodesic distances, which are close to Euclidean distances when the points are in a neighborhood. In Section 4.3.4, we already discussed the relation between Euclidean metric and centering Gram matrix. The Euclidean metric on a data set can be uniquely determined by the data Gram matrix up to shift and rotation. An entry of a Gram matrix is the inner

product of a pair vectors. MVU method keeps the maximum variance in dimensionality reduction processing by preserving the local similarities represented by the local Gram matrix. The intuitive idea of the method is illustrated in Fig. 5.47 and Fig. 5.48.

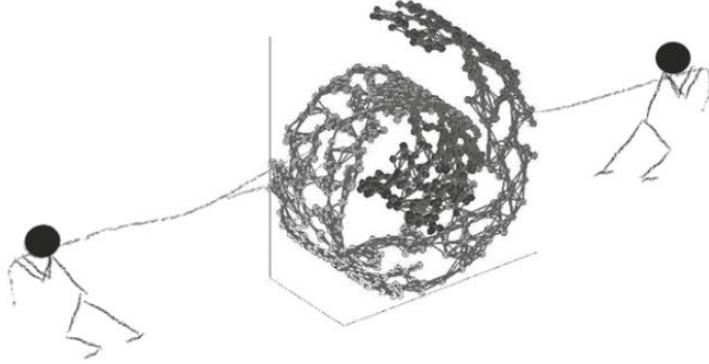


图 5.47: A discrete manifold is revealed by forming the graph that pairwise connects each data point and its 6 nearest neighbors.

### 5.7.1.1 Description of the MVU Method

As usual, we assume that the data set  $\mathcal{X} \subset \mathbb{R}^D$  lies on a convex  $d$ -manifold  $M$ . Let  $h$  be a coordinate mapping on  $M$  such that for each  $\mathbf{p} \in M$ ,  $\mathbf{q} = h(\mathbf{p}) \in \mathbb{R}^d$  is the coordinate representation of  $\mathbf{p}$ . Then  $g = h^{-1} : U \subset \mathbb{R}^d \rightarrow M$  is a parameterization of  $M$ . Without loss of generality, we may assume that  $dh \triangleq dh_{\mathbf{p}}$  is isometric so that  $dh \in \mathcal{O}_{d,D}$  with  $(dh)^T dh = \mathbf{I}$  at each point  $\mathbf{p} \in M$ . We denote the image of  $\mathcal{X}$  under the mapping  $h$  by  $\mathcal{Y} = h(\mathcal{X})$ . Then  $\mathcal{Y}$  is the DR data set we want to find. Recall that  $\mathcal{Y}$  is assumed to have zero mean:

$$\sum_{j=1}^n y_{i,j} = 0, \quad i = 1, \dots, d. \quad (5.131)$$

We denote the Gram matrix of  $\mathcal{Y}$  by  $\mathbf{K} = \mathbf{Y}^T \mathbf{Y}$ . Once  $\mathbf{K}$  is created,  $\mathbf{Y}$  can be obtained from the spectral decomposition of  $\mathbf{K}$ . To derive the kernel  $\mathbf{K}$ , MVU uses the following model.

Let  $P_i$  be a neighborhood of the point  $\mathbf{x}_i \in \mathcal{X}$  and  $Q_i$  be its image  $Q_i = h(P_i) \subset \mathbb{R}^d$ . Since the neighborhood  $P_i$  is very close to the tangent hyperplane of  $M$  at  $\mathbf{x}_i$ , the local geometry of  $P_i$  resembles that of  $Q_i$ . More precisely, we have

$$\|\mathbf{x}_j - \mathbf{x}_k\| \approx \|\mathbf{y}_j - \mathbf{y}_k\|, \quad j, k \in N(i), \quad (5.132)$$

where  $N(i)$  is the index set of the points in  $P_i$ . The MVU method formulates  $\mathcal{Y}$  as the

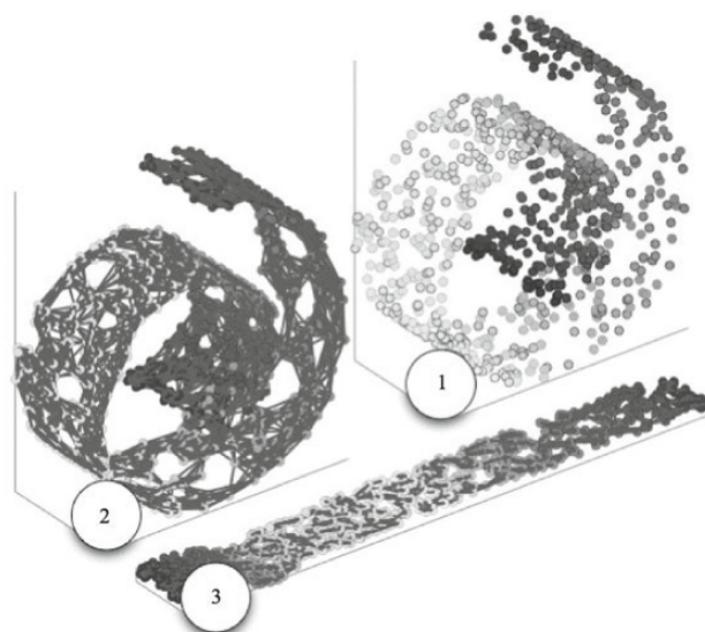


图 5.48: In this figure, (1) 1000 points are sampled from a Swiss roll. (2) The Swiss roll is linearly reconstructed from 12 nearest neighbors for each point. (3) It is an embedding from MVU, with 4 nearest neighbors for each point. All of the distances between 5 points (4 neighbors and the point itself) are used in the constraints of MVU.

solution of the following minimization problem:

$$\mathcal{Y} = \underset{\mathcal{Y} \subset \mathbb{R}^d}{\operatorname{argmin}} \sum_{i=1}^n \sum_{j \sim k} (\|\mathbf{x}_j - \mathbf{x}_k\|^2 - \|\mathbf{y}_j - \mathbf{y}_k\|^2)^2, \quad (5.133)$$

where  $k \sim j$  means that  $\mathbf{x}_k$  and  $\mathbf{x}_j$  are in the same neighborhood, say, in  $P_i$ . We also count  $\mathbf{x}_i$  in the relation “ $\sim$ ” so that  $k \sim i$  and  $j \sim i$ . The minimization problem (5.133) is not convex. Hence, it is hard to be solved directly. MVU then converts it to a semidefinite programming as follows.

Assume that  $\mathbf{K}$  is the Gram matrix of  $\mathbf{Y}$ . By the basic relation between centering Gram matrix and Euclidean metric presented in Section 4.3.4, we have

$$\|\mathbf{y}_j - \mathbf{y}_k\|^2 = \mathbf{K}_{k,k} + \mathbf{K}_{j,j} - 2\mathbf{K}_{k,j},$$

which yields the constraints

$$\mathbf{K}_{k,k} + \mathbf{K}_{j,j} - 2\mathbf{K}_{k,j} = \mathbf{S}_{k,j}, \quad j, k \in N(i), j > k, 1 \leq i \leq n, \quad (5.134)$$

where  $\mathbf{S}_{k,j} = \|\mathbf{y}_j - \mathbf{y}_k\|^2 \approx \|\mathbf{x}_j - \mathbf{x}_k\|^2$ . Because  $\mathcal{Y}$  is centered, Equation (5.131) yields the following constraint on  $\mathbf{K}$ :

$$\sum_{i,j=1}^n \mathbf{K}_{i,j} = 0. \quad (5.135)$$

Finally, the data  $\mathcal{Y}$  is required to have the maximum variance:

$$\mathcal{Y} = \underset{\mathcal{Y} \subset \mathbb{R}^d}{\operatorname{argmax}} \sum_{i,j=1}^n \|\mathbf{y}_j - \mathbf{y}_k\|^2 = \underset{\mathcal{Y} \subset \mathbb{R}^d}{\operatorname{argmax}} \|\mathbf{D}_Y\|_F^2, \quad (5.136)$$

where  $\mathbf{D}_Y$  is the Euclidean metric on  $\mathbf{Y}$ . By maximizing (5.136), the points of  $\mathcal{Y}$  are pulled as far apart as possible, subject to the constraints (5.134) and (5.135). By Lemma 70 in Section 4.3.4.1, recalling that  $\mathbf{K}$  is the centering Gram matrix of  $\mathcal{Y}$ , we have

$$\frac{1}{2n} \|\mathbf{D}_Y\|_F^2 = \operatorname{tr}(\mathbf{G}_Y^c) = \operatorname{tr}(\mathbf{K}).$$

Therefore, the maximization problem (5.136) is finally converted to

$$\mathbf{K} = \underset{\mathbf{K} \in \mathcal{S}_n^+}{\operatorname{argmax}} \operatorname{tr}(\mathbf{K}), \quad (5.137)$$

with the constraints (5.134) and (5.135). Combining them together, we obtain the maximization problem for the kernel  $\mathbf{K}$ :

$$\begin{aligned} & \text{maximize } \operatorname{tr}(\mathbf{K}), \\ & \text{s.t. } \sum_{i,j=1}^n \mathbf{K}_{i,j} = 0, \\ & \quad \mathbf{K}_{k,k} + \mathbf{K}_{j,j} - 2\mathbf{K}_{k,j} = \mathbf{S}_{k,j}, j, k \in N(i), j > k, 1 \leq i \leq n, \\ & \quad \mathbf{K} \succcurlyeq \mathbf{0}, \\ & \quad \mathbf{K} \in \mathcal{S}_n, \end{aligned} \quad (5.138)$$

where  $\mathcal{S}_n$  is the set of all symmetric matrices and  $\mathbf{K} \succcurlyeq \mathbf{0}$  means that  $\mathbf{K}$  is positive semi-definite.

Since the maximization problem for  $\mathbf{K}$  in (5.138) only involves the square distance matrix  $\mathbf{S}$ , MVU algorithm also accepts the local distance matrix  $\mathbf{D}$  as the input. Hence, MVU can be applied to the applications where only the similarity relations of objects are available. In this sense, MVU is similar to MDS.

The data model in MVU method defines a special graphs on the data sets. The constraints (5.134) indicate that the local distances and angles between all points in the neighborhood of  $\mathbf{x}$  are preserved. Therefore, the points in the same neighborhood have to be mutually connected in the data graph. We shall call this type of graph a locally complete graph. In general, for a  $k$ -neighborhood or an  $\varepsilon$ -neighborhood, nodes are not mutually connected. A locally complete graph can be created by refining a usual graph  $G = (\mathcal{X}, \mathbf{E})$ . Let  $G_r = (\mathcal{X}, \mathbf{E}_r)$  denote the locally complete graph refined from  $G$ . Then  $(k, j) \in \mathbf{E}_r$  if and only if  $(k, j) \in \mathbf{E}$  or  $(k, i), (j, i) \in \mathbf{E}$  for some  $i$ . We illustrate the graph refinement in Fig. 5.49, where the left is a 2-degree regular graph, and the right is its refined locally complete graph.

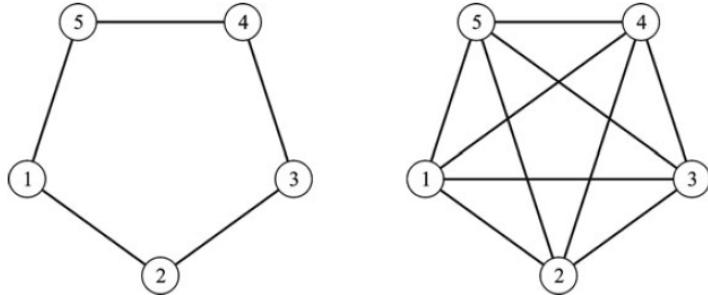


图 5.49: Refined graph for MVU.

### 5.7.1.2 MVU Algorithm

Assume that an observed data set  $\mathcal{X} \subset \mathbb{R}^D$  or a local distance matrix  $\mathbf{D}$  is given. MVU algorithm consists of 5 steps.

- 1. Neighborhood definition.** This step is necessary only if the input is the data set  $\mathcal{X}$ . This step is the same as the first step in the Isomap method. Either  $k$ -neighborhood or  $\varepsilon$ -neighborhood can be used to define the neighborhood system on the data set  $\mathcal{X}$ . In MVU,  $k$ -neighborhood is commonly used. We denote the corresponding graph by  $G = (\mathcal{X}, \mathbf{A})$ . As described in the previous subsection, the

adjacency matrix  $\mathbf{A}$  in MVU is only used to determine the indices of the constraints, i.e., a constraint of (5.134) will be made only if  $\mathbf{A}_{j,k} = 1$ .

2. **Compute Gram matrix of observed data set.** We may compute either the centering Gram matrix  $\mathbf{G}^c$  or the square-distance matrix  $\mathbf{S}$  to build the constraints of the semidefinite programming problem (5.138), depending the type of the input data. If the input is the data set  $\mathcal{X}$ , then the centering Gram matrix  $\mathbf{G}$  can be made. On the other hand, if the (local) square-distance matrix  $\mathbf{S} = (s_{jk})$  (or the local distance matrix  $\mathbf{D} = (d_{jk})$ ) is given, then the adjacency matrix  $\mathbf{A} = (a_{jk})$  is built by setting  $a_{jk} = \text{sign}(s_{jk})$ .
3. **Generate constraints for SDP.** When  $\mathbf{A}_{k,j} = 1$ , set the constraint

$$\mathbf{K}_{k,k} + \mathbf{K}_{j,j} - 2\mathbf{K}_{j,k} = \mathbf{G}_{k,k} + \mathbf{G}_{j,j} - 2\mathbf{G}_{j,k},$$

or

$$\mathbf{K}_{k,k} + \mathbf{K}_{j,j} - 2\mathbf{K}_{j,k} = s_{j,k},$$

for the SDP (5.138), depending on whether the Gram matrix  $\mathbf{G}$  or the square distance matrix  $\mathbf{S}$  is available. The centralization condition

$$\sum_{j=1}^n \mathbf{K}_{k,j} = 0, \quad i = 1, \dots, n,$$

is also needed to add in the constraint set.

4. **Build MVU kernel using SDP.** Apply an SDP algorithm, such as CSDP, SeDuMi, or SDTP3, to create the MVU kernel  $\mathbf{K}$  by solving the semidefinite programming problem (5.138).
5. **Spectral decomposition of MVU kernel.** Using a standard spectral decomposition algorithm to compute  $d$ -leading eigenvectors of  $\mathbf{K}$ , which provide the dimensionality reduction data set  $\mathcal{Y}$ . This step is the same as Step 4 in the Isomap algorithm.

The main step of the algorithm is Step 4, in which the SDP problem can be solved by an existent SDP package (see Subsection 5.7.2). We will not present Matlab code for MVU algorithm. The readers may refer to <http://www.cse.wustl.edu/~kilian/code/code.html>.

### 5.7.2 Semidefinite Programming

In this section, we briefly discuss the general formulation of a semidefinite programming(SDP). Let  $\mathbf{X}$  and  $\mathbf{C}$  be two  $n \times n$  symmetric matrices, which are considered as the vectors in  $\mathbb{R}^{n^2}$ . We define their inner product by

$$\langle \mathbf{C}, \mathbf{X} \rangle = \sum_{i,j=1}^n \mathbf{C}_{i,j} \mathbf{X}_{i,j}.$$

Thus,  $\mathbf{C}$  is a functional on the Euclidean space  $\mathbb{R}^{n^2}$ . Let  $\mathbf{A}_i$ ,  $1 \leq i \leq m$ , be  $m$  symmetric matrices in  $\mathcal{S}_n$  and  $\mathbf{b} = (b_1, \dots, b_m)^T \in \mathbb{R}^m$ . Then the following optimization problem is called a semidefinite programming (SDP):

$$\begin{aligned} & \text{minimize} && \langle \mathbf{C}, \mathbf{X} \rangle, \\ & \text{s.t.} && \langle \mathbf{A}_i, \mathbf{X} \rangle = b_i, \quad i = 1, \dots, m, \\ & && \mathbf{X} \succcurlyeq \mathbf{0}. \end{aligned} \tag{5.139}$$

It is easy to see that the collection  $\mathcal{S}_n^+$  is a convex set in  $\mathbb{R}^{n^2}$  and each constraint  $\langle \mathbf{A}_i, \mathbf{X} \rangle = b_i$  is a hyperplane in  $\mathbb{R}^{n^2}$ . Assume that  $\{\mathbf{A}_1, \dots, \mathbf{A}_m\}$  forms a linearly independent set in  $\mathbb{R}^{n^2}$ . Then the intersection of the  $m$  hyperplanes  $\langle \mathbf{A}_i, \mathbf{X} \rangle = b_i$ ,  $i = 1, \dots, m$ , (denoted by  $S_m$ ) is an  $(n^2 - m)$ -dimensional linear manifold, which is called an affine space. Hence, the semidefinite programming problem (5.139) can be interpreted as to find the minimal value of the linear function  $\langle \mathbf{C}, \mathbf{X} \rangle$  on the intersection of the convex set  $\mathbf{X} \succcurlyeq \mathbf{0}$  and the affine space  $S_m$ . The set of  $n \times n$  symmetric matrices that satisfy the constraints is called a feasible set of SDP. An SDP has a solution if the feasible set is nonempty.

In MVU, the kernel  $\mathbf{K}$  is the solution of the SDP optimization problem (5.138) with the particular setting

$$\mathbf{C} = -\mathbf{I}, \quad \mathbf{A}_i \triangleq (\mathbf{e}_j - \mathbf{e}_k)(\mathbf{e}_j - \mathbf{e}_k)^T, \quad b_i \triangleq s_{k,j},$$

where the double index  $(k, j)$  is one-to-one mapped to the single index  $i$  ( $1 \leq j, k \leq n$ ), and the binary matrix  $\mathbf{A}_i$  has only non-vanished entries at  $(k, k), (j, j), (k, j), (j, k)$ . Under this setting, the problem (5.138) always has a nonempty feasible set since the centering Gram matrix of  $\mathcal{X}$  is in the set. A common method to solve SDP is the interior point method. The following packages are designed for solving SDP problems: SDPA, CSDP, SDPT3, SeDuMi, DSDP, PENSDP, SDPLR, and SBmeth. More detailed discussion on SDP software can be found in [http://www-user.tu-chemnitz.de/~helmburg/sdp\\_software.html](http://www-user.tu-chemnitz.de/~helmburg/sdp_software.html).

### 5.7.3 Experiments and Applications of MVU

#### 5.7.3.1 Testing MVU Algorithm on Artificial Surfaces

We first test MVU method with the same set of 4 artificial surfaces, 3Dcluster, Punched Sphere, Swiss-roll, S-curve, which are used in testing Isomap algorithm. In Fig. 5.50 we apply MVU to reduce their dimension, using 5 nearest points in the neighborhood construction. The experiment shows that MVU works well for these surfaces, but the Swiss roll seems not truly unfolded perhaps because the DR data is not well scaled. The MVU method is not sensitive to the data smoothness. For example, although the data of 3Dcluster lack the smoothness on the line segments, MVU still works well for its DR.

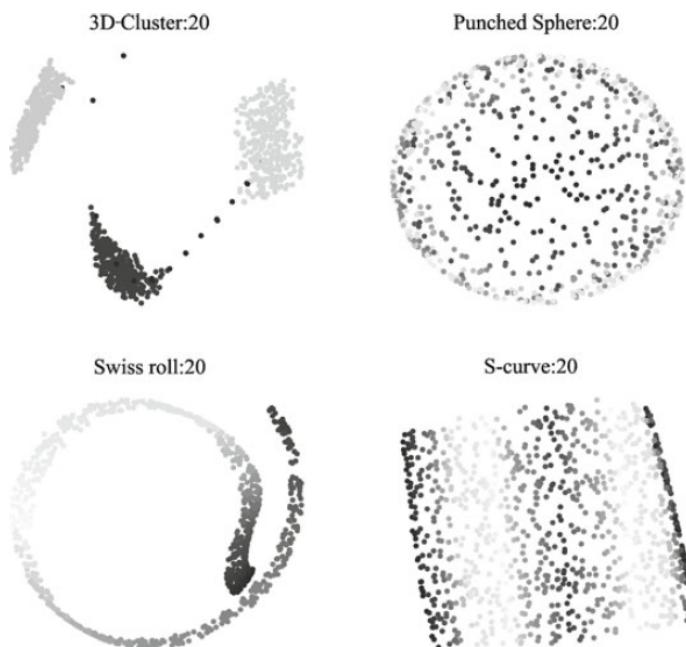


图 5.50: MVU algorithm is tested on 4 surfaces: 3D-cluster, Punched sphere, Swiss roll, and S-curve. 1000 points are randomly selected from each of these surfaces. All of them are reduced to 2-dimensional data (on the plane) by the algorithm, using 5 neighbors of each point. The 2-dimensional DR data of these 4 surface data are shown in the figure. Top left: 3D-cluster. Top right: Punched sphere. Bottom left: Swiss Roll. Bottom right: S-Curve.

In Fig. 5.51, we display how the size of neighborhood impacts the results. When the size of neighborhood is reduced from 5 (that is used in Fig. 5.50) to 3, the algorithm detects the disconnection of the data graph for 3D-cluster. Then no output is produced for the data. The DR results for Punched sphere and S-curve are still satisfactory, while

the result of Swiss Roll shows color mixture in some area. In general, when the size is too small, the data graph may not truly characterize the data similarity and therefore an incorrect dimensionality reduction data is produced. The relation between the size of neighborhood and the accuracy of the DR result is very complicated. A theoretical depiction is expected.

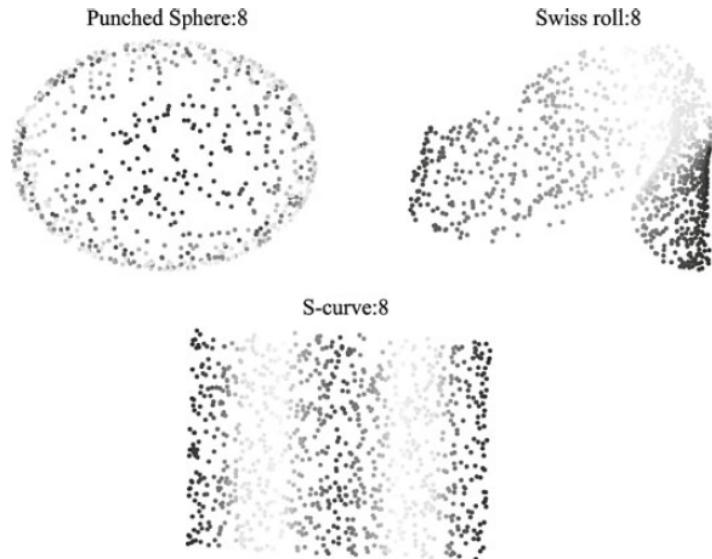


图 5.51: The neighbors for each point are reduced to 3. The robust connection algorithm is not applied in the test. The results indicate that the neighborhood size is too small for 3D-cluster. Hence no valid DR result is produced. The DR results for Punched sphere and S-curve are still satisfactory, while DR data for Swiss roll show mixed colors in some area.

## 5.7.4 Landmark MVU

### 5.7.4.1 Description of Landmark MVU

One of the major problems with MVU is that it requires the use of semidefinite programming, which is computationally expensive. For a data set with  $N$  sample vectors, using  $k$  nearest points, the semidefinite programming employs total  $k(k - 1)/2 \times N$  constraints in (5.134). When  $N$  increases, the number of constraints becomes too large to run the algorithm effectively. MVU is therefore limited to no more than 2000 points which use no more than 6 neighbors. If these bounds are breached, then the runtime of the algorithm becomes unreasonably long. This limitation has become more pressing recently, with the increasing availability of very large data sets and the corresponding increasing

need for scalable dimensionality reduction algorithms. Landmark MVU (LMVU) was introduced to overcome the difficulty caused by large size samples. Landmark MVU technique first applies a standard MVU algorithm on a subset of the samples called landmarks, and then extends the DR results on the landmarks to other samples.

Landmark MVU is effectively online with respect to the introduction of new data points. After the landmark points are selected and the initial calculation is carried out, all other points are embedded independently from each other using a fixed linear transformation. A global calculation is necessary only if the embedding coordinates are required to be aligned with the principal axes of the data. Landmark MVU merges manifold interpolation into positively semidefinite programming. The strategy of LMVU can be sketched as follows.

Let  $\mathcal{X} \subset \mathbb{R}^D$  be the observed data set and  $\mathcal{Y} \subset \mathbb{R}^d$  be the DR data set of  $\mathcal{X}$ , where  $d \leq D$ . From the set  $\mathcal{Y}$  we can find a subset  $\mathcal{Y}_m$  having  $m$  points,  $d < m$ , such that  $\mathcal{Y}_m$  spans  $\mathbb{R}^d$ . We denote the data matrix corresponding to  $\mathcal{Y}_m$  by  $\mathbf{U}$ . Then there is an  $n \times m$  matrix  $\mathbf{Q}$  such that

$$\mathbf{Y} = \mathbf{U}\mathbf{Q}^T. \quad (5.140)$$

The  $m \times m$  matrix:

$$\mathbf{L} = \mathbf{U}^T \mathbf{U} \quad (5.141)$$

is a psd one. Hence

$$\mathbf{K} = \mathbf{Q}\mathbf{L}\mathbf{Q}^T \quad (5.142)$$

is a DR kernel, whose spectral decomposition produces the DR set  $\mathcal{Y}$ . The landmark MVU is the method that finds the DR set  $\mathcal{Y}$  via the constructions of the linear transformation  $\mathbf{Q}$  and the spd matrix  $\mathbf{L}$ . Note that the input data of LMVU is the observed data set  $\mathcal{X}$ . Therefore, the matrices  $\mathbf{Q}$  and  $\mathbf{L}$  in (5.142) need to be computed from  $\mathcal{X}$ . Hence, an LMVU algorithm consists of the following steps.

1. As usual, define a neighborhood system on the data  $\mathcal{X}$ , which is represented by a graph  $G = (\mathcal{X}, \mathbf{A})$ , where  $\mathbf{A}$  is the adjacency matrix.
2. Select a landmark set from  $\mathcal{X}$ .
3. Find the linear transformation  $\mathbf{Q}$  which (approximately) recovers all points in  $\mathcal{X}$  by the linear combinations of landmarks. Therefore, once the DR data of landmarks are computed, the DR data of  $\mathcal{X}$  can be computed by (5.140) too.
4. Construct the landmark MVU kernel  $\mathbf{L}$  and make the spectral decomposition of  $\mathbf{L}$ .

The first step is standard. We already studied the construction of graph neighborhood in Chapter 三. In the following we shall only study the constructions of  $\mathbf{Q}$  and  $\mathbf{L}$ .

#### 5.7.4.2 Linear Transformation from Landmarks to Data Set

We first study the construction of the linear transformation  $\mathbf{Q}$ . As usual, we assume that the data set  $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subset \mathbb{R}^D$  resides on a  $d$ -manifold  $M \subset \mathbb{R}^D$  ( $d < D$ ). Let  $h$  be the coordinate mapping on  $M$  so that the DR set  $\mathcal{Y} = h(\mathcal{X})$  is a manifold coordinate representation of  $\mathcal{X}$ . Let  $\mathcal{Y}_m = h(\mathcal{X}_m)$ , where  $\mathcal{X}_m$  is the landmark set for LMVU. For simplicity, without loss of generality, we assume that the landmark set  $\mathcal{X}_m$  consists of the first  $m$  points of the data set  $\mathcal{X}$ . That is

$$\mathcal{X}_1 \triangleq \mathcal{X}_m = \{\mathbf{x}_1, \dots, \mathbf{x}_m\}.$$

Otherwise, we can permute the data matrix  $\mathbf{X}$  to make it true. Let  $\mathbf{X}_1$  denote the matrix corresponding to  $\mathcal{X}_1$  and let  $\mathbf{Y}_1$  denote the matrix corresponding to  $\mathcal{Y}_1 \triangleq \mathcal{Y}_m$ . Since we construct  $\mathbf{Q}$  based on  $\mathcal{X}$ , the relation of  $\mathcal{Y}$  and  $\mathcal{Y}_1$  in (5.140) needs to be converted to a relation between  $\mathcal{X}$  and  $\mathcal{X}_1$ . Write  $\mathcal{X}_2 = \mathcal{X} \setminus \mathcal{X}_1$ ,  $\mathcal{Y}_2 = \mathcal{Y} \setminus \mathcal{Y}_1$ , and denote their corresponding matrices by  $\mathbf{X}_2$  and  $\mathbf{Y}_2$  respectively. By  $\mathcal{Y} = h(\mathcal{X})$  and  $\mathcal{Y}_1 = h(\mathcal{X}_1)$ , we rewrite (5.140) as

$$h(\mathbf{X}) = h(\mathbf{X}_1)\mathbf{Q}^T.$$

Setting

$$\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2),$$

$$\mathbf{Y} = h(\mathbf{X}) = (h(\mathbf{X}_1), h(\mathbf{X}_2)) = (\mathbf{Y}_1, \mathbf{Y}_2),$$

and

$$\mathbf{Q}^T = (\hat{\mathbf{Q}}_1, \hat{\mathbf{Q}}_2),$$

we have

$$(\mathbf{Y}_1, \mathbf{Y}_2) = (\mathbf{Y}_1 \hat{\mathbf{Q}}_1, \mathbf{Y}_1 \hat{\mathbf{Q}}_2),$$

which yields  $\hat{\mathbf{Q}}_1 = \mathbf{I}_m$  and

$$\mathbf{Y}_2 = \mathbf{Y}_1 \hat{\mathbf{Q}}_2. \quad (5.143)$$

Hence, to find  $\mathbf{Q}$ , we only need to construct  $\hat{\mathbf{Q}}_2$ . To do so, we consider the neighborhood structure on  $\mathcal{X}$ . Assume that the high dimensional data  $\mathcal{X}$  is well sampled on the manifold  $M \subset \mathbb{R}^D$  and a consistent neighborhood system is defined on  $\mathcal{X}$ , which generates the data graph  $G = (\mathcal{X}, \mathbf{A})$ . Since the neighborhood is well defined, we can approximately recover each  $\mathbf{x}_i \in \mathcal{X}$  by a weighted sum of its neighbors. Denote by  $O(i)$  the neighbors of  $\mathbf{x}_i$ ,

and denote by  $N(i) = \{i_1, \dots, i_k\}$  the index set of the vectors in  $O(i)$ . Then each point  $\mathbf{x}_i \in \mathcal{X}_2$  can be approximately represented as a weighted sum of its neighbors:

$$\mathbf{x}_i \approx \sum_{j \in N(i)} w_{ji} \mathbf{x}_j, \quad m+1 \leq i \leq n. \quad (5.144)$$

Since  $\mathbf{x}_j \approx \mathbf{x}_i$  if  $j \in N(i)$ , it is reasonable to require the weight sum to be unit:

$$\sum_{j \in N(i)} w_{ji} = 1. \quad (5.145)$$

We call the weights in (5.144) reconstruction weights. By (5.145), we rewrite (5.144) as

$$\sum_{j \in N(i)} w_{ji} \hat{\mathbf{x}}_j \approx \mathbf{0}, \quad m+1 \leq i \leq n,$$

where  $\hat{\mathbf{x}}_j = \mathbf{x}_i - \mathbf{x}_j$ . Therefore, the weights  $w_{ji}$  can be obtained as the solution of the minimization problem

$$(w_{ji})_{j \in N(i)} = \operatorname{argmin}_{\sum w_{ji}=1} \left\| \sum_{j \in N(i)} w_{ji} \hat{\mathbf{x}}_j \right\|^2. \quad (5.146)$$

Applying Lagrange's method of multipliers to (5.146), we obtain the Lagrangian function

$$L((w_{ji})_{j \in N(i)}, \lambda) = \left\| \sum_{j \in N(i)} w_{ji} \hat{\mathbf{x}}_j \right\|^2 - \lambda \sum_{j \in N(i)} w_{ji}. \quad (5.147)$$

Write  $\mathbf{w}_i = (w_{i1,i}, \dots, w_{ik,i})^T$  and  $\mathbf{A}_i = \sum_{j \in N(i)} \hat{\mathbf{x}}_j \hat{\mathbf{x}}_j^T$ . Then (5.147) leads to

$$\mathbf{A}_i \mathbf{w}_i = \lambda \mathbf{1}, \quad \lambda = \frac{1}{\sum_{j \in N(i)} w_{ji}}. \quad (5.148)$$

The weights  $w_{ji}$  constitute an  $n \times (n-m)$  matrix  $\mathbf{W} = (w_{ji})$ .

We split the sum on the right-hand side of (5.144) into two sums,

$$\sum_{j \in N(i), j \leq m} w_{ji} \mathbf{x}_j + \sum_{j \in N(i), j > m} w_{ji} \mathbf{x}_j, \quad m+1 \leq i \leq n. \quad (5.149)$$

in which the vectors in the first sum is in the set  $\mathcal{X}_1$  and the vectors in the second sum is in the set  $\mathcal{X}_2$ . Therefore, we can write (5.144) in the matrix form

$$\mathbf{X}_2 \approx \mathbf{X}_1 \mathbf{W}_1 + \mathbf{X}_2 \mathbf{W}_2, \quad (5.150)$$

where  $\mathbf{W}_1$  is constituted by the weights in the first sum of (5.149) while  $\mathbf{W}_2$  is constituted by the weights in the second sum. The approximation in (5.150) is the best under the norm in (5.146).

By (5.144), the approximation in (5.150) is local for each  $\mathbf{x}_j$ . Since the neighborhood system is well defined on the manifold  $M$ , we have

$$h(\mathbf{x}_i) \approx \sum_{j \in N(i)} w_{ji} h(\mathbf{x}_j), \quad m+1 \leq i \leq n, \quad (5.151)$$

which yields

$$\mathbf{Y}_2 \approx \mathbf{Y}_1 \mathbf{W}_1 + \mathbf{Y}_2 \mathbf{W}_2, \quad (5.152)$$

or equivalently,

$$\mathbf{Y}_2(\mathbf{I} - \mathbf{W}_2) \approx \mathbf{Y}_1 \mathbf{W}_1. \quad (5.153)$$

Solving the equation directly, we get

$$\mathbf{Y}_2 \approx \mathbf{Y}_1 \mathbf{W}_1 (\mathbf{I} - \mathbf{W}_2)^+, \quad (5.154)$$

where  $+$  denotes the pseudo-inverse, which can be computed by

$$(\mathbf{I} - \mathbf{W}_2)^+ = (\mathbf{I} - \mathbf{W}_2)^T [(\mathbf{I} - \mathbf{W}_2)(\mathbf{I} - \mathbf{W}_2)^T]^{-1}.$$

Comparing (5.154) with (5.143), we can compute  $\hat{\mathbf{Q}}_2$  using the formula

$$\hat{\mathbf{Q}}_2 = \mathbf{W}_1 (\mathbf{I} - \mathbf{W}_2)^+. \quad (5.155)$$

#### 5.7.4.3 Construction of Kernel of Landmark MVU

The kernel of landmark MVU is represented in (5.142). In the previous subsection, we already constructed the linear transformation  $\mathbf{Q}$ . Similar to the construction of a regular MVU kernel, we can compute the psd matrix  $\mathbf{L}$  in (5.142) by applying semidefinite programming to the landmark set. For convenience, we still assume that the landmark set is  $\mathcal{X}_1$ , and write  $\mathbf{K} = \mathbf{Q} \mathbf{L} \mathbf{Q}^T$ . Similar to the regular MVU model (5.138), the semidefinite programming for  $\mathbf{L}$  is the following maximization problem:

$$\begin{aligned} & \text{maximize} \quad \text{tr}(\mathbf{Q} \mathbf{L} \mathbf{Q}^T), \\ & \text{s.t.} \quad \sum_{i,j=1}^n \mathbf{K}_{i,j} = 0, \\ & \quad \mathbf{K}_{k,k} + \mathbf{K}_{j,j} - 2\mathbf{K}_{k,j} = \mathbf{S}_{k,j}, j, k \in N(i), j > k, 1 \leq i \leq n, \\ & \quad \mathbf{L} \succcurlyeq \mathbf{0}, \\ & \quad \mathbf{L} \in \mathcal{S}_n. \end{aligned} \quad (5.156)$$

To find  $\mathbf{L}$ , the semidefinite programming (5.156) has only about  $k(k-1)/2 \times m$  constraints. Since  $m$  is a small number, the computational cost for solving (5.156) is quite low.

#### 5.7.4.4 Experiments of LMVU

In Fig. 5.52, we illustrate the validity of LMVU method for dimensionality reduction when the size of the data set is large. We test LMVU algorithm for the Swiss roll data with 2000 sample points. 3-neighborhood is used to construct the data graph. We randomly select 61 landmarks from 2000 samples to construct the kernel of LMVU. The number of constraints for the corresponding semidefinite programming is now reduced to 367 and the total running time is reduced to 0.49 min.

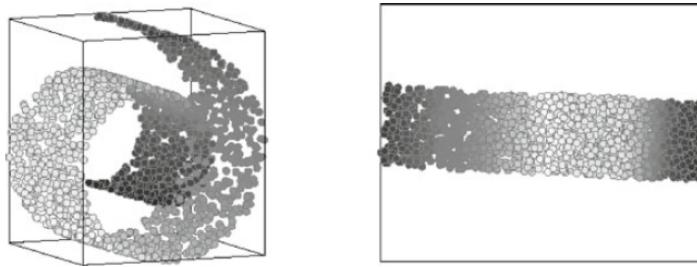


图 5.52: Landmark MVU is applied to reducing the dimension of Swiss roll with 2000 sample points and 3 neighbors of each point. A regular MVU algorithm needs 30 minutes, by iterating 50 times for the DR. The total independent constraints in the SDP are 4316. The displayed result is obtained by LMVU with randomly selected 61 landmarks from 2000 samples and 3 neighbors of each point. It takes 52 times iterating, and number of independent constraints is 367. Total running time is only 0.49 min.

#### 5.7.5 Conclusion

The MVU method has properties similar to those of Isomap method. It is computationally stable but the MVU DR kernel is dense and MVU algorithm is computationally intensive. MVU algorithm can be used to solve the problem, in which only the similarity matrix of data is available. In this sense, it can be considered as a nonlinear multidimensional scaling method. To reduce the computing cost, landmark MVU method is applied. The experiments show that LMVU performs fast and still yields satisfactory results in many cases.

**练习103.** Use one person's face images in *Data for MATLAB hackers* (<http://www.cs.nyu.edu/~roweis/data.html>) as the data for dimensionality reduction with MVU, where the target dimension is 2, and see whether it is better than CMDS, ISOMAP, LLE, LTSA, LE, and Dmaps, e.g., whether the face images are arranged better in some order. Different neighborhood sizes and distance measures are encouraged to be tried.

## 5.8 Summary of Nonlinear DR Methods

Now we summarize the nonlinear DR methods: Isomap, LLE, LTSA, LE, Dmaps, and MVU. We also include HLLE, Logmap, and RML for completeness.

First, they share very similar paradigm:

1. **Build topology** of the data set  $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$  by  $k$ -NN or  $\varepsilon$ -NN;
2. **Minimize a distortion** written as:

$$Q(f) = \sum_i Q_i(f),$$

where  $Q_i(f)$  is a symmetric, positive semi-definite quadratic form that measures the local variations of  $f$  around  $\mathbf{x}_i$ .

3. **Impose quadratic/linear constraints** to avoid trivial or non-unique solutions.
4. **Solve an eigenvalue problem.** The leading eigenvectors that correspond to maximal or minimal eigenvalues provide the optimal embedding coordinates.

Second, comparison among nonlinear DR algorithms:

1. LLE, HLLE, and LE use sparse matrices, so can scale to large data sets, but do not reveal the intrinsic dimensionality.
2. Isomap, LE, and LLE can be incorporated in the kernel framework.
3. LTSA is less sensitive to choice of  $k$  than LLE (Figure 5.53).
4. Nonlinear algorithms require dense sampling. Otherwise, they have more distortion than MDS (Figure 5.54).
5. Isomap is sensitive to shortcut edges. Local methods do not propagate such errors. Diffusion maps is insensitive to shortcut edges.
6. Local methods only look good: sensitive to noise.

Third, we summarize the properties of nonlinear DR methods in Table 5.1.

Todd Wittman conducted experiments to compare various DR methods in various aspects. His slides are available at [http://www.math.pku.edu.cn/teachers/yaoy/Spring2011/lecture11\\_3\\_mani.pdf](http://www.math.pku.edu.cn/teachers/yaoy/Spring2011/lecture11_3_mani.pdf).

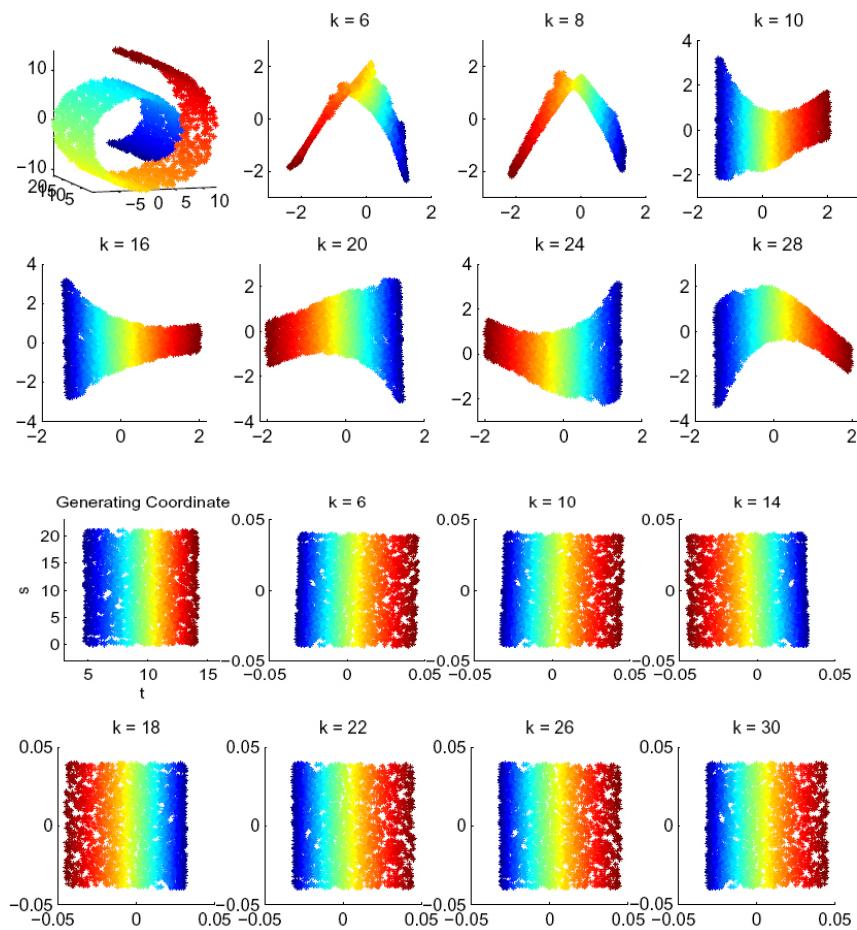


图 5.53: Comparison between LLE (top) and LTSA (bottom) on the sensitivity to the neighborhood size  $k$ .

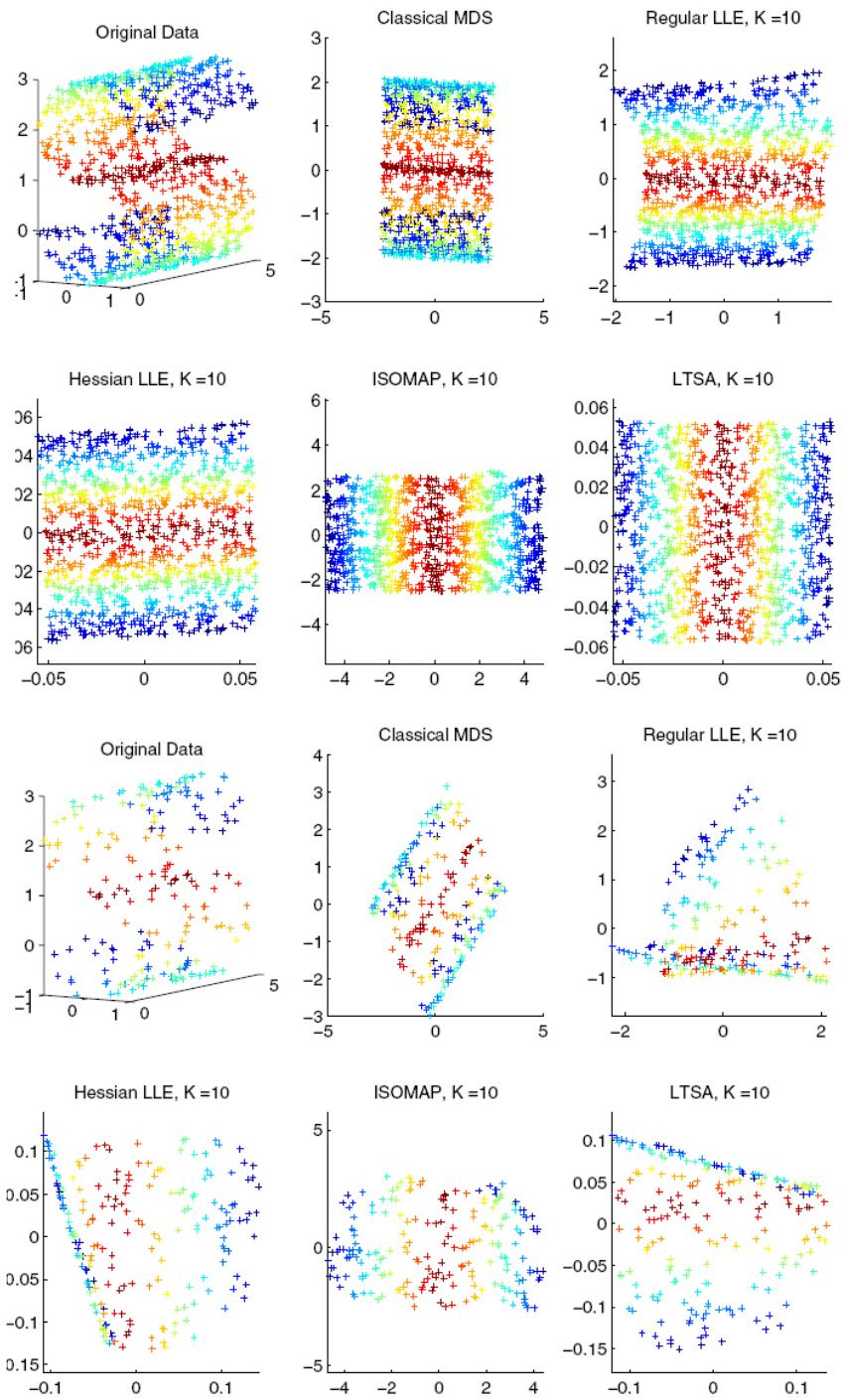


图 5.54: Comparison between nonlinear DR methods and MDS on the sensitivity to the sample density. Top: Densely sampled. Bottom: Sparsely sampled.

| Algorithm      | Preserved Property            | Local or Global | Complexity     |
|----------------|-------------------------------|-----------------|----------------|
| Isomap         | geodesic distance             | global          | very high      |
| LLE            | local linearity               | local           | low            |
| LE             | affinity                      | local           | low            |
| HLLE           | local isometry                | local           | high           |
| LTSA           | local coordinate              | both            | low            |
| Logmap         | geodesic distance & direction | local           | very low       |
| Diffusion Maps | diffusion distance            | global          | moderate       |
| MVU            | local distance                | both            | extremely high |

表 5.1: Comparison among nonlinear DR methods.

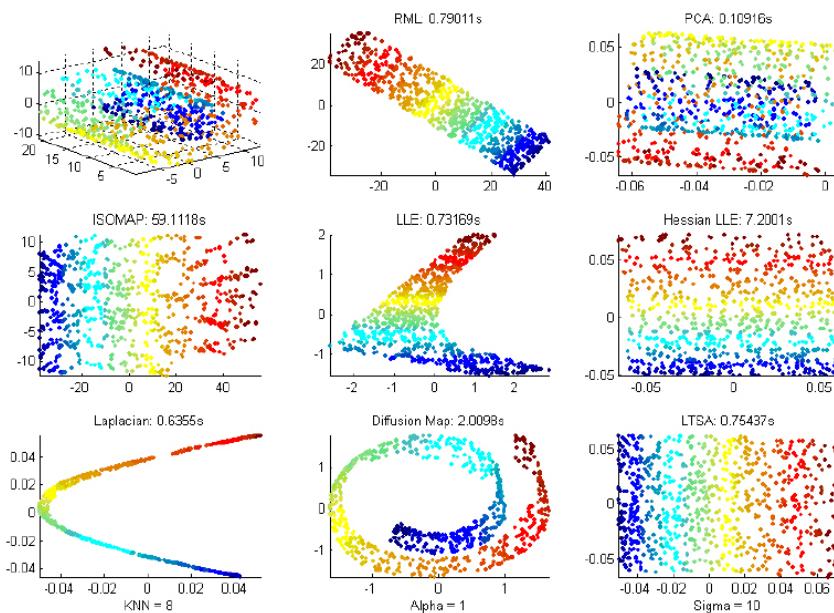


图 5.55: Comparison between nonlinear DR methods on Swiss Roll.

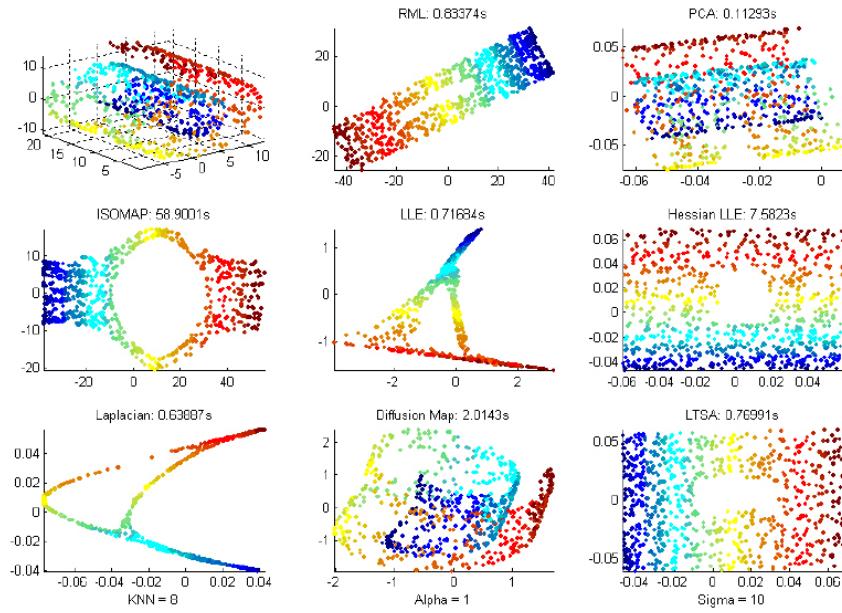


图 5.56: Comparison between nonlinear DR methods on Swiss Hole.

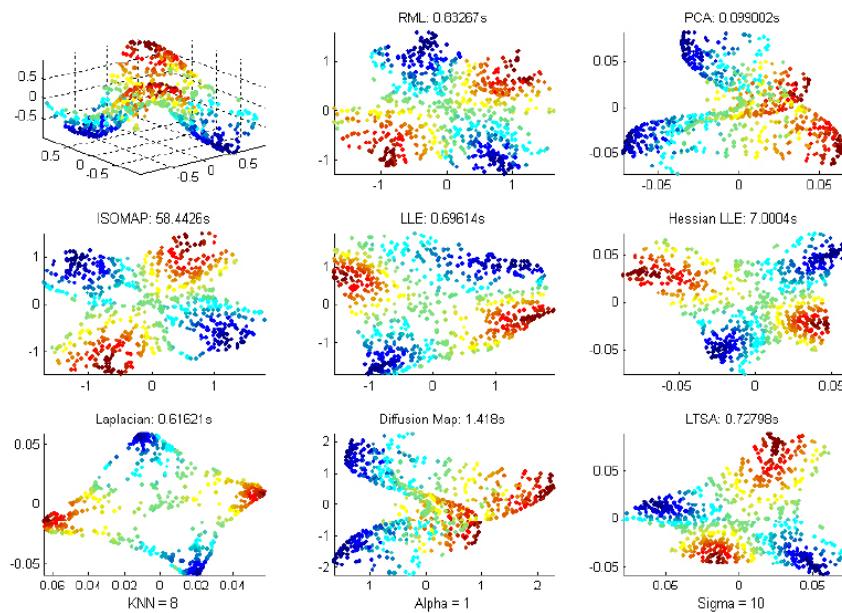


图 5.57: Comparison between nonlinear DR methods on Swiss Twin Peaks.

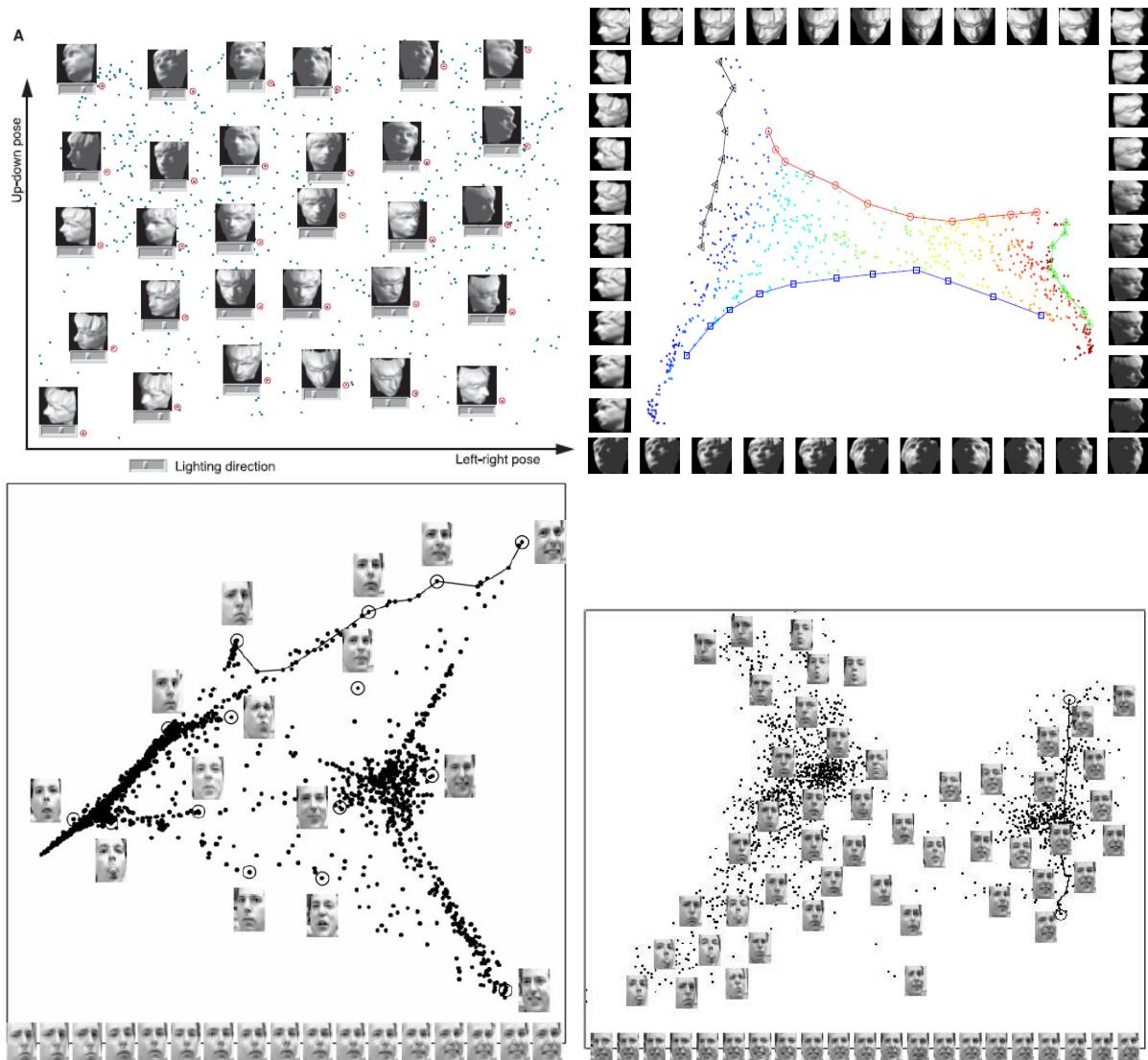


图 5.58: Comparison between nonlinear DR methods on face data. Top Left: Isomap. Top Right: LTSA. Bottom Left: LLE. Bottom Right: LE.

For visualization, nonlinear DR methods are comparable only when the test manifold is already understood (Figures 5.55-5.57). For real data sets, it is hard to compare which performs better (Figure 5.58). Axiomatic approaches of designing new algorithms, e.g., performing perfectly on ideal manifolds, should be developed.

Finally, we comment on the current status of manifold learning. Basically, there are too many independent algorithms. Each has (severe) drawbacks and (limited) advantages, tested on selected data sets. The evaluation of performance is difficult: no criteria. And for most of the DR methods, solid theoretical analysis is lacking.

Further readings:

1. Ham et al. [58] gave a unified kernel view on the most representative DR methods. They showed that three algorithms, Isomap, Laplacian eigenmaps, and LLE can be interpreted as kernel PCA with different kernel matrices.
2. Maaten et al. [130] also compared various DR methods.

## 第六章 一阶稀疏表示

(The materials are taken from [44].)

This field is all about one specific mathematical model for signal sources. Modeling sources is key in signal and image processing. Armed with a proper model, one can handle various tasks such as denoising, restoration, separation, interpolation and extrapolation, compression, sampling, analysis and synthesis, detection, recognition, and more. Indeed, a careful study of the signal and image processing literature reveals that there is an evolution of such models and their use in applications. This course is about one such model, which I call *Sparse-Land* for brevity. This specific model is intriguing and fascinating because of the beauty of its theoretical foundations, the superior performance it leads to in various applications, its universality and flexibility in serving various data sources, and its unified view, which makes all the above signal processing tasks clear and simple.

At the heart of this model lies a simple linear system of equations, the kind of which seems long studied in linear algebra. A full-rank matrix  $\mathbf{A} \in \mathbb{R}^{n \times m}$  with  $n < m$  generates an underdetermined system of linear equations  $\mathbf{Ax} = \mathbf{b}$  having infinitely many solutions. We are interested in seeking its sparsest solution, i.e., the one with the fewest nonzero entries. Can such a solution ever be unique? If so, when? How can such a solution be found in reasonable time? It is hard to believe, but these questions and more of the like form the core engine to this model and the wide field that has grown around it. Positive and constructive answers to these questions in recent years expose a wide variety of surprising phenomena, and those set the stage for *Sparse-Land* to serve in practice. Clearly, research in this field calls for an intensive use of knowledge from linear algebra, optimization, scientific computing, and more.

This field is relatively young. Early signs of its core ideas appeared in a pioneering work by Stephane Mallat and Zhifeng Zhang in 1993, with the introduction of the concept of dictionaries, replacing the more traditional and critically-sampled wavelet transform. Their work put forward some of the core ideas that later became central in this field, such as a greedy pursuit technique that approximates a sparse solution to an underdetermined linear system of equations, characterization of dictionaries by their coherence measure, and more.

A second key contribution in 1995 was by Scott Shaobing Chen, David Donoho, and Michael Saunders, who introduced another pursuit technique that uses the  $\ell_1$ -norm for evaluating sparsity. Surprisingly, they have shown that the quest for the sparsest solution

could be tackled as a convex programming task, often leading to the proper solution. With these two contributions, the stage was set for a deeper analysis of these algorithms and their deployment in applications. A crucial step towards this goal was made in 2001, with the publication of the work by Donoho and Huo. In their daring paper, Donoho and Huo defined and (partly) answered what later became a key question in this field: Can one guarantee the success of a pursuit technique? Under what conditions? This line of analysis later became the skeleton of this field, providing the necessary theoretical backbone for the Sparse-Land model. It is amazing to see how fast and how vast this area of research has grown in recent years, with hundreds of interested researchers, various workshops, sessions, and conferences, and an exponentially growing number of papers.

The activity in this field is spread over all major universities and research organizations around the world, with leading scientists from various disciplines. As this field of research lies at the intersection between signal processing and applied mathematics, active in this field are mathematicians interested in approximation theory, applied mathematicians interested in harmonic analysis, statisticians, and engineering from various fields (computer science, electrical engineering, geophysics, and more).

## 6.1 Prologue

A central achievement of classical linear algebra is a thorough examination of the problem of solving systems of linear equations. The results – definite, timeless, and profound – give the subject a completely settled appearance. As linear systems of equations are the core engine in many engineering developments and solutions, much of this knowledge is practically and successfully deployed in applications. Surprisingly, within this well-understood arena there is an elementary problem that has to do with sparse solutions of linear systems, which only recently has been explored in depth; we will see that this problem has surprising answers, and it inspires numerous practical developments. In this chapter we shall concentrate on defining this problem carefully, and set the stage for its answers in later sections.

### 6.1.1 Underdetermined Linear Systems

Consider a matrix  $\mathbf{A} \in \mathbb{R}^{n \times m}$  with  $n < m$ , and define the underdetermined linear system of equations  $\mathbf{Ax} = \mathbf{b}$ . This system has more unknowns than equations, and thus it has either no solution, if  $\mathbf{b}$  is not in the span of the columns of the matrix  $\mathbf{A}$ , or infinitely many solutions. In order to avoid the anomaly of having no solution, we shall hereafter assume that  $\mathbf{A}$  is a full-rank matrix, implying that its columns span the entire space  $\mathbb{R}^n$ .

In engineering we often encounter problems formulated as such underdetermined linear systems of equations. As an example from image processing, consider the image scale-up problem, where an unknown image undergoes a blur and scale-down operation, and the outcome is given as a lower quality and smaller image  $\mathbf{b}$ . The matrix  $\mathbf{A}$  stands for the degradation operations, and our aim is to reconstruct the original image  $\mathbf{x}$  from the given measurements  $\mathbf{b}$ . Clearly, there are infinitely many possible images  $\mathbf{x}$  that may “explain”  $\mathbf{b}$ , among which there are some that may look better than others. How can we find the proper  $\mathbf{x}$ ?

### 6.1.2 Regularization

Both in the above example and in many other problems that assume the same formulation, we desire a single solution, and the fact that there are infinitely many of those stands as a major obstacle. In order to narrow this choice to one well-defined solution, additional criteria are needed. A familiar way to do this is regularization, where a function  $J(\mathbf{x})$  that evaluates the desirability of a would-be solution  $\mathbf{x}$  is introduced, with smaller values being preferred. Defining the general optimization problem ( $P_J$ )

$$(P_J) : \min_{\mathbf{x}} J(\mathbf{x}), \quad s.t. \quad \mathbf{b} = \mathbf{Ax}, \quad (6.1)$$

it is now in the hands of  $J(\mathbf{x})$  to govern the kind of solution(s) we may obtain. Returning to the image scale-up example, a function  $J(\mathbf{x})$  that prefers smooth or, better yet, piecewise smooth results, is typically used.

The most known choice of  $J(\mathbf{x})$  is the squared Euclidean norm  $\|\mathbf{x}\|_2^2$ . The problem ( $P_2$ ) that results from such a choice has in fact a unique solution  $\hat{\mathbf{x}}$  – the so called minimum-norm solution. Using Lagrange multipliers, we define the Lagrangian

$$\mathcal{L}(\mathbf{x}) = \|\mathbf{x}\|_2^2 + \lambda^T (\mathbf{Ax} - \mathbf{b}), \quad (6.2)$$

with  $\lambda$  being the Lagrange multipliers for the constraint set. Taking a derivative of  $\mathcal{L}(\mathbf{x})$  with respect to  $\mathbf{x}$ , we obtain the requirement

$$\frac{\partial \mathcal{L}(\mathbf{x})}{\partial \mathbf{x}} = 2\mathbf{x} + \mathbf{A}^T \lambda. \quad (6.3)$$

Thus the solution is obtained as

$$\hat{\mathbf{x}}_{opt} = -\frac{1}{2} \mathbf{A}^T \lambda. \quad (6.4)$$

Plugging this solution into the constraint  $\mathbf{Ax} = \mathbf{b}$  leads to

$$\mathbf{A}\hat{\mathbf{x}}_{opt} = -\frac{1}{2} \mathbf{A}\mathbf{A}^T \lambda = \mathbf{b} \Rightarrow \lambda = -2(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{b}. \quad (6.5)$$

Assigning this to Equation (6.4) gives the well-known closed-form pseudo-inverse solution

$$\hat{\mathbf{x}}_{opt} = -\frac{1}{2}\mathbf{A}^T \lambda = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{b} = \mathbf{A}^+\mathbf{b}. \quad (6.6)$$

Note that since we have assumed that  $\mathbf{A}$  is full-rank, the matrix  $\mathbf{A}\mathbf{A}^T$  is positive definite and thus invertible. We also note that for a more general choice of the form  $J(\mathbf{x}) = \|\mathbf{B}\mathbf{x}\|_2^2$  (such that  $\mathbf{B}^T\mathbf{B}$  is invertible), a closed-form expression of the solution also exists,

$$\hat{\mathbf{x}}_{opt} = (\mathbf{B}^T\mathbf{B})^{-1}\mathbf{A}^T(\mathbf{A}(\mathbf{B}^T\mathbf{B})^{-1}\mathbf{A}^T)^{-1}\mathbf{b}, \quad (6.7)$$

following the same line of steps. The use of  $\ell_2$  is widespread in various fields of engineering, and this is mainly due to its simplicity as manifested by the above closed-form and unique solution. In signal and image processing, this regularization has been practiced quite often for various inverse problems, representation of signals, and elsewhere. This, however, is by no means a declaration that  $\ell_2$  is the truly best choice for the various problems it has been brought to serve. In many cases, the mathematical simplicity of  $\ell_2$  is a misleading fact that diverts engineers from making better choices for  $J(\cdot)$ . Indeed, in image processing, after three decades of trying to successfully activate the ( $\ell_2$ -norm based) Wiener filter, it was eventually realized that the solution resides with a different, robust-statistics based choice of  $J(\cdot)$ , of the kind we shall often meet in this section.

### 6.1.3 The Temptation of Convexity

The fact that  $\ell_2$  leads to a unique solution is a manifestation of a wider phenomenon, as selecting any strictly convex function  $J(\cdot)$  guarantees such uniqueness. Let us recall the definition of convexity, first for sets, and then for functions.

**定义104.** A set  $\Omega$  is convex if  $\forall \mathbf{x}_1, \mathbf{x}_2 \in \Omega$  and  $\forall t \in [0, 1]$ , the convex combination  $\mathbf{x} = t\mathbf{x}_1 + (1 - t)\mathbf{x}_2$  is also in  $\Omega$ .

It is easy to verify that the set  $\Omega = \{\mathbf{x} | \mathbf{Ax} = \mathbf{b}\}$  is convex, using the above definition. Thus, the feasible set of solutions of the optimization problem posed in Equation (6.1) is convex. In order for this optimization problem to be convex as a whole, we have to add a convexity requirement on the penalty  $J(\mathbf{x})$ . The core definition for such a property is given by:

**定义105.** A function  $J(\mathbf{x}) : \Omega \rightarrow \mathbb{R}$  is convex if  $\forall \mathbf{x}_1, \mathbf{x}_2 \in \Omega$  and  $\forall t \in [0, 1]$ , the convex combination point  $\mathbf{x} = t\mathbf{x}_1 + (1 - t)\mathbf{x}_2$  satisfies

$$J(t\mathbf{x}_1 + (1 - t)\mathbf{x}_2) \leq tJ(\mathbf{x}_1) + (1 - t)J(\mathbf{x}_2). \quad (6.8)$$

Another way to understand the above definition is this: the epigraph of  $J(\mathbf{x})$  (the area defined by  $\{(\mathbf{x}, y) | y \geq J(\mathbf{x})\}$ ) is a convex set in  $\mathbb{R}^{m+1}$ .

If  $J(\cdot)$  is twice continuously differentiable, its derivatives can be used for alternative definitions of convexity:

$$J(\mathbf{x}_2) \geq J(\mathbf{x}_1) + \nabla J(\mathbf{x}_1)^T (\mathbf{x}_2 - \mathbf{x}_1), \quad (6.9)$$

or alternatively, if and only if the Hessian  $\nabla^2 J(\mathbf{x}_1)$  is positive semi-definite.

Using the definition of the positive semi-definiteness of the Hessian makes the proof of the convexity of the squared  $\ell_2$ -norm trivial, since  $\nabla^2 \|\mathbf{x}\|_2^2 = 2\mathbf{I} \succeq \mathbf{0}$ . Indeed, the squared  $\ell_2$ -norm is strictly convex since this Hessian is positive-definite for all  $\mathbf{x}$ . Returning to the problem posed in Equation (6.2), since the constraint-set is convex and the penalty is strictly so, a unique solution is guaranteed.

So far we have dwelled with the choice  $J(\mathbf{x}) = \|\mathbf{x}\|_2^2$ . However, there are many other choices of functions  $J(\cdot)$  that are convex or strictly so. Even though these options of  $J(\cdot)$  rarely lead to a closed-form solution, the fact that they are strictly convex implies a unique solution. Perhaps more importantly, carefully chosen optimization algorithms for solving (6.2) are guaranteed to converge to the global minimizer of such optimization problems.

These properties make convex problems very appealing in engineering in general, and in signal and image processing in particular. Again, this does not mean that there is no value to non-convex problems – it simply says that when dealing with non-convex optimization tasks, one should be aware of problems that might be encountered because of their imperfection.

Special cases of interest for convex functions are all the  $\ell_p$ -norms for  $p \geq 1$  (use the Hessian to verify that). These are defined as

$$\|\mathbf{x}\|_p^p = \sum_i |x_i|^p. \quad (6.10)$$

In particular, the  $\ell_\infty$ -norm (obviously, without the  $p$ -th power) and the  $\ell_1$ -norm are very interesting and popular; the  $\ell_\infty$  considers the maximal entry of the vector  $\mathbf{x}$ , and  $\ell_1$  sums the absolute entries. We shall have a special interest in  $\ell_1$  due to its tendency to sparsify the solution – a fact that will be discussed and established as we move forward in this section.

#### 6.1.4 A Closer Look at $\ell_1$ Minimization

The choice  $J(\mathbf{x}) = \|\mathbf{x}\|_1$  is convex but not strictly so, a fact easily verified by noticing that if  $\mathbf{x}_1$  and  $\mathbf{x}_2$  are in the same quadrant (i.e., they have the same sign-pattern), then

their convex combination gives an equality in Equation (6.8). Thus, the problem

$$(P_1) : \min_{\mathbf{x}} \|\mathbf{x}\|_1, \quad s.t. \quad \mathbf{b} = \mathbf{Ax} \quad (6.11)$$

may have more than one solution. Nevertheless, even if there are infinitely many possible solutions to this problem, we can claim that

**命题106.** (i) *these solutions are gathered in a set that is bounded and convex, and*  
 (ii) *among these solutions, there exists at least one with at most  $n$  non-zeros (as the number of constraints).*

*Proof.* The convexity of the solution set (in the first property) is immediate, stemming as a direct consequence of the fact that all optimal solutions have the same penalty ( $\ell_1$ -length in this case); any convex combination of two such solutions must give a lower or equal penalty due to the convexity of  $J(\mathbf{x}) = \|\mathbf{x}\|_1$ , and since the two chosen solutions are optimal, a lower penalty is impossible to obtain. Thus, any convex combination of two optimal solutions is also an optimal solution.

The fact that the solution set is bounded is a direct consequence of the fact that all optimal solutions give a penalty of the same height,  $v_{min} = \|\mathbf{x}_{opt}\|_1 < \infty$ . Given any two optimal solutions,  $\mathbf{x}_{opt}^1$  and  $\mathbf{x}_{opt}^2$ , the distance between them satisfies

$$\|\mathbf{x}_{opt}^1 - \mathbf{x}_{opt}^2\|_1 \leq \|\mathbf{x}_{opt}^1\|_1 + \|\mathbf{x}_{opt}^2\|_1 = 2v_{min},$$

implying that all solutions are nearby.

In order to show the second property, let us assume that an optimal solution  $\mathbf{x}_{opt}$  has been found for  $(P_1)$ , with  $k > n$  non-zeros. Clearly, the  $k$  columns combined linearly by  $\mathbf{x}_{opt}$  are linearly-dependent, and thus there exists a non-trivial vector  $\mathbf{h}$  that combines these columns to zero (i.e., the support of  $\mathbf{h}$  is contained within the support of  $\mathbf{x}_{opt}$ ),  $\mathbf{Ah} = \mathbf{0}$ .

We consider the vector  $\mathbf{x} = \mathbf{x}_{opt} + \epsilon\mathbf{h}$ , for very small values of  $\epsilon$  that guarantee that no entry in the new vector changes its sign. Any value satisfying  $|\epsilon| \leq \min_i |x_{opt}^i|/|h^i|$  is suitable. First, it is clear that this vector is guaranteed to satisfy the linear constraint  $\mathbf{Ax} = \mathbf{0}$ , and as such, it is a feasible solution to  $(P_1)$ . Furthermore, since  $\mathbf{x}_{opt}$  is assumed to be optimal, we must assume that

$$\forall |\epsilon| \leq \min_i \frac{|x_{opt}^i|}{|h^i|}, \quad \|\mathbf{x}\|_1 = \|\mathbf{x}_{opt} + \epsilon\mathbf{h}\| \geq \|\mathbf{x}_{opt}\|_1.$$

We argue that the above inequality is in fact an equality. The above relationship applies to both positive and negative values of  $\epsilon$  in a region where the  $\ell_1$  function is continuous

and differentiable (because all the vectors  $\mathbf{x}_{opt} + \epsilon\mathbf{h}$  have the same sign pattern). Thus, as claimed, the only way this could be true is if the above inequality is satisfied as an equality. This implies that the addition/subtraction of  $\mathbf{h}$  under those circumstances does not change the  $\ell_1$ -length of the solution. As an example, if  $\mathbf{x}_{opt}$  has only positive entries, then the entries of  $\mathbf{h}$  should be both positive and negative and sum to zero. More generally, we should require that  $\mathbf{h}^T \text{sign}(\mathbf{x}_{opt}) = 0$ .

Our next step is to tune  $\epsilon$  in such a way that one entry in  $\mathbf{x}_{opt}$  is nulled, while keeping the solution's  $\ell_1$  length. We choose the index  $i$  that gives the minimum ratio  $|x_{opt}^i|/|h^i|$ , and we pick  $\epsilon = -x_{opt}^i/h^i$ . In the resulting vector  $\mathbf{x}_{opt} + \epsilon\mathbf{h}$ , the  $i$ -th entry is nulled, while all the others keep their sign, and we also have that  $\|\mathbf{x}_{opt} + \epsilon\mathbf{h}\|_1 = \|\mathbf{x}_{opt}\|_1$ . This way we got a new optimal solution with  $k - 1$  non-zeros at the most (because it may be that more than one entry have been simultaneously nulled). This process can be repeated until  $k = n$ . Below this, it is possible to null entries only if the columns in  $\mathbf{A}$  are linearly-dependent, and this would rarely be the case.  $\square$

From the above we have learned that the  $\ell_1$ -norm has a tendency to prefer sparse solutions. The property we have proven is well known and considered as the fundamental property of linear programming – a tendency towards basic (sparse) solutions. However, as we shall see later on, while we are very much interested in sparsity, getting  $n$  non-zeros would be considered as far too dense for our needs, and higher sparsity would be sought.

### 6.1.5 Conversion of $(P_1)$ to Linear Programming

Suppose that in  $(P_1)$ , the unknown  $\mathbf{x}$  is replaced by  $\mathbf{x} = \mathbf{u} - \mathbf{v}$ , where  $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$  are both non-negative vectors such that  $\mathbf{u}$  takes all the positive entries in  $\mathbf{x}$ , other entries null, and  $\mathbf{v}$  does the same for the negative entries in  $\mathbf{x}$ . With this replacement, and by denoting the concatenated vector  $\mathbf{z} = (\mathbf{u}^T, \mathbf{v}^T)^T \in \mathbb{R}^{2n}$ , it is easy to see that  $\|\mathbf{x}\|_1 = \mathbf{1}^T(\mathbf{u} + \mathbf{v}) = \mathbf{1}^T\mathbf{z}$  and  $\mathbf{Ax} = \mathbf{A}(\mathbf{u} - \mathbf{v}) = (\mathbf{A}, -\mathbf{A})\mathbf{z}$ . Thus, the  $(P_1)$  optimization problem posed in Equation (6.11) can be re-written as

$$\min_{\mathbf{z}} \mathbf{1}^T \mathbf{z}, \quad s.t. \quad \mathbf{b} = (\mathbf{A}, -\mathbf{A})\mathbf{z} \text{ and } \mathbf{z} \geq \mathbf{0}. \quad (6.12)$$

In this way, rather than minimizing an  $\ell_1$ -norm problem, we face a new problem having a classical Linear-Programming (LP) structure. For the two problems  $((P_1)$  and the LP) to be equivalent, we must show that our assumption about the decomposition of  $\mathbf{x}$  to positive and negative entries is satisfied, and

**命题107.** *The solutions to (6.12) cannot be such that  $\mathbf{u}^T \mathbf{v} \neq 0$  (i.e., the supports of  $\mathbf{u}$  and  $\mathbf{v}$  overlap).*

*Proof.* This is easily verified by observing that if in a given optimal solution of (6.12) the  $k$ -th entry in both  $\mathbf{u}$  and  $\mathbf{v}$  is non-zero (and positive, due to the last constraint), then these two coefficients multiply the same column in  $\mathbf{A}$  with opposing signs. Without loss of generality, if we assume  $u_k > v_k > 0$ , then by replacing these two entries by  $u'_k = u_k - v_k$  and  $v'_k = 0$  we satisfy both the positivity and the linear constraints while also reducing the penalty by  $(u_k + v_k) - (u_k - v_k) = 2v_k > 0$ , contradicting the optimality of the initial solution. Thus, the supports of  $\mathbf{u}, \mathbf{v}$  do not overlap, and LP is indeed equivalent to  $(P_1)$ .  $\square$

### 6.1.6 Promoting Sparse Solutions

As we move from  $\ell_2$  regularization towards  $\ell_1$ , we promote sparser solutions. Using this rationale, we can consider  $\ell_p$ -“norms” with  $p < 1$ . One has to be careful as such  $\ell_p$  are no longer formal norms for  $p < 1$ , as the triangle inequality is no longer satisfied. Nevertheless, we shall use the term norm for these functions as well, keeping in mind this reservation.

Do these norms lead to sparser solutions? In order to get a feeling for the behavior of these norms, we consider the following problem: Assume that  $\mathbf{x}$  is known to have unit length in  $\ell_p$ . Let us find the shortest such vector in  $\ell_q$  for  $q < p$ . Putting this as an optimization problem, this reads

$$\min_{\mathbf{x}} \|\mathbf{x}\|_q^q, \quad s.t. \quad \|\mathbf{x}\|_p^p = 1. \quad (6.13)$$

We shall assume that  $\mathbf{x}$  has  $a$  non-zeros (we shall further assume that these values are all positive, since their sign does not affect the analysis that follows) as the leading entries in the vector, and the rest are zeros. We define the Lagrangian function as

$$\mathcal{L}(\mathbf{x}) = \|\mathbf{x}\|_q^q + \lambda(\|\mathbf{x}\|_p^p - 1) = -\lambda + \sum_i^a (|x_i|^q + \lambda|x_i|^p). \quad (6.14)$$

This function is separable, handling every entry in  $\mathbf{x}$  independently of the others. The optimal solution is given by  $x_k^{p-q} = Const$  for all  $k$ , implying that all the nonzero entries are supposed to be the same. The constraint  $\|\mathbf{x}\|_p^p = 1$  leads to  $x_k = a^{-1/p}$ , and the  $\ell_q$ -norm for this solution is given by  $\|\mathbf{x}\|_q^q = a^{1-q/p}$ . Since  $q < p$ , this means that the shortest  $\ell_q$ -norm is obtained for  $a = 1$ , having only one non-zero entry in  $\mathbf{x}$ .

The above result states that for any pair of  $\ell_p$ - and  $\ell_q$ -norms with  $q < p$ , a unit-length  $\ell_p$ -norm vector becomes the shortest in  $\ell_q$  when it is the sparsest possible. A geometrical interpretation of the analysis given above is the following: The unit  $\ell_p$ -ball surface in  $\mathbb{R}^m$  represents the feasible set of the problem posed in (6.13). We blow an  $\ell_q$  “balloon” in the

same space and search for the first intersection of it with the  $\ell_p$ -ball. The result we got implies that this intersection takes place along the axes, where all the entries apart from one are zeros. This is demonstrated in Figure 6.1.

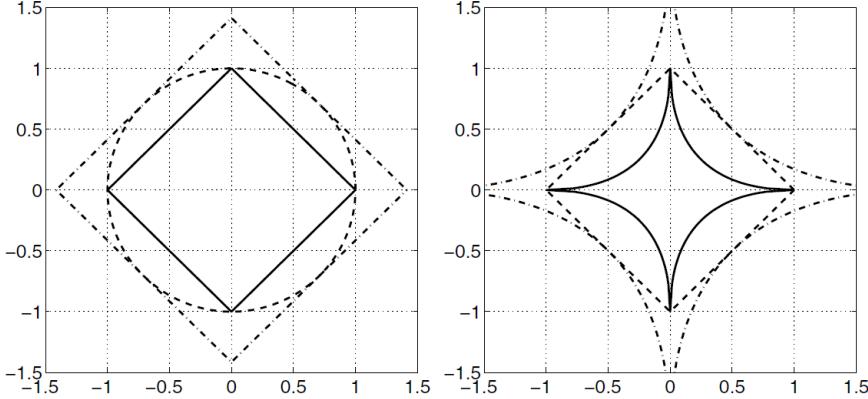


图 6.1: Demonstrating the fact that a unit-length  $\ell_p$ -norm vector (dashed) becomes shortest in  $\ell_q(p > q)$  (solid) when it is the sparsest possible. On the left this is demonstrated for  $p = 2$  and  $q = 1$ , and the right shows it for  $p = 1$  and  $q = 0.5$ . The dash-dotted curves in both graphs show the reverse claim – maximizing the  $\ell_q$  length leads to the most non-sparse outcome.

Another way to illustrate the tendency of  $\ell_p$ -norm to drive results to become sparse is the following: Consider the problem we have been looking at,

$$(P_p) : \quad \min_{\mathbf{x}} \|\mathbf{x}\|_p^p, \quad s.t. \quad \mathbf{b} = \mathbf{A}\mathbf{x}. \quad (6.15)$$

The linear set of equations forming the constraint define a feasible set of solutions that are on an affine subspace (a subspace shifted by a constant vector). This shift can be any possible solution of this system,  $\mathbf{x}_0$ . A linear combination of  $\mathbf{x}_0$  and any vector from the null-space of  $\mathbf{A}$  forms a feasible solution as well. Geometrically, this set appears as a hyperplane of dimension  $\mathbb{R}^{m-n}$  embedded in the  $\mathbb{R}^m$  space.

It is within this space that we seek the solution to the problem posed as  $(P_p)$ . Geometrically speaking, solving  $(P_p)$  is done by “blowing” again an  $\ell_p$  balloon centered around the origin, and stopping its inflation when it first touches the feasible set. The question is what characterizes such an intersection point? Figure 6.2 presents a simple demonstration of this process for a tilted hyperplane (serving as the constraint-set) and several  $p$  values: 2, 1.5, 1, and 0.7. One can see that norms with  $p \leq 1$  tend to give that the intersection point is on the ball-corners, which take place on the axes. This implies that 2 of the 3 coordinates are zeros, which is the tendency to sparsity we are referring

to. As opposed to this,  $\ell_2$  and even  $\ell_{1.5}$  give intersection points which are not sparse, with three non-zero coordinates.

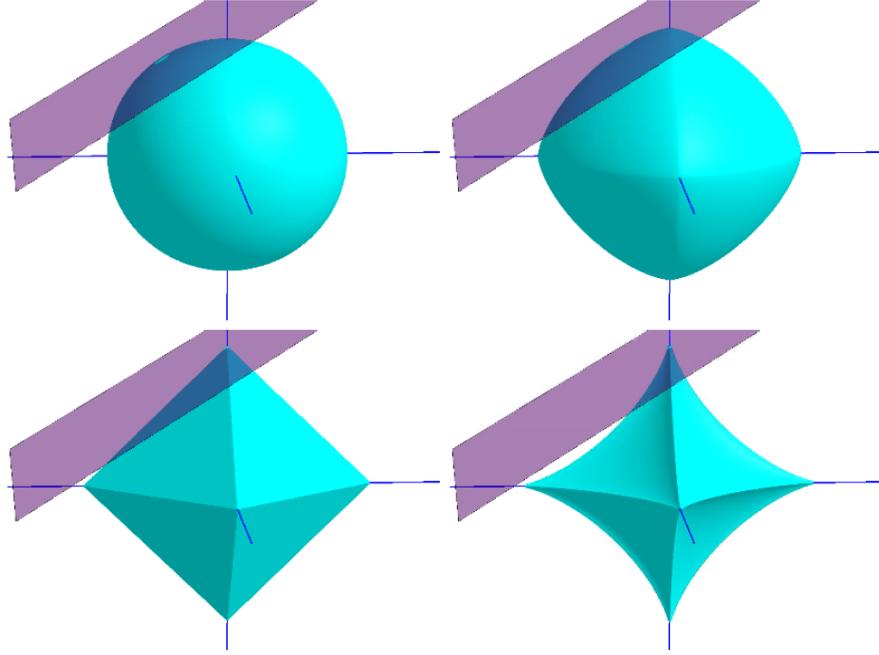


图 6.2: The intersection between the  $\ell_p$ -ball and the set  $\mathbf{Ax} = \mathbf{b}$  defines the solution of  $(P_p)$ . This intersection is demonstrated here in 3D for  $p = 2$  (top left),  $p = 1.5$  (top right),  $p = 1$  (bottom left), and  $p = 0.7$  (bottom right). When  $p \leq 1$ , the intersection takes place at a corner of the ball, leading to a sparse solution.

Put more generally, an intersection of an affine subspace and an  $\ell_p$  ball is expected to take place on the axes for  $p \leq 1$ , and thus lead to a sparse solution. Even the  $\ell_1$ -ball enjoys this property, and in fact, one has to be highly unlucky with the angle of the affine subspace to avoid a sparse outcome.

Another similar measure promoting sparsity is the weak  $\ell_p$ -norm. Denoting by  $N(\epsilon, \mathbf{x})$  the number of entries in  $\mathbf{x}$  exceeding  $\epsilon$ , we define the weak  $\ell_p$ -norm as

$$\|\mathbf{x}\|_{w\ell_p}^p = \sup_{\epsilon > 0} N(\epsilon, \mathbf{x}) \cdot \epsilon^p. \quad (6.16)$$

Here, just as before,  $0 < p \leq 1$  is the interesting range of  $p$ , giving very powerful sparsity measure. The weak  $\ell_p$  norm is a popular measure of sparsity used by the mathematical analysis community. The usual  $\ell_p$ -norm is almost equivalent, and in fact, the two become the same for a vector with uniform spread of its energy among its non-zero entries. The plain  $\ell_p$  is easier to handle and thus more popular in most cases.

It would seem very natural, based on our discussion above, to attempt to solve a problem of the form

$$(P_p) : \quad \min \| \mathbf{x} \|^p_p, \quad s.t. \quad \mathbf{A} \mathbf{x} = \mathbf{b}, \quad (6.17)$$

for example, with  $p = 2/3$ ,  $p = 1/2$ , or even smaller  $p$ , tending to zero. Unfortunately, each choice  $0 < p < 1$  leads to a non-convex optimization problem, and this raises some difficulties, as we have mentioned above. Nevertheless, from an engineering point of view, if sparsity is a desired property, and we know that  $\ell_p$  serves it well, this problem can and should be used, despite its weaknesses.

While all the discussion above focuses on the  $\ell_p$ -norm, there are other functions of  $\mathbf{x}$  that promote sparsity just as well. In fact, any function  $J(\mathbf{x}) = \sum_i \rho(x_i)$  with  $\rho(x)$  being symmetric, monotonically non-decreasing, and with a monotonic nonincreasing derivative for  $x \geq 0$  will serve the same purpose of promoting sparsity. As classic examples of this family, we mention  $\rho(x) = 1 - \exp(-|x|)$ ,  $\rho(x) = \log(1 + |x|)$ , and  $\rho(x) = |x|/(1 + |x|)$ .

### 6.1.7 The $\ell_0$ -Norm and Implications

The extreme among all the sparsifying norms is the case of  $p \rightarrow 0$ . We denote the  $\ell_0$ -norm as

$$\| \mathbf{x} \|_0 = \lim_{p \rightarrow 0} \| \mathbf{x} \|^p_p = \lim_{p \rightarrow 0} \sum_{i=1}^m |x_i|^p = \#\{i | x_i \neq 0\}. \quad (6.18)$$

This is a very simple and intuitive measure of sparsity of a vector  $\mathbf{x}$ , counting the number of nonzero entries in it. In order to see this counting behavior, Figure 6.3 presents the scalar weight function  $|x|^p$  – the core of the norm computation – for various values of  $p$ . It shows that as  $p$  goes to zero, this curve becomes an indicator function, being 0 for  $x = 0$  and 1 for every other value. Thus, summation of such measures on all the entries of  $\mathbf{x}$  becomes a count of the non-zeros in this vector, as claimed.

The term  $\ell_0$ -norm is misleading, as this function does not satisfy all the axiomatic requirements of a norm. More specifically, if we consider the  $\ell_0$  as a continuation of the  $\ell_p$ -norm for  $p \rightarrow 0$ , then for checking its behavior we need to take its 0-th root, and this is impossible. Alternatively, we can simply refer to the function  $\| \mathbf{x} \|_0$  as a candidate function for a norm. While this function satisfies the triangle inequality,  $\| \mathbf{u} + \mathbf{v} \|_0 \leq \| \mathbf{u} \|_0 + \| \mathbf{v} \|_0$ , the homogeneity property is not met: for  $t \neq 0$ ,  $\| t\mathbf{u} \|_0 = \| \mathbf{u} \|_0 \neq t\| \mathbf{u} \|_0$ . This loss of sensitivity to scale will return to haunt us from time to time.

We should note that the  $\ell_0$  norm, while providing a very simple and easily grasped notion of sparsity, is not necessarily the right notion for empirical work. A vector of real data would rarely be representable by a vector of coefficients containing many zeros.

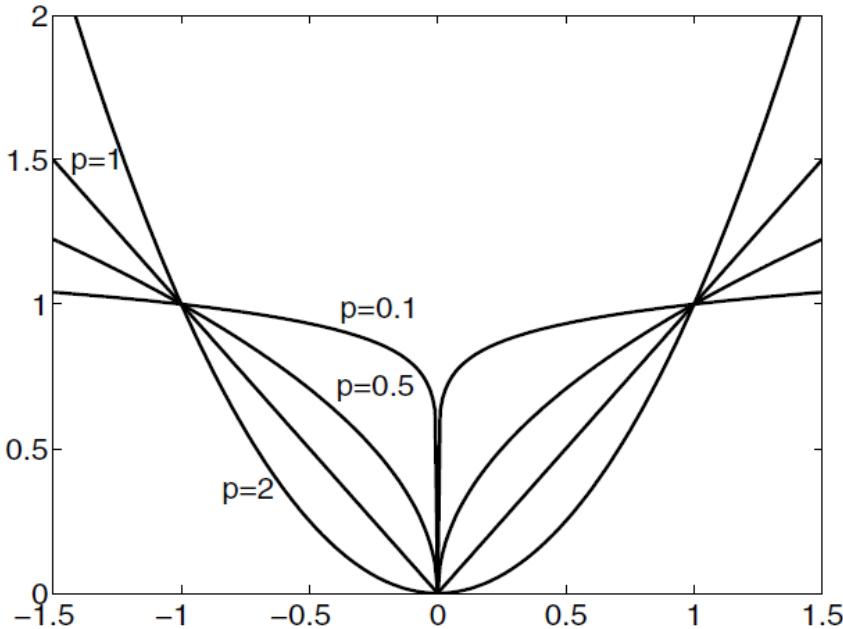


图 6.3: The behavior of  $|x|^p$  for various values of  $p$ . As  $p$  tends to zero,  $|x|^p$  approaches the indicator function, which is 0 for  $x = 0$  and 1 elsewhere.

A more relaxed and forgiving notion of sparsity can and should be built on the notion of approximately representing a vector using a small number of nonzeros; this can be quantified by the weak- $\ell_p$  and the usual  $\ell_p$ -norms described above. Nevertheless, we proceed the discussion here with the assumption that  $\ell_0$  is the measure of interest.

### 6.1.8 The $(P_0)$ Problem – Our Main Interest

Consider the problem  $(P_0)$  obtained from the general prescription  $(P_J)$  with the choice  $J(\mathbf{x}) = J_0(\mathbf{x}) \equiv \|\mathbf{x}\|_0$ ; explicitly:

$$(P_0) : \quad \min \|\mathbf{x}\|_0, \quad s.t. \quad \mathbf{Ax} = \mathbf{b}, \quad (6.19)$$

Sparsity optimization (6.19) looks superficially like the minimum  $\ell_2$ -norm problem  $(P_2)$ , but the notational similarity masks some startling differences. The solution to  $(P_2)$  is always unique, and is readily available through now-standard tools from computational linear algebra.  $(P_0)$  has probably been considered as a possible goal from time to time for many years, but it seems initially to pose many conceptual challenges that have inhibited its widespread study and application. These are rooted in the discrete and discontinuous nature of the  $\ell_0$  norm; the standard convex analysis ideas which underpin the analysis

of  $(P_2)$  do not apply. Many of the most basic questions about  $(P_0)$  seem immune to immediate insight:

1. Can uniqueness of a solution be claimed? Under what conditions?
2. If a candidate solution is available, can we perform a simple test to verify that the solution is actually the global minimizer of  $(P_0)$ ?

Perhaps in some instances, with very special matrices  $\mathbf{A}$  and left hand sides  $\mathbf{b}$ , ways to answer such questions are apparent, but for general classes of problem instances  $(\mathbf{A}, \mathbf{b})$ , such insights initially seem unlikely. Beyond conceptual issues of uniqueness and verification of solutions, one is easily overwhelmed by the apparent difficulty of solving  $(P_0)$ . This is a classical problem of combinatorial search; one sweeps exhaustively through all possible sparse subsets, generating corresponding subsystems  $\mathbf{b} = \mathbf{A}_S \mathbf{x}_S$  where  $\mathbf{A}_S$  denotes the matrix having  $|S|$  columns chosen from those columns of  $\mathbf{A}$  with indices in  $S$ ; and checking if  $\mathbf{b} = \mathbf{A}_S \mathbf{x}_S$  can be solved.

Let us illustrate this complexity by the following simple example: Assume that  $\mathbf{A}$  is of size  $500 \times 2000$  ( $n = 500, m = 2000$ ), and suppose that we know that the sparsest solution to  $(P_0)$  has  $|S| = 20$  non-zeros. We desire to find the proper set of  $|S|$  columns. Thus, we exhaustively sweep through all  $C_m^{|S|} \approx 3.9E + 47$  such options, and per each test the small linear system  $\mathbf{b} = \mathbf{A}_S \mathbf{x}_S$  for equality. Assume that each individual test of such a system requires  $1E - 9$  seconds. A simple calculation reveals that it will take more than  $1.2E + 31$  years(!!!) to conclude these series of tests.

The complexity of exhaustive search is exponential in  $m$ , and indeed, it has been proven that  $(P_0)$  is, in general, NP-Hard. Thus, a mandatory and crucial set of questions arise: Can  $(P_0)$  be efficiently solved by some other means? Can approximate solutions be accepted? How accurate can those be? What kind of approximations will work? These are the questions we aim to answer in this section.

### 6.1.9 The Signal Processing Perspective

We know today that finding sparse solutions to underdetermined linear systems may become better-behaved and may be a much more practically relevant notion than we might have supposed just a few years ago. In parallel with this development, another insight has been developing in signal and image processing, where it has been found that many media types (still imagery, video, acoustic) can be sparsely represented using transform-domain methods, and in fact many important tasks dealing with such media can fruitfully be viewed as finding sparse solutions to underdetermined systems of linear

equations. Many readers will be familiar with the media encoding standard JPEG and its successor, JPEG2000. Both standards are based on the notion of transform encoding that leads to a sparse representation.

We can thus make a simple connection between the above algebraic discussion and the signal representation problem formalized as follows. Let  $\mathbf{b}$  denote the vector of signal/image values to be represented. Let  $\mathbf{A}$  be the matrix whose columns are the elements of the different bases to be used in the representation. The problem we have posed in Equation (6.1) aims to provide a single representation among the many possible ones for  $\mathbf{b}$ . When using the  $\ell_2$ -norm measure, the outcome is a linear operator  $\mathbf{A}^+$  that multiplies  $\mathbf{b}$  in order to compute  $\mathbf{x}$  – this is the well-known Frame approach towards redundant representations, which applies linear operations both for the forward transform (from  $\mathbf{b}$  to  $\mathbf{x}$ ) and its inverse (from  $\mathbf{x}$  to  $\mathbf{b}$ ).

On the other extreme, the problem  $(P_0)$  posed in Equation (6.19) offers literally the sparsest representation of the signal content. This way, while the inverse transform is linear, the forward one is highly non-linear, and quite complex in general. The appeal in such a transform is in the compact representation it provides. This option motivates a closer study of the mathematical problems mentioned above. Much more on this connection will be given in the second part of this book, where we explore ways to harness the understanding of sparse solutions of linear systems of equations to applications in signal and image processing.

## 6.2 Uniqueness and Uncertainty

We return to the basic problem  $(P_0)$ , which is at the core of our discussion,

$$(P_0) : \min_{\mathbf{x}} \|\mathbf{x}\|_0, \quad s.t. \quad \mathbf{b} = \mathbf{Ax}.$$

While we shall refer hereafter to this problem as our main goal, we stress that we are quite aware of its two major shortcomings in leading to any practical tool:

1. The equality requirement  $\mathbf{b} = \mathbf{Ax}$  is too strict, as there are small chances for any vector  $\mathbf{b}$  to be represented by a few columns from  $\mathbf{A}$ . A better requirement would be one that allows for small deviation.
2. The sparsity measure is too sensitive to very small entries in  $\mathbf{x}$ , and a better measure would adopt a more forgiving approach towards such small entries.

Both these considerations will be included in later analysis, but for this to succeed, we must start with the stylized version of the problem as indeed posed by  $(P_0)$ .

For the underdetermined linear system of equations,  $\mathbf{Ax} = \mathbf{b}$  (a full-rank matrix  $\mathbf{A} \in \mathbb{R}^{n \times m}$  with  $n < m$ ), we pose the following questions:

Q1: When can uniqueness of the sparsest solution be claimed?

Q2: Can a candidate solution be tested to verify its (global) optimality?

This section addresses these questions, and some of their extensions. Rather than answering the above questions directly, we first consider special matrices  $\mathbf{A}$  for which the analysis seems to be easier, and then extend our answers to the general  $\mathbf{A}$ . In doing so, we also follow the path taken by researchers who originally addressed these questions.

### 6.2.1 Treating the Two-Ortho Case

We shall first discuss the  $(P_0)$  problem defined above in a concrete setting: the case where  $\mathbf{A}$  is the concatenation of two orthogonal matrices,  $\Psi$  and  $\Phi$ . As a classic example, we can consider the amalgam of the identity and the Fourier matrices  $\mathbf{A} = (\mathbf{I}, \mathbf{F})$ . In such a setting, the fact that the system  $\mathbf{b} = \mathbf{Ax}$  is underdetermined means, concretely, that there are many ways of representing a given signal  $\mathbf{b}$  as a superposition of spikes (i.e., columns from the identity matrix) and sinusoids (i.e., columns from the Fourier matrix). A sparse solution of such a system is a representation of said signal as a superposition of a few sinusoids and a few spikes. The uniqueness of such a sparse solution seemed surprising when first noticed.

#### 6.2.1.1 An Uncertainty Principle

Before addressing sparse solutions for the linear system  $(\Psi, \Phi)\mathbf{x} = \mathbf{b}$ , we shall consider a (seemingly) different problem, inspired by the setting of classical uncertainty principles. As the reader no doubt knows, the classical uncertainty principle states that two conjugate variables (e.g., position and momentum, or any other pair coupled by the Fourier transform) cannot both be known with arbitrary precision. Turning to its mathematical formulation, it states that any function  $f(x)$  and its Fourier transform  $F(\omega)$  must obey the inequality:

$$\int_{-\infty}^{+\infty} x^2 |f(x)|^2 dx \cdot \int_{-\infty}^{+\infty} \omega^2 |F(\omega)|^2 d\omega \geq \frac{1}{2}, \quad (6.20)$$

where we have assumed that these functions are  $\ell_2$ -normalized,

$$\int_{-\infty}^{+\infty} |f(x)|^2 dx = 1.$$

This claim says that a signal cannot be tightly concentrated both in time and in frequency, and there is a lower bound on the product of the spread in time and the spread in frequency.

A comparable claim in our terminology would be that a signal cannot be sparsely represented both in time and in frequency. We now turn to develop this exact viewpoint, as it will be helpful for understanding some of the discussion that follows. Suppose we have a non-zero vector  $\mathbf{b} \in \mathbb{R}^n$  (a signal, say) and two orthobases  $\Psi$  and  $\Phi$ . Then  $\mathbf{b}$  can be represented either as a linear combination of columns of  $\Psi$  or as a linear combination of columns of  $\Phi$ :

$$\mathbf{b} = \Psi\alpha = \Phi\beta. \quad (6.21)$$

Clearly,  $\alpha$  and  $\beta$  are uniquely defined. In a particularly important case,  $\Psi$  is simply the identity matrix, and  $\Phi$  is the matrix of the Fourier transform. Then  $\alpha$  is the time-domain representation of  $\mathbf{b}$  while  $\beta$  is the frequency-domain representation.

For an arbitrary pair of bases  $\Psi$  and  $\Phi$ , an interesting phenomenon occurs: either  $\alpha$  can be sparse, or  $\beta$  can be sparse, but not both! However, this claim is clearly dependent on the distance between  $\Psi$  and  $\Phi$ , since if the two are the same, we can easily construct  $\mathbf{b}$  to be one of the columns in  $\Psi$  and get the smallest possible cardinality (being 1) in both  $\alpha$  and  $\beta$ . Thus, we turn now to define the proximity between two bases by their *mutual-coherence*.

**定义108.** For an arbitrary pair of orthogonal bases  $\Psi$  and  $\Phi$  that construct the matrix  $\mathbf{A}$ , we define the mutual-coherence  $\mu(\mathbf{A})$  as the maximal inner product between columns from these two bases,

$$\text{proximity}(\Psi, \Phi) = \mu(\mathbf{A}) = \max_{1 \leq i, j \leq n} |\psi_i^T \phi_j|. \quad (6.22)$$

The *mutual-coherence* of such two-ortho matrices satisfies  $1/\sqrt{n} \leq \mu(\mathbf{A}) \leq 1$ , where the lower bound is achievable for certain pairs of orthogonal bases, such as the identity and the Fourier, the identity and the Hadamard, and more. To see that this is indeed the lower bound on the possible coherence, one simply notices that  $\Psi^T \Phi$  is an orthonormal matrix, having the sum of squares of its entries in each column equal to 1. All entries cannot therefore be less than  $1/\sqrt{n}$  since then we would have that the sum of all squared entries is less than 1. Using Definition 108 above, we have the following inequality result:

**定理109.** For an arbitrary pair of orthogonal bases  $\Psi$  and  $\Phi$  with mutual-coherence  $\mu(\mathbf{A})$ , and for an arbitrary non-zero vector  $\mathbf{b} \in \mathbb{R}^n$  with representations  $\alpha$  and  $\beta$  correspondingly, the following inequality holds true:

$$\text{Uncertainty Principle 1: } \|\alpha\|_0 + \|\beta\|_0 \geq \frac{2}{\mu(\mathbf{A})}. \quad (6.23)$$

This result suggests that if the mutual-coherence of two bases is small, then  $\alpha$  and  $\beta$  cannot both be very sparse. For example, if, as above,  $\Psi$  is the identity and  $\Phi$  is the Fourier matrix, then  $\mu((\Psi, \Phi)) = 1/\sqrt{n}$ . It follows that a signal cannot have fewer than  $2\sqrt{n}$  nonzeros in both the time and frequency-domains. In such a case, we also know that this is a tight relationship, since the picket-fence signal with a uniform spacing  $\sqrt{n}$  (assuming it is an integer) is converted by the Fourier transform (due to Poisson formula) to the same signal, thus accumulating  $2\sqrt{n}$  non-zeros. Figure 6.4 shows this signal.

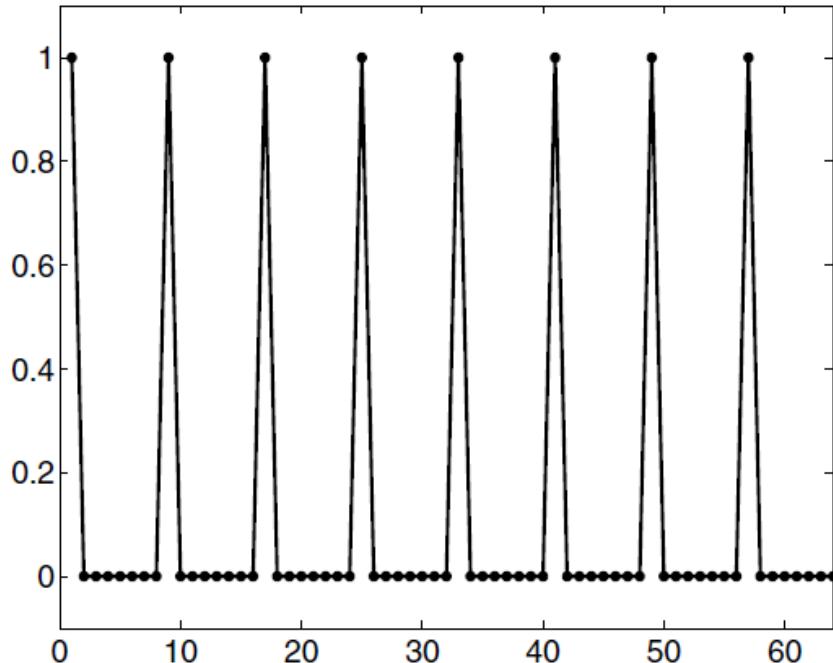


图 6.4: The picket-fence signal for  $n = 64$ . It has 8 uniformly spread non-zeros with equal height. The Discrete-Fourier-Transform (DFT) of this signal looks exactly the same.

Heisenberg's classical uncertainty principle, in the discrete setting, says that, if we view  $\alpha$  and  $\beta$  as probability distributions (by taking the absolute value of the entries and normalizing) then the product of their variances satisfies  $\sigma_\alpha^2 \sigma_\beta^2 \geq \text{const}$ . In contrast, (6.23) gives a lower bound on the sum of the nonzeros, regardless of their locations.

**练习110.** Check that the mutual-coherence of the identity and Fourier matrix is  $1/\sqrt{n}$ .

### 6.2.1.2 Uncertainty of Redundant Solutions

We now make a connection to the uniqueness problem. Consider the problem of finding a solution to  $\mathbf{Ax} = (\Psi, \Phi)\mathbf{x} = \mathbf{b}$  in light of the uncertainty principle (6.23).

Suppose there are two solutions,  $\mathbf{x}_1$  and  $\mathbf{x}_2$ , for the underlying linear system, and that one is very sparse. We will see that the other cannot be very sparse as well. Necessarily, the difference  $\mathbf{e} = \mathbf{x}_1 - \mathbf{x}_2$  must be in the null-space of  $\mathbf{A}$ . Partition  $\mathbf{e}$  into sub-vectors  $\mathbf{e}_\Psi$  and  $\mathbf{e}_\Phi$  of the first  $n$  entries and last  $n$  entries of  $\mathbf{e}$ , respectively. We have

$$\Psi \mathbf{e}_\Psi = -\Phi \mathbf{e}_\Phi = \mathbf{y} \neq \mathbf{0}. \quad (6.24)$$

The vector  $\mathbf{y}$  is nonzero because  $\mathbf{e}$  is nonzero, and both  $\Psi$  and  $\Phi$  are nonsingular. Now invoke (6.23):

$$\|\mathbf{e}\|_0 = \|\mathbf{e}_\Psi\|_0 + \|\mathbf{e}_\Phi\|_0 \geq \frac{2}{\mu(\mathbf{A})}. \quad (6.25)$$

Since  $\mathbf{e} = \mathbf{x}_1 - \mathbf{x}_2$ , we have

$$(\text{Uncertainty Principle 2}): \|\mathbf{x}_1\|_0 + \|\mathbf{x}_2\|_0 \geq \|\mathbf{e}\|_0 \geq \frac{2}{\mu(\mathbf{A})}. \quad (6.26)$$

Here we have used the triangle inequality for the  $\ell_0$  norm,  $\|\mathbf{x}_1\|_0 + \|\mathbf{x}_2\|_0 \geq \|\mathbf{x}_1 - \mathbf{x}_2\|_0$ , which is trivially verified, by counting the non-zeros of the two vectors and considering no overlap of supports (which leads to equality) and an overlap (which gives this inequality). To summarize, we have proven the following result:

**定理111.** *Any two distinct solutions  $\mathbf{x}_1$  and  $\mathbf{x}_2$  of the linear system  $(\Psi, \Phi)\mathbf{x} = \mathbf{b}$  cannot both be very sparse, as governed by the uncertainty principle (6.26).*

We refer to this result as an uncertainty of redundant solutions, as we discuss here solutions to the underdetermined system.

#### 6.2.1.3 From Uncertainty to Uniqueness

A direct consequence of inequality (6.26) is a uniqueness result:

**定理112.** *If a candidate solution for  $(\Psi, \Phi)\mathbf{x} = \mathbf{b}$  has fewer than  $1/\mu(\mathbf{A})$  non-zeros, then it is necessarily the sparsest one possible, and any other solution must be “denser.”*

This seemingly simple claim is wonderful and powerful. At least for the special case discussed here,  $\mathbf{A} = (\Psi, \Phi)$ , we have a complete answer for the two questions we have posed at the beginning of this chapter. Namely, we can certainly claim uniqueness for sparse enough solutions, and once such a sufficiently sparse solution is given to us, we can immediately claim its global optimality. Notice that in general non-convex optimization problems, a given solution can at best be verified as being locally optimal, and here we have the ability to verify its optimality globally.

Since the discussion so far concentrated on the two-ortho case, it is now time to dare and address general matrices  $\mathbf{A}$ , using similar treatment. Clearly, though, an uncertainty result of the kind posed in Theorem 109 would be impossible to imitate, and we will have to bypass it somehow.

### 6.2.2 Uniqueness Analysis for the General Case

#### 6.2.2.1 Uniqueness via the Spark

A key property that is crucial for the study of uniqueness is the *spark* of the matrix  $\mathbf{A}$ , a term coined and defined by Donoho and Elad in 2003. The *spark* is a way of characterizing the null-space of a matrix  $\mathbf{A}$  using the  $\ell_0$ -norm. We start with the following definition:

**定义113.** *The spark of a given matrix  $\mathbf{A}$  is the smallest number of columns from  $\mathbf{A}$  that are linearly dependent.*

Obviously,

$$\text{spark}(\mathbf{A}) = \min_{\mathbf{x} \neq \mathbf{0}, \mathbf{Ax} = \mathbf{0}} \|\mathbf{x}\|_0.$$

Recall that the rank of a matrix is defined as the *largest* number of columns from  $\mathbf{A}$  that are linearly *independent*. Clearly, there is a resemblance between these two definitions – replace largest with smallest, and independent by dependent, and you return to the definition of the spark. Nevertheless, the spark of a matrix is far more difficult to obtain, compared to its rank, as it calls for a combinatorial search over all possible subsets of columns from  $\mathbf{A}$ .

The importance of this property of matrices for the study of the uniqueness of sparse solutions has been unraveled already by Rao and Gorodnitsky in 1998. Interestingly, this property has previously appeared in the literature of psychometrics (termed *Kruskal rank*), used in the context of studying uniqueness of tensor decomposition. The spark is also related to established notions in matroid theory; formally it is precisely the girth of the linear matroid defined by  $\mathbf{A}$ , i.e., the length of the shortest cycle in that matroid. Finally, if we consider the same definition where the arithmetic underlying the matrix product is performed *not* over the fields of real or complex numbers but instead over the ring of integers mod  $q$ , the same quantity arises in coding theory, where it allows to compute the minimum distance of a code. The resemblance between all these concepts is striking and instructive.

The spark gives a simple criterion for uniqueness of sparse solutions. By definition, the vectors in the null-space of the matrix  $\mathbf{A}$  must satisfy  $\|\mathbf{x}\|_0 \geq \text{spark}(\mathbf{A})$ , since these

vectors combine linearly columns from  $\mathbf{A}$  to give the zero vector, and at least spark such columns are necessary by definition. Using the spark we obtain the following result:

**定理114.** (*Uniqueness - Spark*): *If a system of linear equations  $\mathbf{Ax} = \mathbf{b}$  has a solution  $\mathbf{x}$  obeying  $\|\mathbf{x}\|_0 < \text{spark}(\mathbf{A})/2$ , this solution is necessarily the sparsest possible.*

Again, this result is very elementary and yet quite surprising, bearing in mind that  $(P_0)$  is a highly complicated optimization task of combinatorial flavor. In general combinatorial optimization problems, when considering a proposed solution, one hopes only to check local optimality – i.e., that no simple modification gives a better result. Here, we find that simply checking the solution sparsity, and comparing that with the spark, lets us check global optimality.

Clearly, the value of spark can be very informative, and large values of spark are evidently very useful. How large can spark be? By definition, spark must be in the range<sup>1</sup>  $2 \leq \text{spark}(\mathbf{A}) \leq n + 1$ . For example, if  $\mathbf{A}$  comprises random independent and identically distributed entries (say Gaussian), then with probability 1 we have  $\text{spark}(\mathbf{A}) = n + 1$ , implying that no  $n$  columns are linearly-dependent. The same spark is obtained for Vandermonde matrices constructed from distinct  $m$  scalars. In these cases, uniqueness is ensured for every solution with  $n/2$  or fewer non-zero entries. Similarly, the spark of the two-ortho identity-Fourier pair<sup>2</sup> is  $2\sqrt{n}$ , using Poisson formula – a concatenation of two picket-fence signals, each with  $\sqrt{n}$  peaks evenly spread is the sparsest possible vector in the null-space of the matrix  $(\mathbf{I}, \mathbf{F})$ .

**练习115.** *What is the spark of the following matrix?*

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$$

**练习116.** *Prove that if  $\text{spark}(\mathbf{A}) \neq +\infty$ , then  $\text{spark}(\mathbf{A}) \leq \text{rank}(\mathbf{A}) + 1$ .*

### 6.2.2.2 Uniqueness via the Mutual-Coherence

The spark is at least as difficult to evaluate as solving  $(P_0)$ . Thus, simpler ways to guarantee uniqueness are of interest. A very simple way exploits the mutual-coherence

<sup>1</sup>The spark can be as low as 1 if there exists a zero-column in  $\mathbf{A}$ , but we do not consider such a case as relevant to our analysis.

<sup>2</sup>If  $n$  is prime, the spark of the identity-Fourier pair becomes  $n + 1$ , as the picket-fence signals are no longer relevant.

of the matrix  $\mathbf{A}$ , defined by generalizing the definition given for the two-ortho case. In the two-ortho case, computation of the Gram matrix  $\mathbf{A}^T \mathbf{A}$  leads to

$$\mathbf{A}^T \mathbf{A} = \begin{pmatrix} \mathbf{I} & \boldsymbol{\Psi}^T \boldsymbol{\Phi} \\ \boldsymbol{\Phi}^T \boldsymbol{\Psi} & \mathbf{I} \end{pmatrix}. \quad (6.27)$$

Thus, the above-defined mutual-coherence for this case is obtained as the maximal off-diagonal entry (in absolute value) in this Gram matrix. Similarly, we propose a generalization of this definition, as follows:

**定义117.** *The mutual-coherence of a given matrix  $\mathbf{A}$  is the largest absolute normalized inner product between different columns from  $\mathbf{A}$ . Denoting the  $k$ -th column in  $\mathbf{A}$  by  $\mathbf{a}_k$ , the mutual-coherence is given by*

$$\mu(\mathbf{A}) = \max_{1 \leq i, j \leq m, i \neq j} \frac{|\mathbf{a}_i^T \mathbf{a}_j|}{\|\mathbf{a}_i\|_2 \cdot \|\mathbf{a}_j\|_2} \quad (6.28)$$

The mutual-coherence is a way to characterize the dependence between columns of the matrix  $\mathbf{A}$ . For a unitary matrix, columns are pairwise orthogonal, and so the mutual-coherence is zero. For general matrices with more columns than rows,  $m > n$ ,  $\mu$  is necessarily strictly positive, and we desire the smallest possible value so as to get as close as possible to the behavior exhibited by unitary matrices.

We have seen that for structured two-ortho matrices  $\mathbf{A} = (\boldsymbol{\Psi}, \boldsymbol{\Phi})$  the mutual-coherence satisfies  $1/\sqrt{n} \leq \mu(\mathbf{A}) \leq 1$ . When considering random orthogonal matrices of size  $n \times m$ , the work in Donoho and Huo has shown that they tend to be incoherent, implying that  $\mu(\mathbf{A}_{n,m})$  is typically proportional to  $\sqrt{\log(nm)/n}$  for  $n \rightarrow +\infty$ . It has been shown that for full-rank matrices of size  $n \times m$  the mutual-coherence is bounded from below by

$$\mu \geq \sqrt{\frac{m-n}{n(m-1)}}. \quad (6.29)$$

Equality is obtained for a family of matrices named *Grassmannian Frames*. The set of columns in such matrices are called *equiangular lines*. Indeed, this family of matrices has  $\text{spark}(\mathbf{A}) = n + 1$ , the highest value possible. Numerical construction of such matrices has been addressed by Tropp et al. using an iterative projection onto (sometimes not so) convex sets. We will return to this topic towards the end of this section.

We also mention work by Calderbank in quantum information theory, constructing error-correcting-codes using a collection of orthogonal bases with minimal coherence, obtaining similar bounds on the mutual-coherence for amalgams of orthogonal bases. Also related to this line of activity is the more recent contribution by Sochen, Gurevitz, and Hadani, on constructions of signal sets such that their shifts exhibit low coherence.

Mutual-coherence is relatively easy to compute, and as such, it allows us to lower-bound the spark, which is often hard to compute.

**引理118.** *For any matrix  $\mathbf{A} \in \mathbb{R}^{n \times m}$ , the following relationship holds:*

$$\text{spark}(\mathbf{A}) \geq 1 + \frac{1}{\mu(\mathbf{A})}. \quad (6.30)$$

Proof. Apply Gershgorin disk theorem on the submatrix of  $\mathbf{A}$  that corresponds to  $\text{spark}(\mathbf{A})$  and observe that the columns of  $\mathbf{A}$  have been normalized.

We have the following analog of the previous uniqueness theorems, and this time based on the mutual-coherence.

**定理119.** *(Uniqueness – Mutual-Coherence): If a system of linear equations  $\mathbf{Ax} = \mathbf{b}$  has a solution  $\mathbf{x}$  obeying  $\|\mathbf{x}\|_0 < \frac{1}{2}(1 + 1/\mu(\mathbf{A}))$ , this solution is necessarily the sparsest possible.*

Compare Theorems 114 and 119. They are parallel in form, but with different assumptions. In general, Theorem 114, which uses spark, is sharp and is far more powerful than Theorem 119, which uses the coherence and so only a lower bound on spark. The coherence can never be smaller than  $1/\sqrt{n}$ , and therefore, the cardinality bound of Theorem 119 is never larger than  $\sqrt{n}/2$ . However, the spark can easily be as large as  $n$ , and Theorem 114 then gives a bound as large as  $n/2$ .

In fact, for the special matrix  $\mathbf{A} = (\Psi, \Phi)$  we have also obtained such a rule. Interestingly, the lower bound for the general case becomes  $(1 + 1/\mu(\mathbf{A}))/2$  while the special two-ortho case gave a stronger (i.e., higher) lower bound. The general case bound is nearly a factor of 2 weaker than (6.26), because (6.26) uses the special structure  $\mathbf{A} = (\Psi, \Phi)$ .

**练习120.** *What is the mutual-coherence of the following matrix?*

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$$

### 6.2.2.3 Uniqueness via the Babel Function

In the proof of Lemma 118 we considered minors of size  $p \times p$  extracted from the Gram matrix of the normalized matrix  $\tilde{\mathbf{A}}$ . The positive-definiteness of all such minors imply that every  $p$  columns are linearly-independent. However, for simplification of the analysis we bounded all the off-diagonal entries of  $\mathbf{G}$  by a single value  $\mu(\mathbf{A})$ , and thus, lost robustness to possibly few extreme entries in this matrix.

Since we need to check whether the sum of the  $p - 1$  off-diagonal entries in every row in these minors is less than 1 (for the Gershgorin property to hold true), we can define the following Babel function, following Tropp:

**定义121.** For a given matrix  $\tilde{\mathbf{A}}$  with normalized columns, we consider a subset  $\Lambda$  of  $p$  columns from  $\tilde{\mathbf{A}}$ , and compute the sum of the absolute values of their inner product with a column outside this set. Maximizing over both the set  $\Lambda$  and the outside column  $j$  we obtain the Babel function:

$$\mu_1(p) = \max_{\Lambda, |\Lambda|=p} \max_{j \notin \Lambda} \sum_{i \in \Lambda} |\tilde{\mathbf{a}}_i^T \tilde{\mathbf{a}}_j|. \quad (6.31)$$

Clearly, for  $p = 1$ , we get that  $\mu_1(1) = \mu(\mathbf{A})$ . For  $p = 2$  the above implies that we need to sweep through all possible triplets, considering two as those belonging to  $\Lambda$ , and the third as the external vector to compute inner products with. This definition implies that this function is monotonically non-decreasing, and the slower its growth the better the analysis it leads to, compared to the use of the cruder coherence measure.

It might seem that computing this function for large  $p$  becomes exponential and thus prohibitive, but this is in fact not true. By computing  $|\mathbf{G}| = |\tilde{\mathbf{A}}^T \tilde{\mathbf{A}}|$ , and sorting every row in descending order, we obtain the matrix  $\mathbf{G}_s$ . The first entry in every row is 1, being the main diagonal entry, and thus should be disregarded. Computing the sum of the  $p$  leading entries in every row (second and beyond), we get for every  $j$  in the above definition the set of worst  $\Lambda$ 's, and among them we should choose the maximal one, thus

$$\mu_1(p) = \max_{1 \leq j \leq m} \sum_{i=2}^{p+1} |G_s(j, i)|. \quad (6.32)$$

We note that for every  $p$  we have that  $\mu_1(p) \leq p \cdot \mu(\mathbf{A})$ , and the two become equal for Grassmannian matrices, where all the off-diagonal entries in the Gram matrix are of the same magnitude.

How can we use the Babel function for better assessing the uniqueness? It is clear that if  $\mu_1(p) < 1$ , we deduce that all  $p + 1$  sets are linearly-independent. Thus, a lower-bound on the spark could be

$$\text{spark}(\mathbf{A}) \geq \min_{1 \leq p \leq n} \{p | \mu_1(p - 1) \geq 1\}. \quad (6.33)$$

The uncertainty and uniqueness properties follow immediately.

**练习122.** What is the value of  $\mu_1(2)$  of the following matrix?

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$$

**练习123.** Prove (6.33).

**练习124.** 写出用Babel函数刻画的不确定性和唯一性定理并证明。

#### 6.2.2.4 Upper-Bounding the Spark

The spark is impossible to evaluate in general, as it is even harder than solving  $(P_0)$ . This is because its evaluation requires a sweep through all possible groups of columns from  $\mathbf{A}$  with varying cardinalities, seeking for a linearly-dependent subset of columns. This is a combinatorial search of exponential complexity with  $m$ .

While this difficulty explains the necessity to replace the spark with the mutual-coherence, the price paid in loss of tightness of the uniqueness result may be considered as too dear to be permitted. This motivates an alternative method for approximating the spark, and this time using an upper bound. Such a bound implies that we cannot guarantee uniqueness based on the obtained value, but it would give us a rough evaluation of the region of uniqueness.

In order to develop the upper-bound, we redefine the spark as the outcome of  $m$  optimization problems  $(P_0^i)$  for  $i = 1, 2, \dots, m$  of the form:

$$(P_0^i) : \quad \mathbf{x}_{opt}^i = \underset{\mathbf{x}}{\operatorname{argmin}} \|\mathbf{x}\|_0, \quad s.t. \quad \mathbf{Ax} = \mathbf{0} \text{ and } x_i = 1. \quad (6.34)$$

Each of these problems assumes that the sparsest vector in the null-space of  $\mathbf{A}$  uses the  $i$ -th entry. By solving this sequence of  $(P_0)$ -like problems, the sparsest result among  $\{\mathbf{x}_{opt}^i\}_{i=1}^m$  gives the spark,

$$\operatorname{spark}(\mathbf{A}) = \min_{1 \leq i \leq m} \|\mathbf{x}_{opt}^i\|_0. \quad (6.35)$$

Since the set of problems  $(P_0^i)$  is too complex, we define an alternative set of problems, replacing the  $\ell_0$  with the  $\ell_1$ -norm,

$$(P_1^i) : \quad \mathbf{z}_{opt}^i = \underset{\mathbf{x}}{\operatorname{argmin}} \|\mathbf{x}\|_1, \quad s.t. \quad \mathbf{Ax} = \mathbf{0} \text{ and } x_i = 1. \quad (6.36)$$

As we have seen in Section 6.1, these problems have a linear programming structure, they are convex, and solvable in reasonable time. Furthermore, it is clear that for every  $i$ , we have that  $\|\mathbf{x}_{opt}^i\|_0 \leq \|\mathbf{z}_{opt}^i\|_0$ , since  $\|\mathbf{x}_{opt}^i\|_0$  is the sparsest solution possible for this problem by definition, and thus we have the promised bound

$$\operatorname{spark}(\mathbf{A}) \leq \min_{1 \leq i \leq m} \|\mathbf{z}_{opt}^i\|_0. \quad (6.37)$$

Numerical experiments show that this bound tends to be quite tight, and close to the true spark.

### 6.2.3 Constructing Grassmannian Matrices

A Grassmannian (real) matrix  $\mathbf{A}$  of size  $n \times m$  with  $m \geq n$  is a matrix with normalized columns such that its Gram matrix  $\mathbf{G} = \mathbf{A}^T \mathbf{A}$  satisfies

$$\forall k \neq j, |G_{k,j}| = \sqrt{\frac{m-n}{n(m-1)}}. \quad (6.38)$$

As we have stated above, this is the smallest possible mutual-coherence possible. Such matrices do not necessarily exist, and in particular, they are possible only if

$$m < \min(n(n+1)/2, (m-n)(m-n+1)/2).$$

Grassmannian matrices are special in the sense that the angle between each and every pair of columns in it is the same, and it is also the smallest possible. Thus, construction of such matrices has a strong connection with packing of vectors/subspaces in the  $\mathbb{R}^n$ -space. While the case  $m = n$  leads trivially to unitary matrices that are easy to construct, constructing a general Grassmannian matrix is very hard. A numerical algorithm for this task was proposed by Tropp et al. and we bring it here as an illustration of such design procedures. We should add that while image processing finds little interest in such constructions, they find important application in channel coding, wireless communication, and more.

The key idea in the proposed algorithm is to iterate between projections onto the requirements such matrix should satisfy. Starting with an arbitrary matrix  $\mathbf{A}$ , these projections should refer to the following requirements:

1. The columns in  $\mathbf{A}$  are  $\ell_2$ -normalized: This can be forced by normalizing every column.
2. Property (6.38): We should compute  $\mathbf{G} = \mathbf{A}^T \mathbf{A}$ , detect the off-diagonal entries that are above some threshold, and decrease them. Similarly, since too small values are also not permitted in such Gram matrices, one might also add an increase to such off-diagonal entries.
3. The rank of  $\mathbf{G}$  should not exceed  $n$ : Since the above modifications cause  $\mathbf{G}$  to become a full-rank one, an SVD operation and truncation of the singular-values beyond the first  $n$  ones brings the new Gram matrix to the proper rank.

This process can and should be repeated many times. There is no guarantee for convergence, or arrival at a Grassmannian matrix, but tests show that one can get closer to such matrices with this numerical scheme. A Matlab code that follows these guidelines is given in Algorithm 1.

---

**Algorithm 1** Matlab code for building a Grassmannian matrix.

```

D=randn(N,L); % initialization
D=D*diag(1./sqrt(diag(D'*D))); % normalize columns
G=D'*D;% compute the Gram matrix
mu=sqrt((L-N)/N/(L-1));
for k=1:1:Iter
    % shrink the high inner products
    gg=sort(abs(G(:)));
    pos=find(abs(G(:))>gg(round(dd1*(L*L-L))) & abs(G(:))<1;
    G(pos)=G(pos)*dd2;
    % reduce the rank back to N
    [U,S,V]=svd(G);
    S(N+1:end,1+N:end)=0;
    G=U*S*V';
    % Normalize the columns
    G=diag(1./sqrt(diag(G)))*G*diag(1./sqrt(diag(G)));
    % Show status
    gg=sort(abs(G(:)));
    pos=find(abs(G(:))>gg(round(dd1*(L*L-L))) & abs(G(:))<1;
    disp([k,mu,mean(abs(G(pos))),max(abs(G(pos)))]);
end
[U,S,V]=svd(G);
D=sqrt(S(1:N,1:N))*U(:,1:N);

```

---

#### 6.2.4 Summary

We have now given answers to the questions posed at the start of this chapter. We have seen that any sufficiently sparse solution is guaranteed to be unique among all possible solutions. Consequently, any sufficiently sparse solution is necessarily the global optimizer of  $(P_0)$ . These results show that searching for a sparse solution can lead to a well-posed question with interesting properties. We now turn to discuss practical methods for obtaining solutions for  $(P_0)$ .

### 6.3 Pursuit Algorithms – Practice

It is now time to consider reliable and efficient methods for solving  $(P_0)$ , as a straightforward approach seems hopeless. We now discuss methods which, it seems, have

no hope of working – but which, under specific conditions, will work. Looking at the problem  $(P_0)$ ,

$$(P_0) : \min_{\mathbf{x}} \|\mathbf{x}\|_0, \quad s.t. \quad \mathbf{b} = \mathbf{A}\mathbf{x},$$

one observes that the unknown  $\mathbf{x}$  is composed of two effective parts to be found – the support of the solution, and the non-zero values over this support. Thus, one way to attack the numerical solution of  $(P_0)$  is to focus on the support, with the understanding that once found, the non-zero values of  $\mathbf{x}$  are easily detected by plain Least-Squares. As the support is discrete in nature, algorithms that seek it are discrete as well. This line of reasoning leads to the family of greedy algorithms that will be presented hereafter.

An alternative view of  $(P_0)$  can disregard the support, and consider the unknown as a vector  $\mathbf{x} \in \mathbb{R}^m$  over the continuum. Smoothing the penalty function  $\|\mathbf{x}\|_0$ , one can adopt a continuous optimization point of view for solving  $(P_0)$ . The relaxation methods that are presented later in this chapter adopt this view, by smoothing the  $\ell_0$ -norm in various forms, and handling the revised problem as a smooth optimization.

### 6.3.1 Greedy Algorithms

#### 6.3.1.1 The Core Idea

Suppose that the matrix  $\mathbf{A}$  has  $\text{spark}(\mathbf{A}) > 2$ , and the optimization problem  $(P_0)$  has value  $\text{val}(P_0) = 1$  (at the optimal solution), so  $\mathbf{b}$  is a scalar multiple of some column of the matrix  $\mathbf{A}$ , and this solution is known to be unique. We can identify this column by applying  $m$  tests – one per column of  $\mathbf{A}$ . The  $j$ -th test can be done by minimizing  $\epsilon(j) = \|\mathbf{a}_j z_j - \mathbf{b}\|_2$ , leading to  $z_j^* = \mathbf{a}_j^T \mathbf{b} / \|\mathbf{a}_j\|_2^2$ . Plugged back into the error expression, the error becomes

$$\epsilon(j) = \min_{z_j} \|\mathbf{a}_j z_j - \mathbf{b}\|_2^2 = \left\| \frac{\mathbf{a}_j^T \mathbf{b}}{\|\mathbf{a}_j\|_2^2} \mathbf{a}_j - \mathbf{b} \right\|_2^2 \quad (6.39)$$

$$= \|\mathbf{b}\|_2^2 - \frac{(\mathbf{a}_j^T \mathbf{b})^2}{\|\mathbf{a}_j\|_2^2}. \quad (6.40)$$

If this error is zero, we have found the proper solution. Thus the test to be done is simply  $\|\mathbf{a}_j\|_2^2 \|\mathbf{b}\|_2^2 = (\mathbf{a}_j^T \mathbf{b})^2$  (which is equivalent to the statement that the Cauchy-Schwartz inequality should be satisfied with equality), indicating that  $\mathbf{b}$  and  $\mathbf{a}_j$  are parallel. This procedure requires order  $O(mn)$  flops, which may be considered reasonable.

Proceeding with the same rationale, suppose that  $\mathbf{A}$  has  $\text{spark}(\mathbf{A}) > 2k_0$ , and the optimization problem is known to have value  $\text{val}(P_0) = k_0$ . Then  $\mathbf{b}$  is a linear combination of at most  $k_0$  columns of  $\mathbf{A}$ . Generalizing the previous solution, one might think to

enumerate all  $C_m^{k_0} = O(m^{k_0})$  subsets of  $k_0$  columns from  $\mathbf{A}$ , and test each. Enumeration takes  $O(m^{k_0} n k_0^2)$  flops, which seems prohibitively slow in many settings.

A greedy strategy abandons exhaustive search in favor of a series of locally optimal single-term updates. Starting from  $\mathbf{x}^0 = \mathbf{0}$  it iteratively constructs a  $k$ -term approximant  $\mathbf{x}_k$  by maintaining a set of active columns – initially empty – and, at each stage, expanding that set by one additional column. The column chosen at each stage maximally reduces the residual  $\ell_2$  error in approximating  $\mathbf{b}$  from the currently active columns. After constructing an approximant including the new column, the residual  $\ell_2$  error is evaluated; if it now falls below a specified threshold, the algorithm terminates.

### 6.3.1.2 The Orthogonal-Matching-Pursuit

Algorithm 2 presents a formal description of this strategy, and its associated notation. This procedure is known in the literature of signal processing by the name Orthogonal-Matching-Pursuit (OMP), but is very well known (and was used much earlier) by other names in other fields – see below. Note that the Sweep stage gives error values of the following form, which is very similar to what we have seen above in Equation (6.39),

$$\epsilon(j) = \min_{z_j} \|\mathbf{a}_j z_j - \mathbf{r}^{k-1}\|_2^2 = \left\| \frac{\mathbf{a}_j^T \mathbf{r}^{k-1}}{\|\mathbf{a}_j\|_2^2} \mathbf{a}_j - \mathbf{r}^{k-1} \right\|_2^2 \quad (6.41)$$

$$= \|\mathbf{r}^{k-1}\|_2^2 - \frac{(\mathbf{a}_j^T \mathbf{r}^{k-1})^2}{\|\mathbf{a}_j\|_2^2}. \quad (6.42)$$

Thus, the quest for the smallest error is actually equivalent to the quest for the largest (in absolute value) inner product between the residual  $\mathbf{r}^{k-1}$  and the normalized vectors of the matrix  $\mathbf{A}$ .

In the Update Provisional Solution stage, we minimize the term  $\|\mathbf{Ax} - \mathbf{b}\|_2^2$  with respect to  $\mathbf{x}$ , such that its support is  $S^k$ . We denote  $\mathbf{A}_{S^k}$  as a matrix of size  $n \times |S^k|$  that contains the columns from  $\mathbf{A}$  that belong to this support. Thus, the problem to be solved is a minimization of  $\|\mathbf{A}_{S^k} \mathbf{x}_{S^k} - \mathbf{b}\|_2^2$ , where  $\mathbf{x}_{S^k}$  is the non-zero portion of the vector  $\mathbf{x}$ . The solution is given by zeroing the derivative of this quadratic form,

$$\mathbf{A}_{S^k}^T (\mathbf{A}_{S^k} \mathbf{x}_{S^k} - \mathbf{b}) = \mathbf{A}_{S^k}^T \mathbf{r}^k = \mathbf{0}. \quad (6.43)$$

Here we have used the formula of the residual in the  $k$ -th iteration,  $\mathbf{r}^k = \mathbf{b} - \mathbf{Ax}^k = \mathbf{b} - \mathbf{A}_{S^k} \mathbf{x}_{S^k}$ . The above relation suggests that the columns in  $\mathbf{A}$  that are part of the support  $S^k$  are necessarily orthogonal to the residual  $\mathbf{r}^k$ . This in turn implies that in the next iteration (and afterwards), these columns will not be chosen again for the support. This

---

**Algorithm 2** Orthogonal-Matching-Pursuit – a greedy algorithm for approximating the solution of  $(P_0)$ .

---

**Task:** Approximate the solution of  $(P_0)$ :  $\min_{\mathbf{x}} \|\mathbf{x}\|_0$ , s.t.  $\mathbf{Ax} = \mathbf{b}$ .

**Parameters:** We are given the matrix  $\mathbf{A}$ , the vector  $\mathbf{b}$ , and the error threshold  $\epsilon_0$ .

**Initialization:** Initialize  $k = 0$ , and set  $\mathbf{x}^0 = \mathbf{0}$ ,  $\mathbf{r}^0 = \mathbf{b} - \mathbf{Ax}^0 = \mathbf{b}$ ,  $S^0 = \text{Support}(\mathbf{x}^0) = \emptyset$ .

**Main Iteration:** Increase  $k$  by 1 and perform the following steps:

**Sweep:** Compute the errors  $\epsilon(j) = \min_{z_j} \|\mathbf{a}_j z_j - \mathbf{r}^{k-1}\|_2^2$  for all  $j$  using the optimal choice  $z_j^* = \mathbf{a}_j^T \mathbf{r}^{k-1} / \|\mathbf{a}_j\|_2^2$ .

**Update Support:** Find a minimizer,  $j_0$  of  $\epsilon(j) : \forall j \notin S^{k-1}, \epsilon(j_0) \leq \epsilon(j)$ , and update  $S^k = S^{k-1} \cup \{j_0\}$ .

**Update Provisional Solution:** Compute  $\mathbf{x}_k$ , the minimizer of  $\|\mathbf{Ax} - \mathbf{b}\|_2^2$  subject to  $\text{Support}(\mathbf{x}) = S^k$ .

**Update Residual:** Compute  $\mathbf{r}^k = \mathbf{b} - \mathbf{Ax}^k$ .

**Stopping Rule:** If  $\|\mathbf{r}^k\|_2 < \epsilon_0$ , stop. Otherwise, apply another iteration.

**Output:** The proposed solution is  $\mathbf{x}^k$  obtained after  $k$  iterations.

---

orthogonalization is the reason for the name of the algorithm – Orthogonal-Matching-Pursuit.

A more complex and somewhat better behaving variant of the above algorithm can be suggested, where each of the tests in the sweep stage is done as full Least-Squares (LS), considering all the accumulated columns and the candidate one together, and solving for all the coefficients at once. As in the OMP,  $m - |S^{k-1}|$  LS steps are performed at the  $k$ -th step. This method is known as LS-OMP. The LS steps can be made efficient using the formula

$$\begin{pmatrix} \mathbf{M} & \mathbf{b} \\ \mathbf{b}^T & c \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{M}^{-1} + p\mathbf{M}^{-1}\mathbf{b}\mathbf{b}^T\mathbf{M}^{-1} & -p\mathbf{M}^{-1}\mathbf{b} \\ -p\mathbf{b}^T\mathbf{M}^{-1} & p \end{pmatrix}, \quad (6.44)$$

where  $p = 1/(c - \mathbf{b}^T \mathbf{M}^{-1} \mathbf{b})$ . Denoting the sub-matrix that gathers the  $k - 1$  chosen columns from  $\mathbf{A}$  as  $\mathbf{A}_s$ , the LS problem we should solve is

$$\min_{\mathbf{x}_s, z} \left\| (\mathbf{A}_s \mathbf{a}_i) \begin{pmatrix} \mathbf{x}_s \\ z \end{pmatrix} - \mathbf{b} \right\|_2^2. \quad (6.45)$$

The solution is given by zeroing the derivative of the above expression

$$\begin{pmatrix} \mathbf{A}_s^T \\ \mathbf{a}_i^T \end{pmatrix} (\mathbf{A}_s \mathbf{a}_i) \begin{pmatrix} \mathbf{x}_s \\ z \end{pmatrix} - \begin{pmatrix} \mathbf{A}_s^T \\ \mathbf{a}_i^T \end{pmatrix} \mathbf{b} = \mathbf{0}. \quad (6.46)$$

Thus, based on the formula given in Equation (6.44), if we have stored the matrix  $(\mathbf{A}_S^T \mathbf{A}_S)^{-1}$ , then the above formula becomes easy, and leads to an update of this matrix for  $k$  columns. Once the solution is found, the residual error should be evaluated, and the column  $\mathbf{a}_i$  that leads to the smallest error should be chosen. Observe that if  $\mathbf{a}_i$  is orthogonal to the columns of  $\mathbf{A}_S$ , the above matrix to be inverted becomes block-diagonal, and the result would be  $\mathbf{x}_S = \mathbf{A}_S^+ \mathbf{b}$  and  $z = \mathbf{a}_i^T \mathbf{b} / \|\mathbf{a}_i\|_2^2$ . This means that the coefficients of the previous solution remain intact, and a new non-zero is introduced with a coefficient  $z$ .

There are numerical shortcuts for the evaluation of the residuals, but we shall not explore this matter here. Note that the above recursive approach towards the Least-Squares task in the LS-OMP is also relevant for speeding up the OMP, implying that the Update Provisional Solution stage can be made far more efficient.

If the delivered approximation has  $k_0$  non-zeros, the LS-OMP and OMP methods require  $O(k_0 mn)$  flops in general; this can be dramatically better than the exhaustive search, which requires  $O(nm^{k_0} k_0^2)$  flops. Thus, the single-term-at-a-time strategy can be much more efficient than exhaustive search – if it works! The strategy can fail badly, i.e., there are explicit examples constructed by Vladimir Temlyakov where a simple  $k$ -term representation is possible, but this approach yields an  $n$ -term (i.e., dense) representation. In general, all that can be said is that among single-term-at-a-time strategies, the approximation error is always reduced by as much as possible, given the starting approximation and the single-term-at-a-time constraint. This property has earned this algorithm the name “Greedy algorithm” in approximation theory.

### 6.3.1.3 Other Greedy Methods

Many variants on the above algorithm are available, offering improvements either in accuracy and/or in complexity. This family of greedy algorithms is well known and extensively used, and in fact, these algorithms have been re-invented in various fields. In the setting of statistical modeling, greedy stepwise Least-Squares is called forward stepwise regression, and has been widely practiced since at least the 1960s. When used in the signal processing setting this goes by the name of Matching-Pursuit (MP) or Orthogonal-Matching-Pursuit (OMP). Approximation theorists refer to these algorithms as Greedy Algorithms (GA), and consider several variants of them – the Pure (PGA), the Orthogonal (OGA), the Relaxed (RGA), and the Weak Greedy Algorithm (WGA).

The MP algorithm is similar to the OMP, but with an important difference that makes it simpler, and thus less accurate. In the main iteration, after the sweep and the update support stages, rather than solving a Least-Squares for re-evaluating all the

---

**Algorithm 3** Matching-Pursuit for approximating the solution of  $(P_0)$ .

**Task:** Approximate the solution of  $(P_0)$ :  $\min_{\mathbf{x}} \|\mathbf{x}\|_0$ , s.t.  $\mathbf{Ax} = \mathbf{b}$ .

**Parameters:** We are given the matrix  $\mathbf{A}$ , the vector  $\mathbf{b}$ , and the error threshold  $\epsilon_0$ .

**Initialization:** Initialize  $k = 0$ , and set  $\mathbf{x}^0 = \mathbf{0}$ ,  $\mathbf{r}^0 = \mathbf{b} - \mathbf{Ax}^0 = \mathbf{b}$ ,  $S^0 = \text{Support}(\mathbf{x}^0) = \emptyset$ .

**Main Iteration:** Increase  $k$  by 1 and perform the following steps:

**Sweep:** Compute the errors  $\epsilon(j) = \min_{z_j} \|\mathbf{a}_j z_j - \mathbf{r}^{k-1}\|_2^2$  for all  $j$  using the optimal choice  $z_j^* = \mathbf{a}_j^T \mathbf{r}^{k-1} / \|\mathbf{a}_j\|_2^2$ .

**Update Support:** Find a minimizer,  $j_0$  of  $\epsilon(j) : \forall 1 \leq j \leq m, \epsilon(j_0) \leq \epsilon(j)$ , and update  $S^k = S^{k-1} \cup \{j_0\}$ .

**Update Provisional Solution:** Set  $\mathbf{x}_k = \mathbf{x}_{k-1}$ , and update the entry  $x^k(j_0) = x^k(j_0) + z_{j_0}^*$ .

**Update Residual:** Compute  $\mathbf{r}^k = \mathbf{b} - \mathbf{Ax}^k = \mathbf{r}^{k-1} - z_{j_0}^* \mathbf{a}_{j_0}$ .

**Stopping Rule:** If  $\|\mathbf{r}^k\|_2 < \epsilon_0$ , stop. Otherwise, apply another iteration.

**Output:** The proposed solution is  $\mathbf{x}^k$  obtained after  $k$  iterations.

---

coefficients in  $\mathbf{x}$ , the coefficients of the  $S^{k-1}$  original entries remain unchanged, and the new coefficient that refers to the new member  $j_0 \in S^k$  is chosen as  $z_{j_0}^*$ . This algorithm is described in Algorithm 3.

The Weak-MP is a further simplification of the MP algorithm, allowing for a sub-optimal choice of the next element to be added to the support. Embarking from the MP algorithm description in Algorithm 3, the update support is relaxed by choosing any index that is factor  $t$  (in the range  $(0, 1]$ ) away from the optimal choice. This algorithm is described in Algorithm 4.

Let us explain and motivate this method: We have seen the equivalence between the inner products computed,  $|\mathbf{a}_j^T \mathbf{r}^{k-1}|$  (up to a normalization factor), and the corresponding errors  $\epsilon(j)$  among which we seek the minimum. Rather than searching for the largest inner-product value, we settle for the first found that exceeds a  $t$ -weaker threshold. The Cauchy-Schwartz inequality gives

$$\frac{(\mathbf{a}_j^T \mathbf{r}^{k-1})^2}{\|\mathbf{a}_j\|_2^2} \leq \max_{1 \leq j \leq m} \frac{(\mathbf{a}_j^T \mathbf{r}^{k-1})^2}{\|\mathbf{a}_j\|_2^2} \leq \|\mathbf{r}^{k-1}\|_2^2. \quad (6.47)$$

This puts an upper bound on the maximal achievable inner-product. Thus, we can compute  $\|\mathbf{r}^{k-1}\|_2^2$  at the beginning of the *Sweep* stage, and as we search for  $j_0$  that gives the smallest error  $\epsilon(j)$ , we choose the first that gives

$$\frac{(\mathbf{a}_{j_0}^T \mathbf{r}^{k-1})^2}{\|\mathbf{a}_{j_0}\|_2^2} \geq t^2 \cdot \|\mathbf{r}^{k-1}\|_2^2 \geq t^2 \cdot \max_{1 \leq j \leq m} \frac{(\mathbf{a}_j^T \mathbf{r}^{k-1})^2}{\|\mathbf{a}_j\|_2^2}. \quad (6.48)$$

---

**Algorithm 4** Weak-Matching-Pursuit for approximating the solution of  $(P_0)$ .

---

**Task:** Approximate the solution of  $(P_0)$ :  $\min_{\mathbf{x}} \|\mathbf{x}\|_0, \text{ s.t. } \mathbf{Ax} = \mathbf{b}$ .

**Parameters:** We are given the matrix  $\mathbf{A}$ , the vector  $\mathbf{b}$ , the error threshold  $\epsilon_0$ , and the scalar  $0 < t < 1$ .

**Initialization:** Initialize  $k = 0$ , and set  $\mathbf{x}^0 = \mathbf{0}$ ,  $\mathbf{r}^0 = \mathbf{b} - \mathbf{Ax}^0 = \mathbf{b}$ ,  $S^0 = \text{Support}(\mathbf{x}^0) = \emptyset$ .

**Main Iteration:** Increase  $k$  by 1 and perform the following steps:

**Sweep:** Compute the errors  $\epsilon(j) = \min_{z_j} \|\mathbf{a}_j z_j - \mathbf{r}^{k-1}\|_2^2$  for all  $j$  using the optimal choice  $z_j^* = \mathbf{a}_j^T \mathbf{r}^{k-1} / \|\mathbf{a}_j\|_2^2$ . Stop the sweep when  $|\mathbf{a}_j^T \mathbf{r}^{k-1}| / \|\mathbf{a}_j\|_2 \geq t \cdot \|\mathbf{r}^{k-1}\|_2$

**Update Support:** Update  $S^k = S^{k-1} \cup \{j_0\}$ , with  $j_0$  found in the sweep stage.

**Update Provisional Solution:** Set  $\mathbf{x}_k = \mathbf{x}_{k-1}$ , and update the entry  $x^k(j_0) = x^k(j_0) + z_{j_0}^*$ .

**Update Residual:** Compute  $\mathbf{r}^k = \mathbf{b} - \mathbf{Ax}^k = \mathbf{r}^{k-1} - z_{j_0}^* \mathbf{a}_{j_0}$ .

**Stopping Rule:** If  $\|\mathbf{r}^k\|_2 < \epsilon_0$ , stop. Otherwise, apply another iteration.

**Output:** The proposed solution is  $\mathbf{x}^k$  obtained after  $k$  iterations.

---

for a pre-chosen  $t$  in the range  $(0, 1)$ . The above condition can be written alternatively as

$$(\mathbf{a}_{j_0}^T \mathbf{r}^{k-1})^2 \geq t^2 \cdot \|\mathbf{r}^{k-1}\|_2^2 \cdot \|\mathbf{a}_{j_0}\|_2^2. \quad (6.49)$$

This way, the chosen index clearly points to a column that is at the most factor  $t$  away from the possible maximum. Note that it is possible that the complete sweep is performed without any index satisfying the above condition, and then we simply choose the maximum that was found as a by-product of the search. This way, the search could become much faster, with the understanding that the rate-of-decay of the residual has been somewhat compromised.

#### 6.3.1.4 Normalization

All the above greedy algorithms (OMP, MP, and weak-MP) are described for a general matrix  $\mathbf{A}$ , which may have columns that are not of unit  $\ell_2$ -norm. We can normalize the columns by the operation  $\tilde{\mathbf{A}} = \mathbf{AW}$ , using a diagonal matrix  $\mathbf{W}$  containing  $1/\|\mathbf{a}_i\|_2$  on the main-diagonal, and then use  $\tilde{\mathbf{A}}$  in the above algorithms. Will the outcome be different? The answer is stated in the following Theorem.

**定理125.** *The greedy algorithms (OMP, MP, and weak-MP) produce the same solution support  $S^k$  when using either the original matrix  $\mathbf{A}$  or its normalized version  $\tilde{\mathbf{A}}$ .*

It is simpler to work with the normalized matrix, since then plain inner-products are used in all these algorithms for choosing the next support index. Therefore, we shall assume hereafter that these matrices are preceded with a normalization. Note that if the original problem is given with a non-normalized matrix, the normalization does have an effect on the obtained result  $\mathbf{x}^k$ . Thus, if a normalized matrix is used in any of these algorithms, a post de-normalization step of the form  $\mathbf{x}^k = \mathbf{W}\tilde{\mathbf{x}}^k$  must be performed. This is because the algorithm provides a solution that satisfies  $\tilde{\mathbf{A}}\tilde{\mathbf{x}}^k = \mathbf{b}$ , and since  $\tilde{\mathbf{A}} = \mathbf{A}\mathbf{W}$  get that

$$\mathbf{b} = \tilde{\mathbf{A}}\tilde{\mathbf{x}}^k = \mathbf{A}\mathbf{W}\tilde{\mathbf{x}}^k = \mathbf{A}\mathbf{x}^k \Rightarrow \mathbf{x}^k = \mathbf{W}\tilde{\mathbf{x}}^k. \quad (6.50)$$

### 6.3.1.5 Rate of Decay of the Residual in Greedy Methods

In the MP algorithm we have seen (see Algorithm 3) that the residual is updated by the recursive formula<sup>3</sup>

$$\mathbf{r}^k = \mathbf{r}^{k-1} - z_{j_0}^* \mathbf{a}_{j_0} = \mathbf{r}^{k-1} - (\mathbf{a}_{j_0}^T \mathbf{r}^{k-1}) \mathbf{a}_{j_0}, \quad (6.51)$$

where  $j_0$  is chosen such that  $|\mathbf{a}_j^T \mathbf{r}^{k-1}|$  is maximized. Thus, the energy of the residual behaves like

$$\epsilon(j_0) = \|\mathbf{r}^k\|_2^2 = \|\mathbf{r}^{k-1}\|_2^2 - (\mathbf{a}_{j_0}^T \mathbf{r}^{k-1})^2 = \|\mathbf{r}^{k-1}\|_2^2 - \max_{1 \leq j \leq m} (\mathbf{a}_j^T \mathbf{r}^{k-1})^2. \quad (6.52)$$

Let us define the following decay-factor, following the idea developed by Mallat and Zhang (1993):

**定义126.** For the matrix  $\mathbf{A}$  with  $m$  normalized columns  $\{\mathbf{a}_j\}_{j=1}^m$ , and an arbitrary vector  $\mathbf{v}$ , the decay-factor  $\delta(\mathbf{A}, \mathbf{v})$  is defined by

$$\delta(\mathbf{A}, \mathbf{v}) = \max_{1 \leq j \leq m} \frac{|\mathbf{a}_j^T \mathbf{v}|^2}{\|\mathbf{v}\|_2^2}. \quad (6.53)$$

Clearly, by this definition we have that the relation  $\|\mathbf{v}\|_2^2 \cdot \delta(\mathbf{A}, \mathbf{v}) = \max_j |\mathbf{a}_j^T \mathbf{v}|$ . Also evident is the fact that  $0 \leq \delta(\mathbf{A}, \mathbf{v}) \leq 1$  by its definition as a normalized correlation. The maximum is attained for  $\mathbf{v} = \mathbf{a}_j$ . The minimum could be attained only if  $\mathbf{A}$  is rank-deficient. We shall return to this discussion shortly, after using this definition for analyzing the MP error decay rate.

Based on the above definition we now define the following universal decay-factor that minimizes the above over all possible vectors  $\mathbf{v}$ .

---

<sup>3</sup>From now on we shall assume that the columns of  $\mathbf{A}$  are normalized.

**定义127.** For the matrix  $\mathbf{A}$  with  $m$  normalized columns  $\{\mathbf{a}_j\}_{j=1}^m$ , the universal decay-factor  $\delta(\mathbf{A})$  is defined by

$$\delta(\mathbf{A}) = \inf_{\mathbf{v}} \max_{1 \leq j \leq m} \frac{|\mathbf{a}_j^T \mathbf{v}|^2}{\|\mathbf{v}\|_2^2} = \inf_{\mathbf{v}} \delta(\mathbf{A}, \mathbf{v}). \quad (6.54)$$

This definition seeks the vector  $\mathbf{v}$  that leads to the smallest possible inner product with the columns of  $\mathbf{A}$ . As we show next, this implies the weakest decay of the residual. Returning to Equation (6.52) and using  $\delta(\mathbf{A})$ , the decay-factor of the matrix  $\mathbf{A}$ , we obtain

$$\begin{aligned} \|\mathbf{r}^k\|_2^2 &= \|\mathbf{r}^{k-1}\|_2^2 - \max_{1 \leq j \leq m} |\mathbf{a}_j^T \mathbf{r}^{k-1}|^2 \\ &= \|\mathbf{r}^{k-1}\|_2^2 - \max_{1 \leq j \leq m} \frac{|\mathbf{a}_j^T \mathbf{r}^{k-1}|^2}{\|\mathbf{r}^{k-1}\|_2^2} \cdot \|\mathbf{r}^{k-1}\|_2^2 \\ &= \|\mathbf{r}^{k-1}\|_2^2 - \delta(\mathbf{A}, \mathbf{r}^{k-1}) \cdot \|\mathbf{r}^{k-1}\|_2^2 \\ &\leq \|\mathbf{r}^{k-1}\|_2^2 - \delta(\mathbf{A}) \cdot \|\mathbf{r}^{k-1}\|_2^2 \\ &= (1 - \delta(\mathbf{A})) \|\mathbf{r}^{k-1}\|_2^2. \end{aligned} \quad (6.55)$$

Application of this formula recessively leads to

$$\|\mathbf{r}^k\|_2^2 \leq (1 - \delta(\mathbf{A}))^k \|\mathbf{r}^0\|_2^2 = (1 - \delta(\mathbf{A}))^k \|\mathbf{b}\|_2^2, \quad (6.56)$$

establishing an exponential rate-of-convergence bound of the residual. A similar analysis for the OMP reveals that this is also the bound on its residual rate of decay, as the LS update in the OMP reduces the error even more, when compared to the MP. Similarly, the Weak-MP leads to an exponential rate of convergence with  $t\delta(\mathbf{A})$  replacing  $\delta(\mathbf{A})$ , leading to a weaker decay-factor.

For all the above to show convergence, we must verify that  $\delta(\mathbf{A})$  is strictly positive and not zero. Since the  $\mathbf{A}$  we consider is a full-rank matrix with more columns than rows, its columns span the entire  $\mathbb{R}^n$  space. Thus, no vector  $\mathbf{v}$  could be orthogonal to all these columns, and thus  $\delta(\mathbf{A}) > 0$ .

As an interesting example, if  $\mathbf{A} = \mathbf{I}$  (or any orthogonal matrix for that matter), it is easy to verify that  $\delta(\mathbf{A}) = 1/n$ . Interestingly, the explanation for this is very similar to the one explaining the smallest mutual-coherence for two-ortho matrices. Take the identity matrix, concatenate the vector  $\mathbf{v}$ , and then compute the Gram matrix. The decay-factor  $\delta(\mathbf{A})$  is simply the largest off-diagonal of this Gram matrix, and this is the square of the largest entry in  $\mathbf{v}$ . Thus, the (normalized) vector that gives the smallest possible inner product with the columns of  $\mathbf{I}$  is  $\mathbf{v}_{opt} = 1/\sqrt{n}$ .

An alternative expression can be used for the definition of  $\delta(\mathbf{A})$ , by normalizing the

---

**Algorithm 5** The Thresholding Algorithm – an algorithm for approximating the solution of  $(P_0)$ .

---

**Task:** Approximate the solution of  $(P_0)$ :  $\min_{\mathbf{x}} \|\mathbf{x}\|_0, \text{ s.t. } \mathbf{Ax} = \mathbf{b}$ .

**Parameters:** We are given the matrix  $\mathbf{A}$ , the vector  $\mathbf{b}$ , and the number of atoms desired,  $k$ .

**Quality Evaluation:** Compute the errors  $\epsilon(j) = \min_{z_j} \|\mathbf{a}_j z_j - \mathbf{b}\|_2^2$  for all  $j$  using the optimal choice  $z_j^* = \mathbf{a}_j^T \mathbf{b} / \|\mathbf{a}_j\|_2^2$ .

**Update Support:** Find the set of indices  $S$  of cardinality  $k$  that contains the smallest errors:  $\forall j \in S, \epsilon(j) \leq \min_{i \notin S} \epsilon(i)$ .

**Update Provisional Solution:** Compute  $\mathbf{x}^k$ , the minimizer of  $\|\mathbf{Ax} - \mathbf{b}\|_2^2$  subject to  $\text{Support}(\mathbf{x}) = S$ .

**Output:** The proposed solution is  $\mathbf{x}^k$ , the minimizer of  $\|\mathbf{Ax} - \mathbf{b}\|_2^2$  subject to  $\text{Support}(\mathbf{x}) = S$ .

---

columns of  $\mathbf{A}$  as we did before, obtaining  $\tilde{\mathbf{A}}$ . Using this leads to

$$\begin{aligned} \delta(\tilde{\mathbf{A}}) &= \inf_{\mathbf{v}} \max_{1 \leq j \leq m} \frac{|\mathbf{a}_j^T \mathbf{v}|^2}{\|\mathbf{v}\|_2^2} \\ &= \inf_{\mathbf{v}} \frac{1}{\|\mathbf{v}\|_2^2} \max_{1 \leq j \leq m} |\mathbf{a}_j^T \mathbf{v}|^2 \\ &= \inf_{\mathbf{v}} \frac{\|\tilde{\mathbf{A}}^T \mathbf{v}\|_\infty^2}{\|\mathbf{v}\|_2^2}. \end{aligned} \tag{6.57}$$

If the  $\ell_\infty$  would have been replaced by  $\ell_2$ , this would have become the smallest eigenvalue of the matrix  $\tilde{\mathbf{A}} \tilde{\mathbf{A}}^T$ , which is also known to be strictly positive since this matrix is positive-definite.

### 6.3.1.6 Thresholding Algorithm

We conclude the discussion on the greedy algorithms by introducing another method, which is far simpler than all the previous ones, and slightly different in the way the greediness is practiced. A simplification of the OMP algorithm can be proposed, where the decision made about the support of the solution is based on the first projection alone. The idea is to choose the  $k$  largest inner products as the desired support. This method, called the thresholding algorithm, is depicted in Algorithm 5.

The error  $\epsilon(j)$  according to which the support is chosen is the same one used in earlier algorithms, and is given by

$$\epsilon(j) = \min_{z_j} \|\mathbf{a}_j z_j - \mathbf{b}\|_2^2 = \left\| \frac{\mathbf{a}_j \mathbf{b}}{\|\mathbf{a}_j\|_2^2} \mathbf{a}_j - \mathbf{b} \right\|_2^2 = \|\mathbf{b}\|_2^2 - \frac{(\mathbf{a}_j \mathbf{b})^2}{\|\mathbf{a}_j\|_2^2}, \tag{6.58}$$

Thus, the same result is obtained by first normalizing the columns in the matrix  $\mathbf{A}$  and then using the simpler formula,

$$\epsilon(j) = \|\mathbf{b}\|_2^2 - (\mathbf{a}_j^\top \mathbf{b})^2. \quad (6.59)$$

This also implies that the search for the  $k$  elements of the support amounts to a simple sort of the entries of the vector  $|\mathbf{A}^T \mathbf{b}|$ . Clearly, this algorithm is far simpler than the greedy techniques presented above.

We note that in the above algorithm description, we assume that  $k$ , the number of required non-zeros, is known. Alternatively, we can increase  $k$  until the error  $\|\mathbf{Ax}^k - \mathbf{b}\|_2$  reaches a pre-specified value  $\epsilon_0$ .

### 6.3.2 Convex Relaxation Techniques

#### 6.3.2.1 Relaxation of the $\ell_0$ -Norm

As already mentioned, a second way to render  $(P_0)$  more tractable is to relax the (highly discontinuous)  $\ell_0$ -norm, replacing it by a continuous or even smooth approximation. Examples of such relaxation options include replacing it with  $\ell_p$  norms for some  $p \in (0, 1]$  or even by smooth functions such as  $\sum_j \log(1 + \alpha x_j^2)$ ,  $\sum_j x_j^2 / (\alpha + x_j^2)$ , or  $\sum_j [1 - \exp(-\alpha x_j^2)]$ .

An interesting example of this family is the FOcal Underdetermined System Solver (FOCUSS) algorithm by Gorodnitsky and Rao. This method uses a method called Iterative-Reweighted-Least-Squares (IRLS) to represent  $\ell_p$  (for some fixed  $p \in (0, 1]$ ) as a weighted  $\ell_2$ -norm. In an iterative algorithm, given a current approximate solution  $\mathbf{x}_{k-1}$ , set  $\mathbf{X}_{k-1} = \text{diag}(|\mathbf{x}_{k-1}|^q)$ . Assuming, for a moment, that this matrix is invertible, the term  $\|\mathbf{X}_{k-1}^{-1} \mathbf{x}\|_2^2$  is equal to  $\|\mathbf{x}\|_{2-2q}^{2-2q}$ , since every entry in  $\mathbf{x}$  is raised to the power  $2 - 2q$  and then they are summed. Thus, by choosing  $q = 1 - p/2$ , this expression imitates the  $\ell_p$ -norm,  $\|\mathbf{x}\|_p^p$ .

Instead of using the inverse of  $\mathbf{X}_{k-1}$  as done above, we use the pseudo-inverse,  $\|\mathbf{X}_{k-1}^+ \mathbf{x}\|_2^2$ , defined as plain inversion for non-zero entries  $x_i$  and zeros elsewhere. Note that with this change the above interpretation of the  $\ell_p$ -norm is not ruined, as those entries contribute zeros to the summation of the norm anyhow. Based on this, if we attempt to solve the problem

$$(M_k) : \min_{\mathbf{x}} \|\mathbf{X}_{k-1}^+ \mathbf{x}\|_2^2, \quad s.t. \quad \mathbf{b} = \mathbf{Ax}, \quad (6.60)$$

this is an imitation of the  $(P_p)$  problem around the previous solution  $\mathbf{x}_{k-1}$ . In particular, for  $q = 1$ , this would be a variant of the  $(P_0)$  problem. However, as opposed to the direct

formulation of  $(P_p)$ , this problem is solvable using standard linear algebra and Lagrange multipliers, as the penalty uses a plain  $\ell_2$ -norm. Thus

$$\begin{aligned}\mathcal{L}_{\mathbf{x}} &= \|\mathbf{X}_{k-1}^+ \mathbf{x}\|_2^2 + \lambda^T (\mathbf{b} - \mathbf{A} \mathbf{x}) \\ \Rightarrow \frac{\partial \mathcal{L}(\mathbf{x})}{\partial \mathbf{x}} &= 2(\mathbf{X}_{k-1}^+)^2 \mathbf{x} - \mathbf{A}^T \lambda = \mathbf{0}.\end{aligned}\quad (6.61)$$

If we assume that  $\mathbf{X}_{k-1}^+$  is invertible, this implies that  $\mathbf{X}_{k-1}^+$  has no zero entries on the main-diagonal, and thus  $(\mathbf{X}_{k-1}^+)^{-1} = \mathbf{X}_{k-1}$ . In this case the solution is given by

$$\mathbf{x}_k = \frac{1}{2} \mathbf{X}_{k-1}^2 \mathbf{A}^T \lambda. \quad (6.62)$$

In the general case, where some entries in the main-diagonal of  $\mathbf{X}_{k-1}$  are zero, the solution given above nulls the corresponding entries in  $\mathbf{x}_k$ . Assuming hereafter that a zero entry in  $\mathbf{x}_{k-1}$  should remain as such in subsequent iterations (in order to promote sparsity in the solution), this formula remains true and behaves properly. Plugging this to the constraint  $\mathbf{A} \mathbf{x} = \mathbf{b}$  gives

$$\frac{1}{2} \mathbf{A} \mathbf{X}_{k-1}^2 \mathbf{A}^T \lambda = \mathbf{b} \Rightarrow \lambda = 2(\mathbf{A} \mathbf{X}_{k-1}^2 \mathbf{A}^T)^{-1} \mathbf{b}. \quad (6.63)$$

Here again the inversion must be done with care, and in fact should be replaced by a pseudo-inverse.<sup>4</sup> Thus,

$$\mathbf{x}_k = \mathbf{X}_{k-1}^2 \mathbf{A}^T (\mathbf{A} \mathbf{X}_{k-1}^2 \mathbf{A}^T)^{-1} \mathbf{b}. \quad (6.64)$$

The approximate solution  $\mathbf{x}_k$  is used to update the diagonal matrix  $\mathbf{X}_k$  and a new iteration can begin. This algorithm is formally described in Algorithm 6.

While this algorithm is guaranteed to converge to a fixed-point, it is not necessarily the optimal one. Interestingly, Gorodnitsky and Rao show that the sequence of solutions obtained necessarily give a descent over the function  $\prod_{i=1}^m |x_i|^p$  for  $p = 0$ . Another intriguing property, already mentioned above, and obvious from Equation (6.64), is the fact that once an entry in  $\mathbf{x}_k$  is nulled, it is never revived. This implies that initialization of this algorithm should be non-zero for all entries, to enable each to be chosen in the converged result.

The FOCUSS algorithm is a practical strategy, but little is known about circumstances where it will be successful, i.e., when a numerical local minimum will actually be a good approximation to a global minimum of  $(P_0)$ . Another popular strategy is to

---

<sup>4</sup>The matrix  $\mathbf{X}_{k-1}$  chooses a subset of columns from  $\mathbf{A}$ . If the number of columns is  $n$  or more, and those span the complete space, then the inversion is the same as the pseudo-one. In case the number of columns is below  $n$  or if their rank is smaller than  $n$ , there might be a difference between the two inversions. Nevertheless, if in addition those columns span a space that contains  $\mathbf{b}$ , then the pseudo-inverse deployed provides the proper solution for  $\lambda$ . Otherwise,  $\mathbf{b}$  is not contained in those columns' span, and no  $\lambda$  satisfies this equation anyhow. However, this option never takes place if the algorithm prunes the columns of  $\mathbf{A}$  wisely (requires a proof).

---

**Algorithm 6** The IRLS strategy for solving  $(P_p)$ .

**Task:** Approximate the solution of  $(P_p)$ :  $\min_{\mathbf{x}} \|\mathbf{x}\|_p^p$ , s.t.  $\mathbf{Ax} = \mathbf{b}$ .

**Initialization:** Initialize  $k = 0$ , and set  $\mathbf{x}_0 = \mathbf{0}$  and  $\mathbf{X}_0 = \mathbf{I}$ .

**Main Iteration:** Increase  $k$  by 1, and apply these steps:

**Solve:** the linear system

$$\mathbf{x}_k = (\mathbf{X}_{k-1})^2 \mathbf{A}^T (\mathbf{A} \mathbf{X}_{k-1}^2 \mathbf{A}^T)^+ \mathbf{b},$$

either directly or iteratively (several Conjugate-Gradient iterations may suffice), producing result  $\mathbf{x}_k$ .

**Weight Update:** Update the diagonal weight matrix  $\mathbf{X}$  using  $\mathbf{x}_k : X_k(j, j) = |x_k(j)|^{1-p/2}$ .

**Stopping Rule:** If  $\|\mathbf{x}_k - \mathbf{x}_{k-1}\|_2$  is smaller than some predetermined threshold, stop. Otherwise, apply another iteration.

**Output:** The proposed solution is  $\mathbf{x}^k$ .

---

replace the  $\ell_0$  norm by the  $\ell_1$ -norm, which is, in a natural sense, its best convex approximant; many optimization tools are available ‘off the shelf’ for solving  $(P_1)$  (see next section).

Turning from  $(P_0)$  to its relaxed version,  $(P_p)$  with  $0 < p \leq 1$ , care must be taken with respect to normalization of the columns in  $\mathbf{A}$ . While the  $\ell_0$ -norm is indifferent to the magnitude of the non-zero entries in  $\mathbf{x}$ , the  $\ell_p$ -norms tend to penalize higher magnitudes, and thus bias the solution towards choosing to put non-zero entries in  $\mathbf{x}$  in locations that multiply large norm columns in  $\mathbf{A}$ . In order to avoid this bias, the columns should be scaled appropriately. Convexifying with the  $\ell_1$ -norm, the new objective becomes:

$$(P_1) : \min_{\mathbf{x}} \|\mathbf{W}^{-1} \mathbf{x}\|_1, \quad \text{s.t. } \mathbf{b} = \mathbf{Ax}. \quad (6.65)$$

The matrix  $\mathbf{W}$  is a diagonal positive-definite matrix that introduces the above-described pre-compensating weights. A natural choice for the  $(i, i)$  entry in this matrix for this case is  $w(i, i) = 1/\|\mathbf{a}_i\|_2$ . Assuming that  $\mathbf{A}$  has no zero columns, all these norms are strictly positive and the problem  $(P_1)$  is well-defined. The case where all the columns of  $\mathbf{A}$  are normalized (and thus  $\mathbf{W} = \mathbf{I}$ ) was named Basis-Pursuit (BP) by Chen, Donoho, and Saunders (1995). We will use this name for the more general setup in Equation (6.65) hereafter.

Considering (6.65) and defining  $\tilde{\mathbf{x}} = \mathbf{W}^{-1} \mathbf{x}$ , this problem can be reformulated as

$$(P_1) : \min_{\tilde{\mathbf{x}}} \|\tilde{\mathbf{x}}\|_1, \quad \text{s.t. } \mathbf{b} = \mathbf{A} \mathbf{W} \tilde{\mathbf{x}} = \tilde{\mathbf{A}} \tilde{\mathbf{x}}. \quad (6.66)$$

This means that, just as before, we can normalize the columns of  $\mathbf{A}$  and use its normalized version  $\tilde{\mathbf{A}}$ , getting the classic BP format. Just as for the greedy methods, once a solution  $\tilde{\mathbf{x}}$  has been found, it should be de-normalized to provide the required vector  $\mathbf{x}$ . Thus, from now on we can safely assume that  $(P_1)$  is given in a canonic structure with a normalized matrix.

### 6.3.3 Summary

In this chapter we have concentrated on numerical solutions to  $(P_0)$ , by either greedy techniques or relaxation ones. Each family of methods has its own merits, and choosing between the various options depends on the application at hand. All the presented algorithms are approximation ones, implying that they may fail from time to time to provide the truly sparsest solution of the linear system  $\mathbf{Ax} = \mathbf{b}$ . The natural question to ask is whether there could be conditions under which such algorithms are guaranteed to operate well. This is exactly the topic of our next section.

**练习128.** Generate a random matrix  $\mathbf{A}$  with a size  $n \times m$  ( $n \ll m$ ). Estimate  $\text{spark}(\mathbf{A})$  by the method described by (6.34)-(6.35), where subproblem (6.34) is solved by OMP. Both code and data should be submitted. (Hint: the constraints in (6.34) should be rewritten as  $\hat{\mathbf{A}}\hat{\mathbf{x}} = \hat{\mathbf{b}}$  so as to apply OMP.)

**练习129.** Use the same matrix  $\mathbf{A}$  in last exercise, redo last exercise by using IRLS. Check whether the estimation can be improved. Both code and data should be submitted.

## 6.4 Restricted Isometry Property (Chosen from [49])

The coherence is a simple and useful measure of the quality of a measurement matrix. However, the lower bound on the coherence provided in (6.29) limits the performance analysis of recovery algorithms to rather small sparsity levels. A finer measure of the quality of a measurement matrix is needed to overcome this limitation. This is provided by the concept of restricted isometry property, also known as uniform uncertainty principle. It ensures the success of most sparse recovery algorithms.

### 6.4.1 Definitions and Basic Properties

Unlike the coherence, which only takes pairs of columns of a matrix into account, the restricted isometry constant of order  $s$  involves all  $s$ -tuples of columns and is therefore more suited to assess the quality of the matrix. As with the coherence, small restricted isometry constants are desired. Here is their formal definition.

**定义130.** The  $s$ -th restricted isometry constant  $\delta_s = \delta_s(\mathbf{A})$  of a matrix  $\mathbf{A} \in \mathbb{R}^{m \times N}$  is the smallest  $\delta \geq 0$  such that

$$(1 - \delta)\|\mathbf{x}\|^2 \leq \|\mathbf{Ax}\|^2 \leq (1 + \delta)\|\mathbf{x}\|^2 \quad (6.67)$$

for all  $s$ -sparse vectors  $\mathbf{x} \in \mathbb{R}^N$ . Equivalently, it is given by

$$\delta_s = \max_{S \subset [N], \text{card}(S) \leq s} \|\mathbf{A}_S^T \mathbf{A}_S - \mathbf{I}\|_2. \quad (6.68)$$

We say that  $\mathbf{A}$  satisfies the restricted isometry property if  $\delta_s$  is small for reasonably large  $s$  – the meaning of small  $\delta_s$  and large  $s$  will be made precise later.

We make a few remarks before establishing the equivalence of these two definitions. The first one is that the sequence of restricted isometry constants is nondecreasing, i.e.,

$$\delta_p \leq \delta_q, \quad \text{if } p < q. \quad (6.69)$$

The second one is that, although  $\delta_s \geq 1$  is not forbidden, the relevant situation occurs for  $\delta_s < 1$ . Indeed, (6.68) says that each column submatrix  $\mathbf{A}_S$ ,  $S \subset [N]$  with  $\text{card}(S) \leq s$ , has all its singular values in the interval  $[1 - \delta_s, 1 + \delta_s]$  and is therefore injective when  $\delta_s < 1$ . In fact,  $\delta_{2s} < 1$  is more relevant, since the inequality (6.67) yields  $\|\mathbf{A}(\mathbf{x} - \mathbf{x}')\|^2 > 0$  for all distinct  $s$ -sparse vectors  $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^N$ ; hence, distinct  $s$ -sparse vectors have distinct measurement vectors.

**练习131.** Prove (6.69).

For a given matrix  $\mathbf{A} \in \mathbb{R}^{m \times N}$  and a sparsity level  $s$ , computing  $\delta_s$  is an NP-hard problem.

It is now possible to compare the restricted isometry constants of a matrix with its coherence  $\mu$  and coherence function  $\mu_1$  (the Babel function); see Definitions 117 and 121.

**命题132.** If the matrix  $\mathbf{A}$  has  $\ell_2$ -normalized columns, then

$$(1 - \mu_1(s - 1))\|\mathbf{x}\|^2 \leq \|\mathbf{Ax}\|^2 \leq (1 + \mu_1(s - 1))\|\mathbf{x}\|^2,$$

or equivalently, for each set  $S \subset [N]$  with  $\text{card}(S) \leq s$ , the eigenvalues of the matrix  $\mathbf{A}_S^T \mathbf{A}_S$  lie in the interval  $[1 - \mu_1(s - 1), 1 + \mu_1(s - 1)]$ . In particular, if  $\mu_1(s - 1) < 1$ , then  $\mathbf{A}_S^T \mathbf{A}_S$  is invertible.

*Proof.* For a set  $S \subset [N]$  with  $\text{card}(S) \leq s$ , since the matrix  $\mathbf{A}_S^T \mathbf{A}_S$  is positive semidefinite, it has an orthonormal basis of eigenvectors associated with real, positive eigenvalues.

We denote the minimal eigenvalue by  $\lambda_{\min}$  and the maximal eigenvalue by  $\lambda_{\max}$ . Then, since  $\mathbf{A}\mathbf{x} = \mathbf{A}_S\mathbf{x}_S$  for any  $\mathbf{x} \in \mathbb{R}^N$  supported on  $S$ , it is easy to see that the maximum of

$$\|\mathbf{A}\mathbf{x}\|^2 = \langle \mathbf{A}_S\mathbf{x}_S, \mathbf{A}_S\mathbf{x}_S \rangle = \langle \mathbf{A}_S^T \mathbf{A}_S \mathbf{x}_S, \mathbf{x}_S \rangle$$

over the set  $\{\mathbf{x} \in \mathbb{R}^N | \text{supp } \mathbf{x} \subset S, \|\mathbf{x}\| = 1\}$  is  $\lambda_{\max}$  and that its minimum is  $\lambda_{\min}$ . This explains the equivalence mentioned in the theorem. Now, due to the normalizations  $\|\mathbf{a}_j\| = 1$  for all  $j \in [N]$ , the diagonal entries of  $\mathbf{A}_S^T \mathbf{A}_S$  all equal one. By Gershgorin's disk theorem, the eigenvalues of  $\mathbf{A}_S^T \mathbf{A}_S$  are contained in the union of the disks centered at 1 with radii

$$r_j \equiv \sum_{l \in S, l \neq j} |(\mathbf{A}_S^T \mathbf{A}_S)_{j,l}| = \sum_{l \in S, l \neq j} |\langle \mathbf{a}_l, \mathbf{a}_j \rangle| \leq \mu_1(s-1), \quad j \in S.$$

Since these eigenvalues are real, they must lie in  $[1 - \mu_1(s-1), 1 + \mu_1(s-1)]$ , as announced.  $\square$

**命题133.** *If the matrix  $\mathbf{A}$  has  $\ell_2$ -normalized columns, then*

$$\delta_1 = 0, \quad \delta_2 = \mu, \quad \delta_s \leq \mu_1(s-1) \leq (s-1)\mu, \quad s \geq 2.$$

*Proof.* The  $\ell_2$ -normalization of the columns means that  $\|\mathbf{A}\mathbf{e}_j\|_2^2 = \|e_j\|_2^2$  for all  $j \in [N]$ , that is to say  $\delta_1 = 0$ . Next, with  $\mathbf{a}_1, \dots, \mathbf{a}_N$  denoting the columns of the matrix  $\mathbf{A}$ , we have

$$\delta_2 = \max_{1 \leq i \neq j \leq N} \|\mathbf{A}_{\{i,j\}}^T \mathbf{A}_{\{i,j\}} - \mathbf{I}\|_2, \quad \mathbf{A}_{\{i,j\}}^T \mathbf{A}_{\{i,j\}} = \begin{pmatrix} 1 & \langle \mathbf{a}_i, \mathbf{a}_j \rangle \\ \langle \mathbf{a}_i, \mathbf{a}_j \rangle & 1 \end{pmatrix}.$$

The eigenvalues of the matrix  $\mathbf{A}_{\{i,j\}}^T \mathbf{A}_{\{i,j\}} - \mathbf{I}$  are  $|\langle \mathbf{a}_i, \mathbf{a}_j \rangle|$  and  $-|\langle \mathbf{a}_i, \mathbf{a}_j \rangle|$ , so its operator norm is  $|\langle \mathbf{a}_i, \mathbf{a}_j \rangle|$ . Taking the maximum over  $1 \leq i \neq j \leq N$  yields the equality  $\delta_2 = \mu$ . The inequality  $\delta_s \leq \mu_1(s-1) \leq (s-1)\mu$  follows from Proposition 132.  $\square$

In view of the existence of  $m \times m^2$  matrices with coherence  $\mu$  equal to  $1/\sqrt{m}$ , this already shows the existence of  $m \times m^2$  matrices with restricted isometry constant  $\delta_s < 1$  for  $s \leq \sqrt{m}$ . We will establish that, given  $\delta < 1$ , there exist  $m \times N$  matrices with restricted isometry constant  $\delta_s \leq \delta$  for  $s \leq cm/\ln(eN/m)$ , where  $c$  is a constant depending only on  $\delta$ . This is essentially the largest range possible. Matrices with a small restricted isometry constant of this optimal order are informally said to satisfy the *restricted isometry property* or *uniform uncertainty principle*.

Certain random matrices  $\mathbf{A} \in \mathbb{R}^{m \times N}$  satisfy  $\delta_s \leq \delta$  with high probability for some  $\delta > 0$  provided

$$m \geq C\delta^{-2}s \ln(eN/s). \tag{6.70}$$

This is an optimal bound. To date, finding deterministic (i.e., explicit or at least constructible in polynomial time) matrices satisfying  $\delta_s \leq \delta$  in this regime is a major open problem. Essentially all available estimations of  $\delta_s$  for deterministic matrices combine a coherence estimation and Proposition 133 in one form or another.

Candès and Tao showed that the condition  $\delta_{2s} + \delta_{3s} < 1$  guarantees exact  $s$ -sparse recovery via  $\ell_1$ -minimization. Candès et al. further showed that the condition  $\delta_{3s} + 3\delta_{4s} < 2$  guarantees stable and robust  $s$ -sparse recovery via  $\ell_1$ -minimization. Later, a sufficient condition for stable and robust  $s$ -sparse recovery involving only  $\delta_{2s}$  was obtained by Candès, namely,  $\delta_{2s} < \sqrt{2} - 1 \approx 0.414$ .

**练习134.** Compute  $\delta_3$  for the following matrix by exhaustive search:

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$$

#### 6.4.2 Restricted Isometry Property for Subgaussian Matrices

We have mentioned that recovery of  $s$ -sparse vectors via various algorithms including  $\ell_1$ -minimization is guaranteed if the restricted isometry constants of the measurement matrix satisfy  $\delta_{\kappa s} \leq \delta_*$ . for an appropriate small integer  $\kappa$  and some  $\delta_* \in (0, 1)$ . The derived condition for  $\ell_1$ -minimization is, for instance,  $\delta_{2s} < 0.6246$ . But at this point of the theoretical development,  $m \times N$  matrices with small  $\delta_s$  are not known to exist when  $m$  is significantly smaller than  $C s^2$ . The purpose of this section is to show their existence when  $m \geq C_\delta s \ln(N/s)$  using probabilistic arguments. We consider subgaussian random matrices whose entries are drawn independently according to a subgaussian distribution. This includes Gaussian and Rademacher variables. For such matrices, the restricted isometry property holds with high probability in the stated parameter regime.

We consider a matrix  $\mathbf{A} \in \mathbb{R}^{m \times N}$  having random variables as their entries. Such  $\mathbf{A}$  is called a random matrix or random matrix ensemble.

**定义135.** (a) If the entries of  $\mathbf{A}$  are independent Rademacher variables (i.e., taking values  $\pm 1$  with equal probability), then  $\mathbf{A}$  is called a Bernoulli random matrix.

(b) If the entries of  $\mathbf{A}$  are independent standard Gaussian random variables, then  $\mathbf{A}$  is called a Gaussian random matrix.

(c) If the entries of  $\mathbf{A}$  are independent mean-zero subgaussian random variables with variance 1 and same subgaussian parameters  $\beta, \kappa$ , i.e.,

$$P(|A_{j,k}| \geq t) \leq \beta \exp(-\kappa t^2), \quad \forall t > 0, j \in [m], k \in [N], \quad (6.71)$$

then  $A$  is called a subgaussian random matrix.

Clearly, Gaussian and Bernoulli random matrices are subgaussian. Also note that the entries of a subgaussian matrix do not necessarily have to be identically distributed. We start by stating the main result on the restricted isometry property for subgaussian random matrices.

**定理136.** *Let  $\mathbf{A}$  be an  $m \times N$  subgaussian random matrix. Then there exists a constant  $C > 0$  (depending only on the subgaussian parameters  $\beta, \kappa$ ) such that the restricted isometry constant of  $\frac{1}{\sqrt{m}}\mathbf{A}$  satisfies  $\delta_s \leq \delta$  with probability at least  $1 - \varepsilon$  provided*

$$m \geq C\delta^{-2}(s \ln(eN/s) + \ln(2\varepsilon^{-1})). \quad (6.72)$$

Setting  $\varepsilon = 2 \exp(-\delta^2 m / (2C))$  yields the condition

$$m \geq 2C\delta^{-2}s \ln(eN/s),$$

which guarantees that  $\delta_s \leq \delta$  with probability at least  $1 - 2 \exp(-\delta^2 m / (2C))$ . This is the statement often found in the literature.

The normalization  $\frac{1}{\sqrt{m}}\mathbf{A}$  is natural because  $E \left\| \frac{1}{\sqrt{m}}\mathbf{A}\mathbf{x} \right\|^2 = \|\mathbf{x}\|^2$  for a fixed vector  $\mathbf{x}$  and a subgaussian random matrix  $\mathbf{A}$  (where by convention all entries have variance 1). Expressed differently, the squared  $\ell_2$ -norm of the columns of  $\mathbf{A}$  is one in expectation. Therefore, the restricted isometry constant  $\delta_s$  measures the deviation of  $\left\| \frac{1}{\sqrt{m}}\mathbf{A} \right\|^2$  from its mean, uniformly over all  $s$ -sparse vectors  $\mathbf{x}$ .

**练习137.** *Prove that  $E \left\| \frac{1}{\sqrt{m}}\mathbf{A}\mathbf{x} \right\|^2 = \|\mathbf{x}\|^2$  for a fixed vector  $\mathbf{x}$  and a subgaussian random matrix  $\mathbf{A}$  (where by convention all entries have variance 1).*

#### 6.4.3 Variants of RIP

Inspired by RIP for vectors, researchers have proposed other RIP under different situations.

**定义138.** *(Page 174 of [49]) For a linear mapping  $\mathcal{A} : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^p$ , it is said to have Rank Restricted Isometry Property ( $R$ -RIP( $r$ )) if there exists a constant  $\delta \geq 0$  such that*

$$(1 - \delta)\|\mathbf{X}\|_F^2 \leq \|\mathcal{A}(\mathbf{X})\|^2 \leq (1 + \delta)\|\mathbf{X}\|_F^2 \quad (6.73)$$

for all  $\mathbf{X} \in \mathbb{R}^{m \times n}$  whose rank does not exceed  $r$ . The smallest  $\delta$  is call the  $r$ -th rank restricted isometric constant  $\delta_r = \delta_r(\mathcal{A})$ .

R-RIP has been widely used to analyze low-rank recovery problems.

**定义139.** ([9]) For a linear mapping  $\mathcal{A} : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^p$ , it is said to have Scalable Restricted Isometry Property (S-RIP( $d, \alpha$ )) if there exist two positive constants  $\mu_d$  and  $\nu_d$  such that  $\mu_d/\nu_d \leq \alpha$  and

$$\nu_d \|\mathbf{X}\|_F \leq \|\mathcal{A}(\mathbf{X})\| \leq \mu_d \|\mathbf{X}\|_F, \quad \forall \mathbf{X} \in \mathcal{C}_d, \quad (6.74)$$

where  $\mathcal{C}_d$  is a set indexed by nonnegative integer  $d$ . The smallest  $\alpha$  is called the  $d$ -th scalable restricted isometric constant  $\alpha_d = \alpha_d(\mathcal{A})$ .

Usually,  $\mathcal{C}_d$  is defined as

$$\mathcal{C}_d = \{\mathbf{X} | \varphi(\mathbf{X}) \leq d\},$$

where  $\varphi : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^+$  has the following properties:

1.  $\varphi(\mathbf{0}) = 0$ ;
2.  $\varphi(\mathbf{X}) = \varphi(-\mathbf{X})$  (symmetry);
3.  $\varphi(\mathbf{X} + \mathbf{Y}) \leq \varphi(\mathbf{X}) + \varphi(\mathbf{Y})$  (subadditivity).

Such  $\varphi$  includes  $\ell_0$ -norm,  $\ell_p$ -norm and rank, which can be non-convex.

**命题140.** Suppose that  $d_1 \leq d_2$  and  $\alpha_1 \geq \alpha_2$ . If SRIP( $d_1, \alpha_1$ ) is satisfied, then SRIP( $d_2, \alpha_2$ ) is also satisfied.

Plugging

$$\mu_d^2 = 1 + \delta_d, \quad \nu_d^2 = 1 - \delta_d$$

in S-RIP, the relationship between RIP and S-RIP is as follows.

**命题141.** Let  $\beta \in (0, 1)$ . If RIP( $d, \delta_d$ ) is satisfied for  $\delta_d < \beta$ , then SRIP( $d, \sqrt{\frac{1+\beta}{1-\beta}}$ ) holds true.

Consider the nonconvex feasibility problem:

$$\text{Find } \mathbf{X} \in \mathcal{C}_s, \text{ such that } \mathcal{A}(\mathbf{X}) = \mathbf{b}.$$

We may consider the related nonconvex minimization problem instead:

$$\min_{\mathbf{X} \in \mathcal{C}_s} f(\mathbf{X}) \equiv \frac{1}{2} \|\mathcal{A}(\mathbf{X}) - \mathbf{b}\|^2.$$

**定理142.** If  $SRIP(2s, \sqrt{2}\eta)$  is satisfied, then the function values of the sequence generated by the gradient projection algorithm<sup>5</sup>, with stepsize  $T_k \in [\mu_{2s}^2, 2\nu_{2s}^2]$  and backtracking constant  $\eta$ <sup>6</sup>, converge linearly to zero, i.e., a feasible solution is found.

**练习143.** Prove Proposition 140.

**练习144.** Prove Proposition 141.

## 6.5 The Quest for a Dictionary

A fundamental ingredient in the definition of Sparse-Land's signals and its deployment to applications is the dictionary  $\mathbf{A}$ . How can we wisely choose  $\mathbf{A}$  to perform well on the signals in question? This is the topic of this chapter, and our emphasis is put on learning methods for dictionaries, based on a group of examples.

### 6.5.1 Choosing versus Learning

In the quest for the proper dictionary to use in applications, one line of work considers choosing pre-constructed dictionaries, such as undecimated wavelets, steerable wavelets, contourlets, curvelets, and more. Many of these recently proposed dictionaries are tailored specifically to images, and in particular to stylized ‘cartoon-like’ image content, assumed to be piecewise smooth and with smooth boundaries.

Some of these proposed dictionaries (which are often referred to also as transforms) are accompanied by a detailed theoretical analysis establishing the sparsity of the representation coefficients for such simplified content of signals. This is typically done in terms of the rate of decay of the  $M$ -term approximation – the representation of the signal using the best  $M$  non-zeros from the transform coefficients.

Alternatively, one can use a tunable selection of a dictionary, in which a basis or Frame is generated under the control of a particular parameter (discrete or continuous). The two most known examples that belong to this category are the wavelet packets and the bandelets. The wavelet packets, due to Coifman et al., suggest a control over the time-frequency subdivision, tuned to optimize performance over a specified instance of a signal. The bandelets, due to Mallat et al., are spatially adapted to better treat images, by orienting the regular wavelet atoms along the principal direction of the local area processed.

While pre-constructed or adapted dictionaries typically lead to fast transforms (of complexity  $O(n \log n)$  instead of  $nm$  in a direct use), they are typically limited in their

---

<sup>5</sup> $\mathbf{X}_{k+1} \in P_{\mathcal{C}_s}(\mathbf{X}_k - \nabla f(\mathbf{X}_k)/T_k)$

<sup>6</sup>If  $\|\mathcal{A}(\mathbf{X}_{k+1} - \mathbf{X}_k)\| > \sqrt{T_k} \|\mathbf{X}_{k+1} - \mathbf{X}_k\|$ , then  $T_k \leftarrow \eta T_k$

ability to sparsify the signals they are designed to handle. Furthermore, most of those dictionaries are restricted to signals/images of a certain type, and cannot be used for a new and arbitrary family of signals of interest. This leads us to yet another approach for obtaining dictionaries that overcomes these limitations – by adopting a learning point-of-view.

This learning option starts by building a training database of signal instances, similar to those anticipated in the application, and constructing an empirically learned dictionary, in which the generating atoms come from the underlying empirical data, rather than from some theoretical model. Such a dictionary can then be used in the application as a fixed and redundant dictionary. We shall explore this third option in more detail in this chapter.

As opposed to the pre-constructed and adapted dictionaries, the learning method is able to adapt to any family of signals that complies with the Sparse-Land model. However, this comes at the price of a much higher computational load – learned dictionaries are held as explicit structure-less matrices, and using them in applications involves many more computations, compared to the pre-constructed dictionaries. Another shortcoming of the training methodology is its restriction to low dimensional signals. This is why handling of images is done with such dictionaries on small patches. We will discuss ways to overcome these two limitations later in this section.

### 6.5.2 Dictionary-Learning Algorithms

We now turn to discuss the learning methodology for constructing  $\mathbf{A}$ . Assume that a training database  $\{\mathbf{y}_i\}_{i=1}^M$  is given, and thought to have been generated by some fixed but unknown model  $\mathcal{M}_{\{\mathbf{A}, k_0, \alpha, \varepsilon\}}$ . Can this training database allow us to identify the generating model, and specifically the dictionary  $\mathbf{A}$ ? This rather difficult problem has been studied initially by Field and Olshausen in 1996, who were motivated by an analogy between the atoms of a dictionary and the population of simple cells in the visual cortex. They thought that learning a dictionary empirically might model evolutionary processes that led to the existing collection of simple cells; and they indeed were able to find a rough empirical match between the properties of a learned dictionary and some known properties of the population of simple cells.

Later work by Lewicki, Engan, Rao, Gribonval, Aharon, and others, extended their methodology and algorithm in various ways. Here we describe two training mechanisms, the first named Method of Optimal Directions (MOD) by Engan et al., and the second named K-SVD, by Aharon et al..

### 6.5.2.1 Core Questions in Dictionary-Learning

Assume that  $\varepsilon$  – the model deviation – is known, and our aim is the estimation of  $\mathbf{A}$ . Consider the following optimization problem:

$$\min_{\mathbf{A}, \{\mathbf{x}_i\}_{i=1}^M} \sum_{i=1}^M \|\mathbf{x}_i\|_0, \quad s.t. \quad \|\mathbf{y}_i - \mathbf{Ax}_i\| \leq \varepsilon, 1 \leq i \leq M. \quad (6.75)$$

This problem describes each given signal  $\mathbf{y}_i$  as the sparsest representation  $\mathbf{x}_i$  over the unknown dictionary  $\mathbf{A}$ , and aims to jointly find the proper representations and the dictionary. Clearly, if a solution has been found such that every representation has  $k_0$  or fewer non-zero entries, a candidate feasible model  $\mathcal{M}$  has been found.

The roles of the penalty and the constraints in Equation (6.75) might also be reversed, if we choose to constrain the sparsity and obtain the best fit for it,

$$\min_{\mathbf{A}, \{\mathbf{x}_i\}_{i=1}^M} \sum_{i=1}^M \|\mathbf{y}_i - \mathbf{Ax}_i\|^2, \quad s.t. \quad \|\mathbf{x}_i\|_0 \leq k_0, 1 \leq i \leq M. \quad (6.76)$$

Are these two problems properly posed? Do they have meaningful solutions? There are some obvious indeterminacies (scaling and permutation of the columns). Still, if we fix a scale and ordering, is there a meaningful solution to the problems posed in Equations (6.75) and (6.76) in general? As we show next, the answers to these questions are positive.

In terms of well-posedness of the dictionary-learning problem presented above, a fundamental question is whether there is a uniqueness property underlying this problem, implying that only one dictionary exists, that sparsely explains the set of training vectors. Surprisingly, at least for the case of  $\varepsilon = 0$ , there is an answer to this question, as shown by Aharon et al.. Suppose there exists a dictionary  $\mathbf{A}_0$  and a sufficiently diverse database of examples, all of which are representable using at most  $k_0 < \text{spark}(\mathbf{A}_0)/2$  atoms. Then, up to a re-scaling and permutation of the columns,  $\mathbf{A}_0$  is the unique dictionary that achieves this sparsity for all the elements in the training database.

Some readers may prefer to think in terms of matrix factorizations. Concatenate all the database vectors column-wise, forming an  $n \times M$  matrix  $\mathbf{Y}$ , and similarly, all the corresponding sparse representations into a matrix  $\mathbf{X}$  of size  $m \times M$ ; thus the dictionary satisfies  $\mathbf{Y} = \mathbf{AX}$  in the noiseless case. The problem of discovering an underlying dictionary is thus the same as the problem of discovering a factorization of the matrix  $\mathbf{Y}$  as  $\mathbf{AX}$  where  $\mathbf{A}$  and  $\mathbf{X}$  have the indicated sizes, and  $\mathbf{X}$  has sparse columns. The matrix factorization viewpoint connects this problem with related problems of nonnegative matrix factorization and in particular, sparse nonnegative matrix factorization.

### 6.5.2.2 The MOD Algorithm

Clearly, there is no general practical algorithm for solving problem (6.75) or (6.76), for the same reasons that there is no general practical algorithm for solving  $(P_0)$ , only more so. However, just as with  $(P_0)$ , the lack of general guarantees is no reason not to try heuristic methods and see how they perform in specific cases.

We can view the problem posed in Equation (6.75) as a nested minimization problem: an inner minimization of the number of nonzeros in the representation vectors  $\mathbf{x}_i$ , for a given fixed  $\mathbf{A}$  and an outer minimization over  $\mathbf{A}$ . A strategy of alternating minimization thus seems very natural; at the  $k$ -th step, we use the dictionary  $\mathbf{A}_{(k-1)}$  from the  $(k-1)$ -th step and solve  $M$  instances of  $(P_0^\varepsilon)$ , one for each database entry  $\mathbf{y}_i$ , and each using the dictionary  $\mathbf{A}_{(k-1)}$ . This gives us the matrix  $\mathbf{X}_{(k)}$ , and we then solve for  $\mathbf{A}_{(k)}$  by Least-Squares,

$$\begin{aligned}\mathbf{A}_{(k)} &= \underset{\mathbf{A}}{\operatorname{argmin}} \|\mathbf{Y} - \mathbf{AX}_{(k)}\|_F^2 \\ &= \mathbf{Y}\mathbf{X}_{(k)}^+, \end{aligned}\tag{6.77}$$

where we evaluate the error using a Frobenius norm. We may also re-scale the columns of the obtained dictionary. We increase  $k$  and unless we have satisfied a convergence criterion, we repeat the above loop. Such a Block-Coordinate-Relaxation algorithm has been proposed by Engan et al., and termed Method of Optimal Directions (MOD).

### 6.5.2.3 The K-SVD Algorithm

A different update rule for the dictionary can be proposed, in which the atoms (i.e., columns) in  $\mathbf{A}$  are handled sequentially. This leads to the K-SVD algorithm, as developed by Aharon et al. Keeping all the columns fixed apart from the  $j_0$ -th one,  $\mathbf{a}_{j_0}$ , this column can be updated along with the coefficients that multiply it in  $\mathbf{X}$ . We isolate the dependency on  $\mathbf{a}_{j_0}$  by rewriting (6.77) as

$$\begin{aligned}\|\mathbf{Y} - \mathbf{AX}_{(k)}\|_F^2 &= \|\mathbf{Y} - \sum_{j=1}^m \mathbf{a}_j \mathbf{x}_j^T\|_F^2 \\ &= \left\| \left( \mathbf{Y} - \sum_{j \neq j_0} \mathbf{a}_j \mathbf{x}_j^T \right) - \mathbf{a}_{j_0} \mathbf{x}_{j_0}^T \right\|_F^2. \end{aligned}\tag{6.78}$$

In this description,  $\mathbf{x}_j^T$  stands for the  $j$ -th row of  $\mathbf{X}$ . The update step targets on both  $\mathbf{a}_{j_0}$  and  $\mathbf{x}_{j_0}^T$ , and refers to the term in parentheses,

$$\mathbf{E}_{j_0} = \mathbf{Y} - \sum_{j \neq j_0} \mathbf{a}_j \mathbf{x}_j^T\tag{6.79}$$

as a known pre-computed error matrix.

The optimal  $\mathbf{a}_{j_0}$  and  $\mathbf{x}_{j_0}^T$  minimizing Equation (6.78) are the rank-1 approximation of  $\mathbf{E}_{j_0}$ , and can be obtained via an SVD, but this typically would yield a dense vector  $\mathbf{x}_{j_0}^T$ , implying that we increase the number of non-zeros in the representations in  $\mathbf{X}$ .

In order to minimize this term while keeping the cardinalities of all the representations fixed, a subset of the columns of  $\mathbf{E}_{j_0}$  should be taken – those that correspond to the signals from the example-set that are using the  $j_0$ -th atom, namely those columns where the entries in the row  $\mathbf{x}_{j_0}^T$  are non-zero. This way, we allow only the existing non-zero coefficients in  $\mathbf{x}_{j_0}^T$  to vary, and the cardinalities are preserved.

Therefore, we define a restriction operator,  $\mathbf{P}_{j_0}$ , that multiplies  $\mathbf{E}_{j_0}$  from the right to remove the non-relevant columns. The matrix  $\mathbf{P}_{j_0}$  has  $M$  rows (the number of overall examples), and  $M_{j_0}$  columns (the number of examples using the  $j_0$ -th atom). We define  $(\mathbf{x}_{j_0}^R)^T = \mathbf{x}_{j_0}^T \mathbf{P}_{j_0}$  as the restriction on the row  $\mathbf{x}_{j_0}^T$ , choosing the non-zero entries only.

For the sub-matrix,  $\mathbf{E}_{j_0} \mathbf{P}_{j_0}$ , a rank-1 approximation via SVD can be applied, updating both the atom  $\mathbf{a}_{j_0}$  and the corresponding coefficients in the sparse representations,  $\mathbf{x}_{j_0}^R$ . This simultaneous update may lead to a substantial speedup in the convergence of the training algorithm. Algorithm 7 describes the K-SVD algorithm in detail.

Note that one need not use the full SVD as only the first rank part of  $\mathbf{E}_{j_0} \mathbf{P}_{j_0}$  is required. An alternative numerical approach for finding  $\mathbf{a}_{j_0}$  and  $\mathbf{x}_{j_0}^R$  can be proposed, following the same block-coordinate rationale: Fixing  $\mathbf{a}_{j_0}$ , we can update  $\mathbf{x}_{j_0}^R$  by a plain Least-Squares that solving

$$\min_{\mathbf{x}_{j_0}^R} \left\| \mathbf{E}_{j_0} \mathbf{P}_{j_0} - \mathbf{a}_{j_0} (\mathbf{x}_{j_0}^R)^T \right\|_F^2 \Rightarrow \mathbf{x}_{j_0}^R = \frac{\mathbf{P}_{j_0}^T \mathbf{E}_{j_0}^T \mathbf{a}_{j_0}}{\|\mathbf{a}_{j_0}\|^2}.$$

Once updated, it is kept fixed, and we update  $\mathbf{a}_{j_0}$  by

$$\min_{\mathbf{a}_{j_0}} \left\| \mathbf{E}_{j_0} \mathbf{P}_{j_0} - \mathbf{a}_{j_0} (\mathbf{x}_{j_0}^R)^T \right\|_F^2 \Rightarrow \mathbf{a}_{j_0} = \frac{\mathbf{E}_{j_0} \mathbf{P}_{j_0} \mathbf{x}_{j_0}^R}{\|\mathbf{x}_{j_0}^R\|^2}.$$

Few such rounds of updates for these two unknowns is sufficient for getting the desired dictionary update.

Interestingly, if the above process is considered for the case where  $k_0 = 1$ , and constraining the representation coefficients to be binary (1 or 0), this problem reduces to a simple clustering task. Furthermore, in such a case, both the above training algorithms simplify to become the well-known K-means algorithm. While each iteration of K-means computes the means of  $K$  different subsets, the K-SVD algorithm performs an SVD for each of the  $K$  different sub-matrices, and thus the name K-SVD ( $K$  is used as the number of columns in  $\mathbf{A}$ ).

The fact that the dictionary-learning problem is a generalization of clustering reveals more on the learning task considered here:

1. Just as for the clustering problem and the K-Means which aims to solve it, we cannot guarantee that the MOD and K-SVD algorithms obtain the global minimum of the penalty function posed in Equation (6.76). Indeed, even a local minimum solution cannot be guaranteed, as these algorithms may get stuck on a saddle-point steady-state solution.
2. Still on the matter of convergence, since the pursuit applied in the sparse coding stage may get to a sub-optimal solution, we cannot guarantee a monotonic nonincreasing penalty value as a function of the iterations. However, we can modify the algorithm such that after the pursuit we adopt only those examples that show a descent in the representation error, thus guaranteeing an overall descent.
3. The similarity between the K-Means and the K-SVD (and MOD) enables us to exploit the vast empirical experience accumulated on ways to accelerate the K-Means. These techniques include a gradual change in the number of atoms within the learning procedure, a similar change in the number of atoms to use per example, a multi-scale approach, where the examples are represented by multiscale pyramids, and the training is done on lower-dimensions first, and more. We shall not dwell on these techniques here.

More dictionary learning methods can be found in [33, 125].

Add discriminative dictionary learning, review on IEEE SPM.

Add group sparsity.

### 6.5.3 Ordering a Dictionary (Chosen from [77])

Different from Fourier or wavelet transform, which depends upon pre-constructed or pre-defined bases, sparse representation uses data-adaptive over-complete bases, also called over-complete *dictionaries*. Besides being over-complete, current dictionaries are usually unstructured. In contrast, in Fourier or wavelet transform, the bases are well ordered according to their frequencies, which have an intuitive physical interpretation: low and high frequency bases correspond to slow-varying and fast-varying waveforms, respectively (Fig. 6.6). Note that the concept of frequency plays a critical role in the traditional signal processing theory. Unfortunately, for over-complete dictionaries such a concept is lacking. This incurs a lot of disadvantages. For example, one has to display an

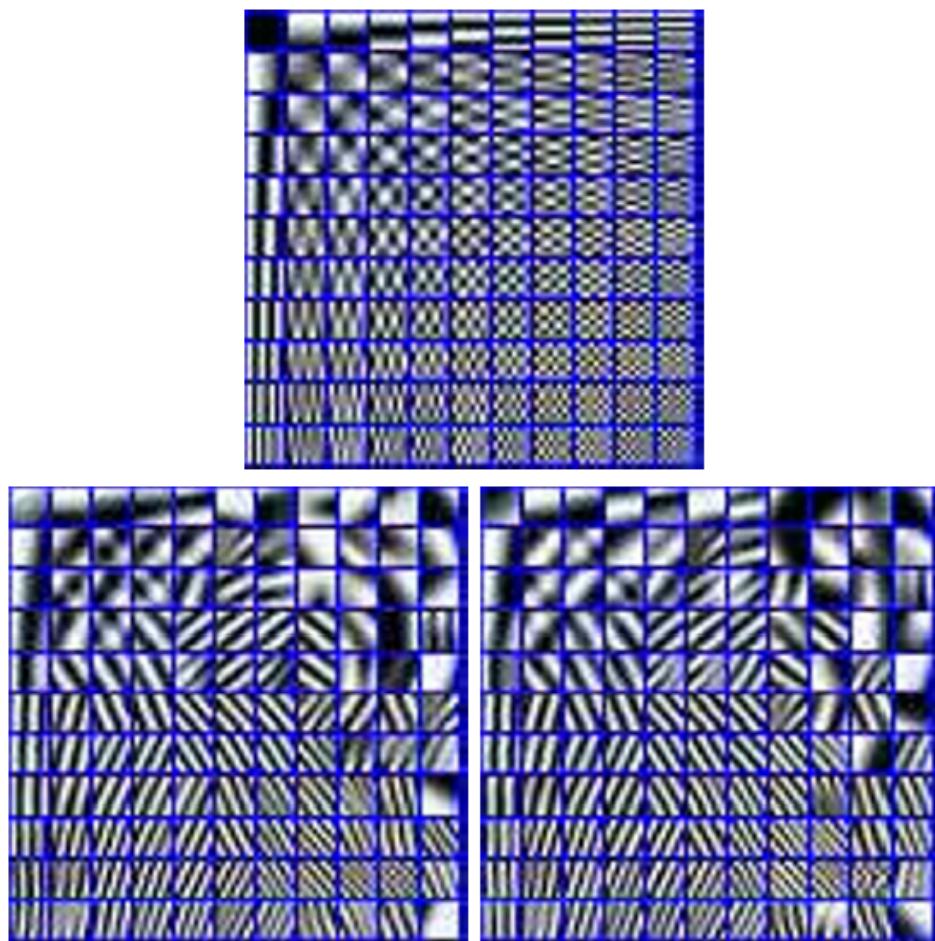


图 6.5: The DCT (top), the MOD (bottom left) and the K-SVD (bottom right) dictionaries, trained using  $8 \times 8$  patches from the image Barbara.

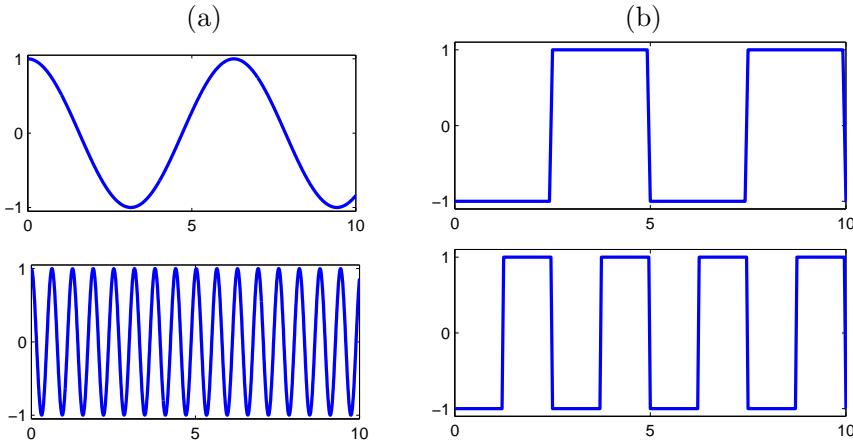


图 6.6: The low frequency bases and high frequency bases. (a) Cosine functions. (b) Haar wavelets.

over-complete dictionary in a random order, and when performing data compression it is unclear which bases should be kept and which should be discarded. In this paper, we aim at generalizing the concept of “frequency” for over-complete dictionaries. Once the generalized frequency is defined, many concepts and techniques in the traditional signal processing may be transplanted to sparse representation.

For general over-complete dictionaries, we can no longer define their frequencies by looking at whether a basis is fast or slowly varying, because the bases may not be in waveforms. For example, for a given basis of 128-dimensional SIFT features (see Fig. 6.7 (b)) it is hard to say whether its “frequency” is low or high. This is nontrivial even for a dictionary learnt from raw image patches (see Fig. 6.7 (a)). As a result, in the literature, people simply show and store their over-complete dictionaries in a random order. In this paper, we propose to define “frequency” for an over-complete dictionary by sorting the bases appropriately, so that the generalized frequency can indeed be a generalization of frequency in the traditional sense, when the dictionary reduces to the traditional bases. The algorithm is called Bases Sorting (BS). After sorting, the leading bases can be regarded as of “low frequency” while the ending bases regarded as of “high frequency.” The sorting criterion is inspired by the  $1/f$ -power law of the Fourier spectra of natural images. As the over-complete dictionary is data-adaptive, so should the orders of the bases be. Given training data, for each sample BS first computes the sparsest representation coefficients with respect to the dictionary, then sorts the bases according to the magnitudes of the coefficients. The final order of a basis is its average position over the training data.

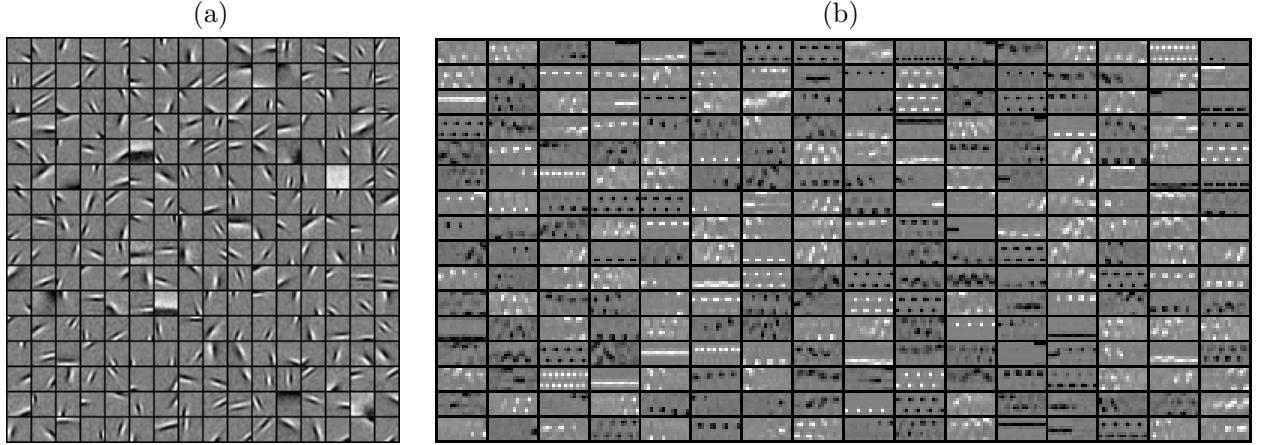


图 6.7: Unsorted dictionaries. (a) Dictionary learnt from  $14 \times 14$  raw image patches. (b) Dictionary learnt from 128-dimensional SIFT features.

### 6.5.3.1 Basis Sorting Algorithm

It has been observed that statistics of many man-made or natural objects, such as city size, incomes, word frequencies, earthquake magnitudes, and the vertex degrees in the World Wide Web, obey a power law distribution, i.e., small magnitudes are very common, whereas large magnitudes are very rare. In the study of natural image statistics, Field discovered a  $1/f$ -power law, i.e., by performing Fourier transform on natural images the magnitudes of the Fourier coefficients at frequency  $f$  (averaged over orientations) are proportional to  $1/f$ . For wavelet transforms, the magnitudes of wavelet coefficients also obey this law.

Inspired by the above observations, we propose to sort the bases in an over-complete dictionary so that the magnitudes of the sparsest representation coefficients of data are also in a descending order. In this way, the indices of bases can be naturally *interpreted* as frequencies.

According to this principle, a straightforward way, called the baseline method, is to first compute the average magnitudes of the representation coefficients of the training samples and then sort the bases in the descending order of the averaged magnitudes. However, we will show that such a method is *not* the best approach. A much better way is to first sort the bases according to the magnitudes of the representation coefficients for each training sample, then record the position of each basis, and finally resort the bases according to their averaged positions. We call our method the Bases Sorting (BS) algorithm, as detailed below.

Given an over-complete dictionary  $U$  which is learnt from data set  $X$ , for  $\mathbf{v} \in \mathbb{R}^m$ ,  $\text{supp}(\mathbf{v}) = \{j \in I | v_j \neq 0\}$  denotes the support of  $\mathbf{v}$ , where  $I = \{1, 2, \dots, m\}$ . For the sparsest representation  $\mathbf{v}$  of  $\mathbf{x} \in X$  w.r.t.  $U$ , we define the non-zero components  $\{v_j | j \in \text{supp}(\mathbf{v})\}$  as the *active coefficients* of  $\mathbf{x}$  and the subset of bases  $U^{(\mathbf{x})} = \{\mathbf{u}_j | j \in \text{supp}(\mathbf{v})\}$  as  $\mathbf{x}$ -*activated bases*. Then we can sort  $U^{(\mathbf{x})}$  in an descending order of the absolute values of their coefficients  $\{|v_j|\}$ . The indices of the  $\mathbf{x}$ -activated bases are recorded as  $\pi(U^{(\mathbf{x})})$ . Then the final order of bases are the expectation of the orders, i.e.,

$$\pi(U) = \mathbb{E}[\pi(U^{(\mathbf{x})})], \quad (6.80)$$

where  $\mathbb{E}[\cdot]$  is the expectation operator.

Formally our BS algorithm consists of four steps:

1. For each data vector  $\mathbf{x}_i \in X$ , calculate its sparsest representation vector  $\mathbf{v}_i$  with respect to the dictionary  $U$ ;
2. Determine the order  $\pi(U^{(\mathbf{x}_i)})$  of bases in  $U^{(\mathbf{x}_i)}$  by sorting the magnitudes of the corresponding active coefficients  $\{|v_j| | j \in \text{supp}(\mathbf{v}_i)\}$  of  $\mathbf{x}_i$  in descending order;
3. Record all the order  $\pi(U^{(\mathbf{x}_i)})$  to obtain an averaged frequency order  $\pi(U)$ .
4. Sort the bases in  $U$  in an ascending order of  $\pi(U)$ .

The pseudo code of BS algorithm is presented in Alg. 8. Note that as calculating the sparsest representation of training samples is a part of dictionary learning, sorting the bases is simply a by-product of dictionary learning and actually does not add to the computation load.

With the sorted dictionary  $U$ , we can say that  $\mathbf{u}_j$  is *of frequency*  $j$  and the bases in  $U$  are sorted from low to high frequencies. The sparsest representation coefficients  $\mathbf{v}$  of a data vector  $\mathbf{x}$ , arranged according to the corresponding bases, is called the *spectrum* of  $\mathbf{x}$ . If the active bases of a data vector  $\mathbf{x}$  are all low, middle, or high frequency bases, we can say that  $\mathbf{x}$  is a *low, middle, or high frequency signal*, respectively. Thus the traditional concepts related to frequency are naturally generalized.

### 6.5.3.2 Experiments

To visualize sorted dictionaries when the bases are 2D signals, we arrange their bases in two schemes, row-by-row and zigzag, as illustrated in Fig. 6.8.

**Validations:** A natural question about our generalized frequency is whether it is consistent with the classical concept of frequency. To answer this question, we first apply

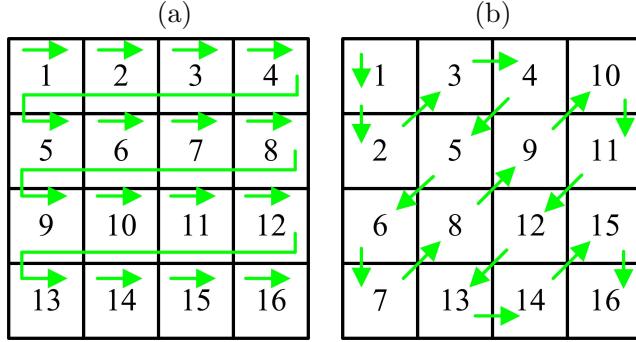


图 6.8: Two schemes to arrange the sorted bases. (a) The row-by-row arrangement. (b) The zigzag arrangement.

our BS algorithm to sort the standard 2D-DCT bases. The training data  $X$  consist of 200,000 image patches of  $16 \times 16$  pixels, which are randomly sampled from the images of Scene-15<sup>7</sup>. The dictionary  $U$  simply consists of the 256 standard 2D-DCT bases (see Fig. 6.9(a)), rearranged into 256-dimensional vectors.

The 256 2D-DCT bases sorted by the BS algorithm are displayed in Fig. 6.9 (b) and (c), in row-by-row and zigzag schemes, respectively. As can be seen, the 2D-DCT bases are organized in a good order: the visually low frequency bases are in the front and the visually high frequency bases are at the end.

As the BS results on the 2D-DCT bases is not identical to the natural order, it is not easy to tell visually how much they differ. To compute the difference between them quantitatively, we recall that the 2D-DCT bases on the same anti-diagonals in Fig. 6.9(a) are considered of the same frequency. So we define the following zigzag step function  $z(j)$  that maps the  $j$ -th sorted basis to the  $z(j)$ -th anti-diagonal (or called the  $z(j)$ -th frequency index):

$$z(j) = \begin{cases} 1, & j = 1, \\ 2, & j = 2, 3, \\ 3, & j = 4, 5, 6, \\ \dots \\ 29, & j = 251, 252, 253, \\ 30, & j = 254, 255, \\ 31, & j = 256. \end{cases} \quad (6.85)$$

<sup>7</sup>Scene-15 is a benchmark data set for scene classification, available at [http://www-cvr.ai.uiuc.edu/ponce\\_grp/data/](http://www-cvr.ai.uiuc.edu/ponce_grp/data/).

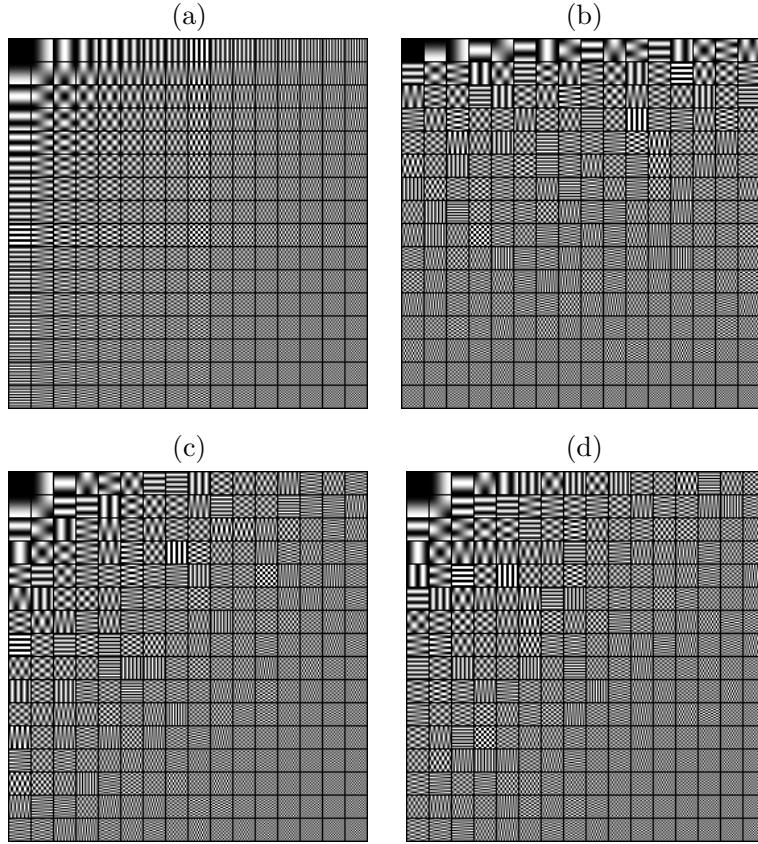


图 6.9: Validation experiments on 256 standard 2D-DCT bases. (a) The standard 2D-DCT bases in the natural order. (b) and (c) are the results of BS shown in two arrangements. (d) The result of baseline shown in zigzag arrangement.

The zigzag step functions of both the natural order and generalized frequency order are shown in Fig. 6.10 (a). We can see that they agree with each other fairly well. The average difference between their zigzag step functions is only 0.4430. For comparison, we present the results of baseline in Fig. 6.9 (d) and Fig. 6.10 (b). It can be seen that baseline is worse than BS, especially when the index is between 100 and 250. Actually, the average difference between the zigzag step functions of baseline and natural order is 0.6172.

The generalized frequency defined by BS is actually also more robust than that by baseline. To show this, we compute the zigzag step functions of bases sorted using different numbers of training samples and then calculate their difference as follows:

$$d^{(n)} = \frac{1}{m} \|\mathbf{z}^* - \mathbf{z}^{(n)}\|_1, \quad (6.86)$$

where  $\mathbf{z}^*$  is the zigzag step function of the generalized frequency order estimated by using

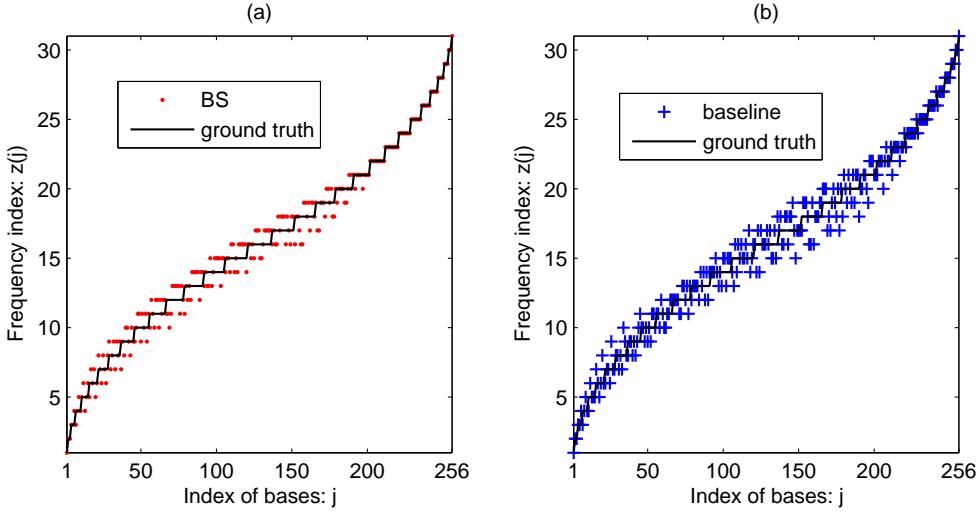


图 6.10: Frequency indices of the 256 2D-DCT bases sorted by (a) BS and (b) baseline, respectively.

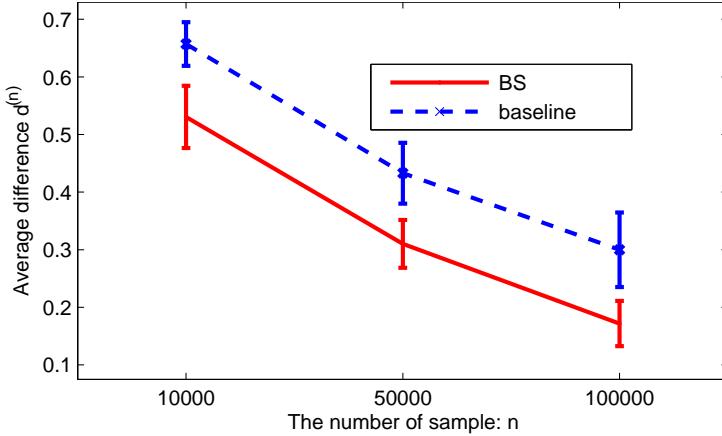


图 6.11: Stability comparison of the estimated generalized frequency order.

200,000 training samples,  $\mathbf{z}^{(n)}$  is the zigzag step function estimated by using  $n$  training samples, and  $m$  is the number of bases in  $U$ . We randomly select  $n = 10^4, 5 \times 10^4, 10^5$  training samples to estimate the generalized frequency order and calculate the distances  $d^{(n)}$ . The means and standard variances of distance  $d^{(n)}$  over 20 trials are displayed in Fig. 6.11. We can see that the generalized frequency order by BS is much more stable than that by baseline when the number of training samples changes. This is because BS uses the expectation of order to define its generalized frequency order.

To test whether the  $1/f$ -power law can be preserved by our generalized frequency, we randomly sample one million raw image patches from the images of Scene-15 as test data and show in Fig. 6.12 (a) the average magnitudes of the coefficients of the bases when representing the *test* data. We can see that the average magnitudes fall off quickly

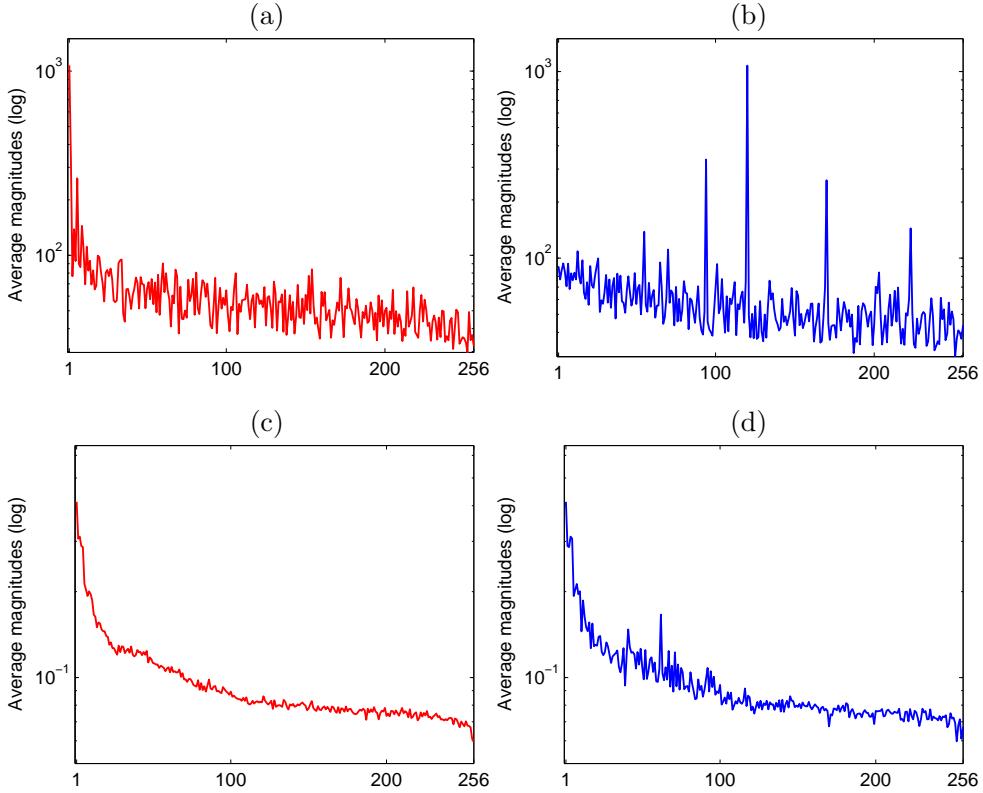


图 6.12: The average spectra of the test data in log scale. The top and bottom rows are the results of using the sorted dictionary consisting of 256 bases learnt from raw image patches and SIFT features, respectively. The left and right columns are the average spectra defined by BS and baseline, respectively.

when the generalized frequency increases. The fall-off curve is very close to Laplacian. This suggests that the  $1/f$ -power law is indeed preserved by our generalized frequency. For the baseline method, there are several high peaks in the “mid-and-high frequencies,” as shown in Fig. 6.12 (b). Similar phenomenon can be observed on the dictionary learnt from 50,000 128-D SIFT features (Fig. 6.12 (c) and (d)).

These experiments suggest that our generalized frequency is indeed a good generalization of the traditional frequency.

**Dictionary Visualization:** A straightforward application of generalized frequency is to display the bases in a good order.

First, we randomly sample 50,000 image patches of  $14 \times 14$  pixels from 10 natural scene images<sup>8</sup> and learn an over-complete dictionary with 256 bases (Fig. 6.13 (a)). The dictionary sorted by our BS algorithm is displayed in Fig. 6.13 (b) and (c) in two

<sup>8</sup>The 10 images are available at <http://ai.stanford.edu/~hllee/softwares/nips06-sparsecoding.htm>.

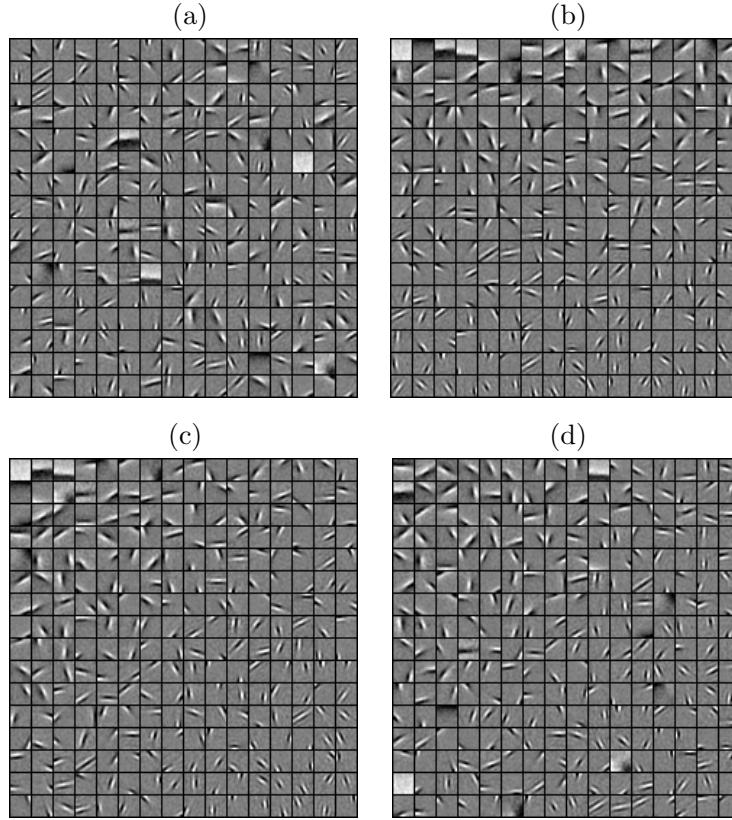


图 6.13: Visualization of an over-complete dictionary of size 256, learnt from 50,000 image patches. (a) The orderless display. (b) and (c) are the sorted results by BS in row-by-row and zigzag arrangements, respectively. (d) The result of the baseline in zigzag arrangement.

arrangement schemes. We can see that visually low frequency bases are sorted before visually high frequency ones. The bases sorted by the baseline method is presented in Fig. 6.13 (d). As can be seen, the bases are not sorted properly. For example, the lowest frequency basis (i.e. the brightest block) is not at the leading position. So our BS algorithm can yield better sorting result than the baseline.

Second, we learn an over-complete dictionary on 50,000 128-dimensional SIFT features (Fig. 6.14 (a)) randomly sampled from images in the Scene-15 data set and sort it by BS. The learnt bases look like Morse code. The sorted dictionary is displayed in Fig. 6.14 (b) and (c) in two arrangement schemes. Compared with the orderless display in Fig. 6.14 (a), the sorted bases are in an interesting order, which seems to change from simple pattern (e.g., regularly spaced dots or dashes) to complex patterns (e.g., irregularly spaced dots). The bases sorted by the baseline method is presented in Fig. 6.14 (d). It

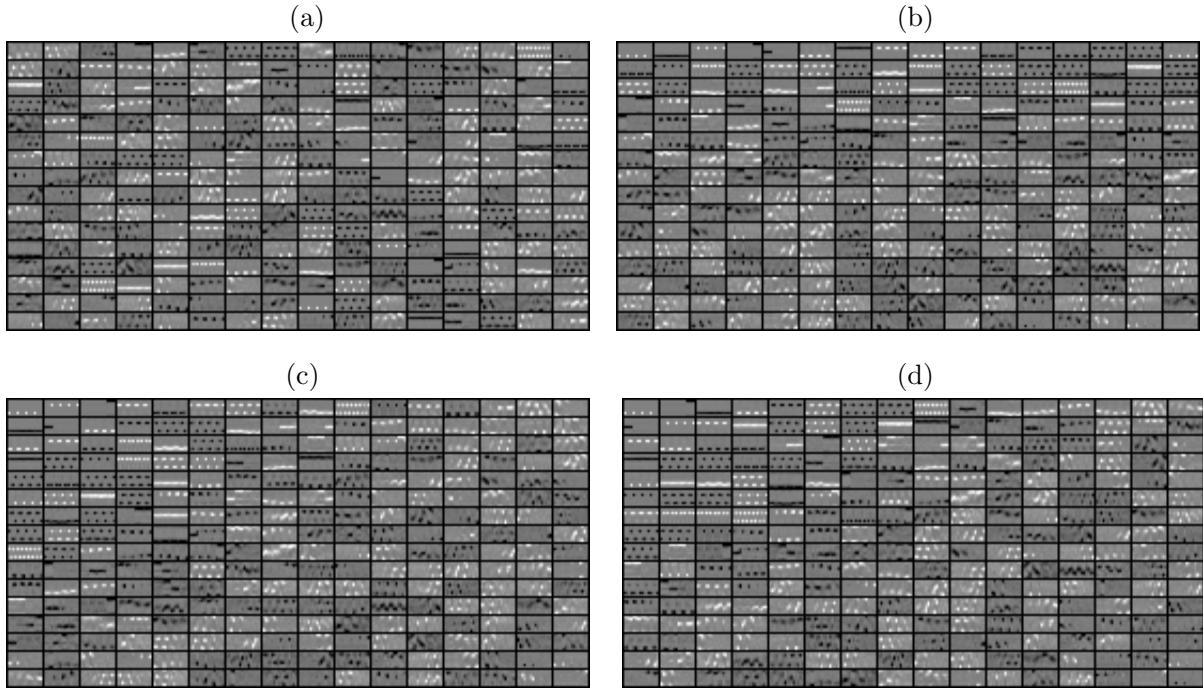


图 6.14: Visualization of a dictionary of size 256, learnt from 50,000 128-dimensional SIFT features. (a) The orderless display. (b) and (c) are the sorted results by BS in row-by-row and zigzag arrangement schemes, respectively. (d) The result of the baseline shown in zigzag arrangement.

is difficult to tell the difference between the results of BS and baseline intuitively because SIFT feature is not visually perceivable.

These experiments show that BS can find the intrinsic order among the bases of an over-complete dictionary.

**Data Compression:** DCT and Discrete Wavelet Transform (DWT) have been widely used in data compression thanks to an important property of DCT and DWT. Namely, the high frequency coefficients can be truncated and the data can still be well approximated by the low frequency ones. In this section we show that over-complete dictionaries also have such a property in the sense of generalized frequency.

We prepare an over-complete dictionary learnt from raw image patches as in Sec. 6.5.3.2. Then we compute the average reconstruction errors using a part of the bases. The bases are chosen in three ways:

1. Randomly choose  $k$  bases from the dictionary.
2. Choose the  $k$  lowest frequency bases of the dictionary.

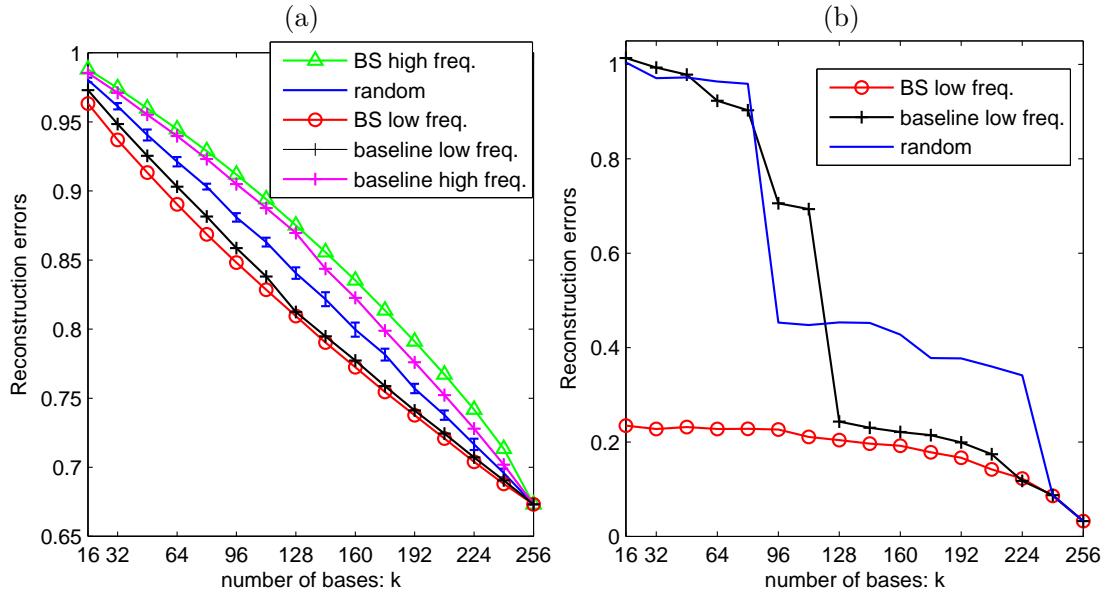


图 6.15: Average reconstruction errors of different choices of  $k$  bases. (a) Average reconstruction errors on training samples. That the reconstruction error is not zero when all the bases are used is because we have used a noise tolerance parameter in dictionary learning (see (6.81)). (b) Average reconstruction errors of reconstructing test images by truncated  $k$  bases.

### 3. Choose the $k$ highest frequency bases of the dictionary.

The average reconstruction errors on the training samples are presented in Fig. 6.15 (a). One can observe that using the  $k$  lowest frequency bases always leads to the best reconstruction accuracy, while using the  $k$  highest frequency bases results in the worst accuracy, and using random  $k$  bases is between these two choices. Moreover, the reconstruction errors using  $k$  lowest frequency bases defined by BS are consistently lower than those by baseline. Although the difference between BS and baseline on the training samples are not salient, their difference is drastic on the testing samples (Fig. 6.15 (b)), especially when  $k$  is small.

For visual comparison, we reconstruct an image with the dictionary in Fig. 6.13 and display in Fig. 6.16 the images reconstructed by using  $k$  lowest frequency bases. Clearly, using the  $k$  lowest frequency bases defined by BS yields much better visual quality than the baseline, which is only slightly better than using  $k$  randomly chosen bases. This is because BS can result in better average spectra that facilitate data compression. Namely, high frequencies have lower representation magnitudes, as having been shown in Fig. 6.12.

**练习145.** Use image Lena and K-SVD to learn a 256-atom dictionary for  $8 \times 8$  patches.

*Order the dictionary from “low frequency” to “high frequency” and rearrange in 2D in the zigzag manner.*

**Algorithm 7** The K-SVD dictionary-learning algorithm.

---

**Initialize:**  $k = 0$ , build  $\mathbf{A}_{(0)} \in \mathbb{R}^{n \times m}$ , either by using random entries, or using  $m$  randomly chosen examples, normalize the columns of  $\mathbf{A}_{(0)}$ .

**while** If the change in  $\|\mathbf{Y} - \mathbf{A}_{(k)}\mathbf{X}_{(k)}\|_F^2$  is not small enough **do**

**Sparse Coding:** Use a pursuit algorithm to approximate the solution of

$$\hat{\mathbf{x}}_i = \underset{\mathbf{x}}{\operatorname{argmin}} \|\mathbf{y}_i - \mathbf{A}_{(k-1)}\mathbf{x}\|^2, \quad s.t. \quad \|\mathbf{x}\|_0 \leq k_0,$$

obtaining sparse representations  $\hat{\mathbf{x}}_i$  for  $1 \leq i \leq M$ . These form the matrix  $\mathbf{X}_{(k)}$ .

**K-SVD Dictionary-Update:** Use the following procedure to update the columns of the dictionary and obtain  $\mathbf{A}_{(k)}$ :

Repeat for  $j_0 = 1, 2, \dots, m$ ,

1. Define the group of examples that use the atom  $\mathbf{a}_{j_0}$ ,

$$\Omega_{j_0} = \{i | 1 \leq i \leq M, \mathbf{X}_{(k)}[j_0, i] \neq 0\}.$$

2. Compute the residual matrix

$$\mathbf{E}_{j_0} = \mathbf{Y} - \sum_{j \neq j_0} \mathbf{a}_j \mathbf{x}_j^T,$$

where  $\mathbf{x}_j^T$  are the  $j$ 'th rows in the matrix  $\mathbf{X}_{(k)}$ .

3. Restrict  $\mathbf{E}_{j_0}$  by choosing only the columns corresponding to  $\Omega_{j_0}$ , and obtain  $\mathbf{E}_{j_0}^R$ .
4. Apply SVD on  $\mathbf{E}_{j_0}^R = \mathbf{U}\Sigma\mathbf{V}^T$ . Update the dictionary atom  $\mathbf{a}_{j_0} = \mathbf{u}_1$ , and the representations by  $\mathbf{x}_{j_0}^R = \Sigma(1, 1) \cdot \mathbf{v}_1$ .
5. **Update  $k$ :** Increase  $k$  by 1.

**end while**

**Ensure:** Output  $\mathbf{A}_{(k)}$ .

---

**Algorithm 8** Bases Sorting (BS) algorithm

---

- 1: **Input:** training data matrix  $X = (\mathbf{x}_1, \dots, \mathbf{x}_n)$ , dictionary  $U = (\mathbf{u}_1, \dots, \mathbf{u}_m)$ , and parameter  $\epsilon > 0$ .
- 2: Initialize the accumulation buffer:  $\mathbf{s} = (0, \dots, 0)$ .
- 3: **for**  $i = 1$  to  $n$  **do**
- 4:     Calculate the sparsest representation of  $\mathbf{x}_i$  by solving

$$\mathbf{v}_i = \arg \min_{\mathbf{v}} \|\mathbf{v}\|_1, \quad s.t. \quad \|\mathbf{x}_i - U\mathbf{v}\|_2^2 < \epsilon. \quad (6.81)$$

- 5:     Identify the support of sparse representation vector  $\mathbf{v}_i$  and the set of active bases:

$$\begin{aligned} I^{(i)} &= \{j \mid v_{ij} \neq 0, j = 1, \dots, m\}, \\ U^{(\mathbf{x}_i)} &= \{\mathbf{u}_j \mid j \in I^{(i)}\}. \end{aligned} \quad (6.82)$$

- 6:     Compute the magnitude vector  $\mathbf{a}_i$  of active coefficients, i.e.,  $a_{ij} = |v_{ij}|$ .
- 7:     Sort the activated bases in  $U^{(\mathbf{x}_i)}$  according to the descending order of the magnitude vector  $\mathbf{a}_i$ :

$$\pi(U^{(\mathbf{x}_i)}) = \text{sort}(I^{(i)}, \mathbf{a}_i). \quad (6.83)$$

- 8:     Update the accumulation buffer:  $s_j = s_j + 1$  for all  $j \in I^{(i)}$ .

9: **end for**

- 10: Compute the averaged order by

$$\pi(U) = \sum_{i=1}^n \pi(U^{(\mathbf{x}_i)}) ./ \mathbf{s}, \quad (6.84)$$

where  $./$  is element-wise division.

- 11: Sort  $U$  in an ascending order of  $\pi(U)$ .
-

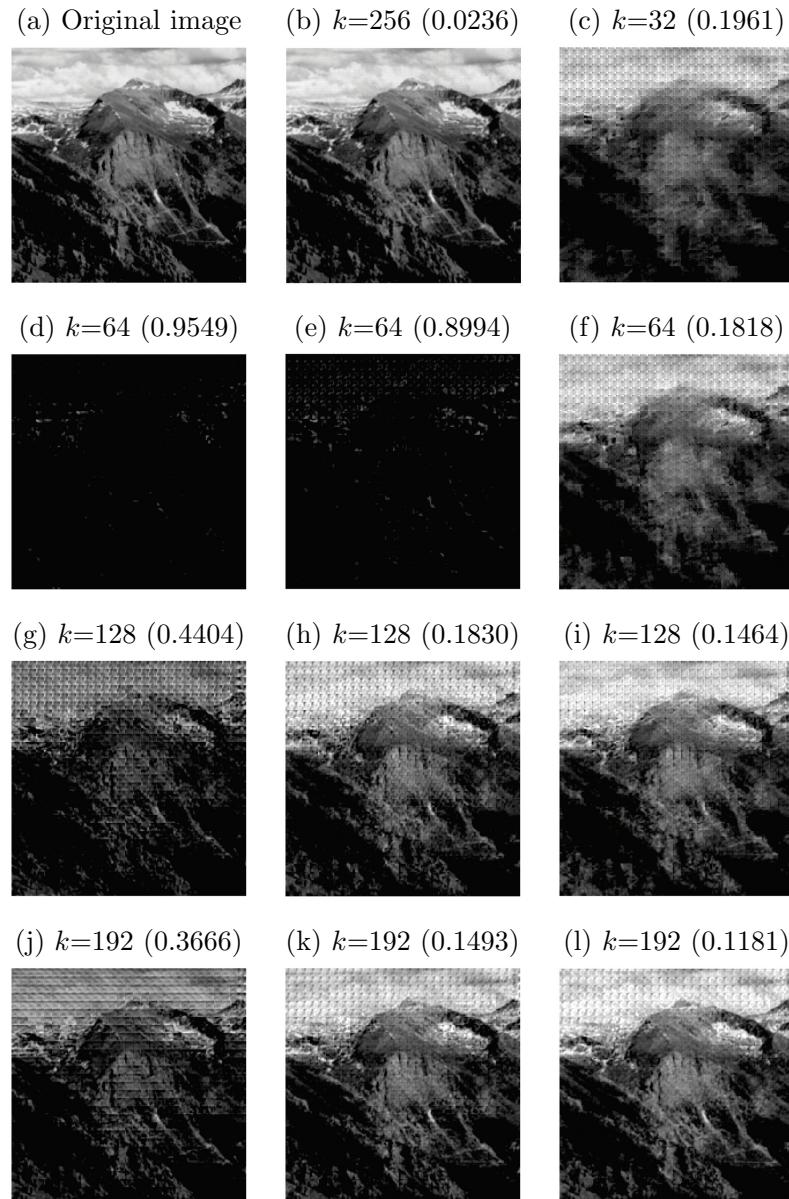


图 6.16: Reconstructed images by using truncated  $k$  bases. The values in brackets are the reconstruction errors. (a) The original image. (b) Reconstructed by using all 256 bases. (d), (g), and (j) are reconstructed by using  $k$  randomly chosen bases. (e), (h), and (k) are reconstructed by using the  $k$  lowest frequency bases defined by the baseline method. (c), (f), (i), and (l) are reconstructed by the  $k$  lowest frequency bases defined by BS.



## 第七章 凸分析简介

(本章内容取自[14], 从[7]可以取到更多和凸优化有关的知识)

### 7.1 基本概念

凸集、凸函数、次梯度、集合的凸包、函数的凸包、法锥、切锥

**定义146.** 线性空间 $X$ 的子集 $C$ 称为凸集, 如果:

$$\alpha x + (1 - \alpha)y \in C, \quad \forall x, y \in C, \alpha \in [0, 1]. \quad (7.1)$$

空集定义为凸集。凸集的例子:  $\{\mathbf{x}|\mathbf{x} \geq 0\}$ ,  $\{\mathbf{x}|\|\mathbf{x}\| \leq r\}$ , 其中 $\|\cdot\|$ 是任意向量范数,  $\mathbb{S}_+ = \{\mathbf{X}|\mathbf{X} \succcurlyeq 0\}$ .

**定理147.** 凸集有如下基本性质:

1. 任意多个凸集的交还是凸集。
2. 如果 $C_1$ 、 $C_2$ 是凸集, 则

$$C_1 + C_2 \equiv \{x_1 + x_2 | x_1 \in C_1, x_2 \in C_2\}$$

也是凸集。

3. 凸集的闭包和内部也是凸集。
4. 凸集经仿射变换后还是凸集; 凸集的仿射变换的原像也是凸集。

*Proof.*

□

**定义148.** 线性空间 $X$ 的子集 $C$ 称为锥, 如果:

$$\lambda x \in C, \quad \forall x \in C, \lambda > 0. \quad (7.2)$$

**定义149.** 设 $X$ 为点集,  $x_i \in X$ ,  $i = 1, \dots, m$ , 则

$$\sum_{i=1}^m \alpha_i x_i, \quad \alpha_i \geq 0, \quad \sum_{i=1}^m \alpha_i = 1, \quad (7.3)$$

$$\sum_{i=1}^m \alpha_i x_i, \quad \sum_{i=1}^m \alpha_i = 1, \quad (7.4)$$

$$\sum_{i=1}^m \alpha_i x_i, \quad \alpha_i \geq 0 \quad (7.5)$$

分别称为 $\{x_i\}$ 的凸组合、仿射组合和非负组合。

**定义150.** 设 $X$ 为点集，则 $X$ 的凸包为包含 $X$ 的所有凸集的交。它由所有 $X$ 的点的凸组合构成，记作 $\text{conv}(X)$ 。

**定义151.** 设 $X$ 为点集，则 $X$ 的仿射包为包含 $X$ 的所有仿射空间的交。它由所有 $X$ 的点的仿射组合构成，记作 $\text{aff}(X)$ 。 $\text{aff}(X)$ 的维数称为 $X$ 的维数。

**定义152.** 设 $X$ 为点集，由所有 $X$ 的点的非负组合构成的集合称为 $X$ 生成的锥，记作 $\text{cone}(X)$ 。

**定义153.** 设 $X \subset \mathbb{R}^m$ 为点集， $x \in X$ ， $B_r(x)$ 为中心在 $x$ 、半径为 $r$ 的 $m$ 维球。如果存在 $r > 0$ 使得 $B_r(x) \cap \text{aff}(X) \subset X$ ，则称 $x$ 为 $X$ 的相对内点(*relative interior*)。 $X$ 的相对内点集记为 $\text{ri}(X)$ 。

**定义154.** 设 $C$ 为凸集， $f : C \rightarrow \mathbb{R}$ 称为凸函数，如果：

$$f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha)f(y), \quad \forall x, y \in C, \alpha \in [0, 1]. \quad (7.6)$$

函数 $g$ 称为凹函数，如果 $-g$ 是凸函数。

如果 $f$ 二次可微，则 $f$ 凸等价于其Hessian矩阵非负定： $H_f \succcurlyeq 0$ .

严格凸：

$$f(\alpha x + (1 - \alpha)y) < \alpha f(x) + (1 - \alpha)f(y), \quad \forall \alpha \in (0, 1).$$

强凸：

$$f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha)f(y) - \frac{\nu}{2}\alpha(1 - \alpha)\|x - y\|^2, \quad \forall \alpha \in [0, 1].$$

$\nu > 0$ 称为强凸系数(modulus)。如果 $f$ 二次可微，则 $f$ 强凸等价于其Hessian矩阵 $H_f \succ \nu I$ .

**定理155.** 凸函数有如下基本性质：

1. 凸函数都是连续函数，且几乎处处可微(Rademacher定理)。
2. 设对任意 $i \in I$ ， $f_i$ 都是凸函数，则 $f(x) = \sup_{i \in I} f_i(x)$ 也是凸函数。
3. 设 $f$ 是凸函数， $h$ 是线性变换，则 $g = f \circ h$ 也是凸函数。
4. 如果 $f_i$ 是凸函数， $\lambda_i > 0$ ， $i = 1, \dots, m$ ，则 $f(x) = \sum_{i=1}^m \lambda_i f_i(x)$ 也是凸函数。
5. (Jensen不等式) 设 $x_i \in C$ ， $\alpha_i \geq 0$ ， $i = 1, \dots, m$ ， $\sum_{i=1}^m \alpha_i = 1$ ，则

$$f\left(\sum_{i=1}^m \alpha_i x_i\right) \leq \sum_{i=1}^m \alpha_i f(x_i). \quad (7.7)$$

*Proof.*

□

**定义156.** 函数 $f(x)$ 沿 $y$ 的方向导数定义为：

$$f'(x; y) = \lim_{\alpha \downarrow 0} \frac{f(x + \alpha y) - f(x)}{\alpha}, \quad (7.8)$$

如果右端极限存在。

**定义157.** 设凸函数 $f(x)$ 定义在凸集 $C$ 上。 $f$ 在 $x$ 处的次梯度(*subgradient*)定义为：

$$\partial f(x) = \{g | f(y) \geq f(x) + \langle g, y - x \rangle, \forall y \in C\}. \quad (7.9)$$

类似地，可以定义凹函数的超梯度(*supergradient*)。

**定理158.** 凸函数的次梯度有如下性质：

1. 凸函数在任一点的次梯度总是一个非空的凸紧集。

2. 如果 $f_i : C_i \rightarrow \mathbb{R}$ 是凸函数， $i = 1, \dots, m$ ， $f = \sum_{i=1}^m f_i$ ，则

$$\partial f(x) = \sum_{i=1}^m \partial f_i(x), \quad \forall x \in \cap_{i=1}^m C_i.$$

3. 如果 $f : \mathbb{R}^m \rightarrow \mathbb{R}$ 是凸函数， $A$ 是 $m \times n$ 阶矩阵，则 $F(x) = f(Ax)$ 的次梯度为：

$$\partial F(x) = A^T \partial f(Ax) = \{A^T g | g \in \partial f(Ax)\}.$$

4. 如果 $f : \mathbb{R}^m \rightarrow \mathbb{R}$ 是凸函数， $g : \mathbb{R} \rightarrow \mathbb{R}$ 是单调非降的凸函数，则 $F(x) = g(f(x))$ 也是凸函数，其次梯度为：

$$\partial F(x) = \partial g(f(x)) \partial f(x) = \{g_1 g_2 | g_1 \in \partial g(f(x)), g_2 \in \partial f(x)\}.$$

*Proof.*

□

$$\langle g_1 - g_2, x_1 - x_2 \rangle \geq 0, \quad \forall g_i \in \partial f(x_i), i = 1, 2.$$

设 $f$ 强凸，则上式可加强为：

$$\langle g_1 - g_2, x_1 - x_2 \rangle \geq \nu \|x_1 - x_2\|^2, \quad \forall g_i \in \partial f(x_i), i = 1, 2.$$

**定理159.** 设 $f(x)$ 是凸函数，则 $f$ 沿 $y$ 的方向导数总是存在：

$$f'(x; y) = \max_{g \in \partial f(x)} \langle g, y \rangle. \quad (7.10)$$

*Proof.*

**定理160.** (*Danskin定理*) 设 $Z$ 是 $\mathbb{R}^m$ 上的紧子集,  $\phi : \mathbb{R}^n \times Z \rightarrow \mathbb{R}$ 满足: 对每一 $z \in Z$ ,  $\phi(\cdot, z) : \mathbb{R}^n \rightarrow \mathbb{R}$ 都是凸函数。定义函数 $f(x) = \max_{z \in Z} \phi(x, z)$ ,  $Z(x) = \{z | \phi(x, z) = f(x)\}$ , 则

1.  $f(x)$ 是凸函数, 其沿 $y$ 的方向导数为

$$f'(x; y) = \max_{z \in Z(x)} \phi'(x, z; y). \quad (7.11)$$

2. 特别地, 如果 $Z(x)$ 只有一个点 $\bar{z}$ , 且 $\phi(x, \bar{z})$ 在 $x$ 处可微, 则 $f(x)$ 在 $x$ 处可微, 且 $\text{grad}f(x) = \text{grad}_x \phi(x, \bar{z})$ 。

3.

$$\partial f(x) = \text{conv}\{\partial_x \phi(x, z) | z \in Z(x)\}. \quad (7.12)$$

*Proof.*

**定理161.**

$$\partial\|\mathbf{X}\|_* = \{\mathbf{U}\mathbf{V}^T + \mathbf{U}^\perp\mathbf{W}(\mathbf{V}^\perp)^T | \|\mathbf{W}\|_2 \leq 1\}, \quad (7.13)$$

where  $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$  is the skinny SVD of  $\mathbf{X}$  and  $\mathbf{U}^\perp$  and  $\mathbf{V}^\perp$  are the orthogonal complement of  $\mathbf{U}$  and  $\mathbf{V}$ , respectively.

Proof. see [135].

**例162.** Compute  $\partial\|\mathbf{X}\|_2$ .

We first recall that  $\|\mathbf{X}\|_2 = \max_{\|\mathbf{u}\|=\|\mathbf{v}\|=1} \mathbf{u}^T \mathbf{X} \mathbf{v} = \max_{\|\mathbf{u}\|=\|\mathbf{v}\|=1} \text{tr}(\mathbf{X} \mathbf{v} \mathbf{u}^T)$ . Then by *Danskin theorem*,

$$\partial\|\mathbf{X}\|_2 = \text{conv}\{\mathbf{u}\mathbf{v}^T | \mathbf{u}^T \mathbf{X} \mathbf{v} = \|\mathbf{X}\|_2, \|\mathbf{u}\| = \|\mathbf{v}\| = 1\}.$$

If  $\mathbf{X} \neq \mathbf{0}$ , then the  $(\mathbf{u}, \mathbf{v})$  that satisfy  $\mathbf{u}^T \mathbf{X} \mathbf{v} = \|\mathbf{X}\|_2$  are simply the unit left and right singular vectors associated to the largest singular value  $\sigma_1$  of  $\mathbf{X}$  and  $\mathbf{u} = \sigma_1^{-1} \mathbf{X} \mathbf{v}$ .

Suppose the dimension of the principal singular subspace is  $d$ , then there are orthonormal right singular vectors  $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_d$ , and left singular vectors  $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_d$ , such that  $\mathbf{u}_i = \sigma_1^{-1} \mathbf{X} \mathbf{v}_i$ ,  $i = 1, 2, \dots, d$ .

Then every pair of unit left and right singular vectors associated to the largest singular value  $\sigma_1$  of  $\mathbf{X}$  can be written as  $\mathbf{u} = \mathbf{U}\alpha$  and  $\mathbf{v} = \mathbf{V}\alpha$ , where  $\mathbf{U} = (\mathbf{u}_1, \dots, \mathbf{u}_d)$ ,  $\mathbf{V} = (\mathbf{v}_1, \dots, \mathbf{v}_d)$ ,  $\|\alpha\| = 1$ . So

$$\partial\|\mathbf{X}\|_2 = \text{conv}\{\mathbf{U}\alpha\alpha^T \mathbf{V}^T | \|\alpha\| = 1\} = \{\mathbf{U}\mathbf{W}\mathbf{V}^T | \mathbf{W} \succcurlyeq \mathbf{0}, \text{tr}\mathbf{W} = 1\}. \quad (7.14)$$

If  $\mathbf{X} = \mathbf{0}$ , then

$$\partial\|\mathbf{X}\|_2 = \text{conv}\{\mathbf{u}\mathbf{v}^T \mid \|\mathbf{u}\| = \|\mathbf{v}\| = 1\} = \{\mathbf{W} \mid \|\mathbf{W}\|_* \leq 1\}. \quad (7.15)$$

**定义163.** 给定  $\mathbb{R}^n$  的子集  $X$ , 它的极锥  $X^*$  定义为

$$X^* = \{y \mid \langle y, x \rangle \leq 0, \forall x \in X\}. \quad (7.16)$$

**定义164.** 给定  $\mathbb{R}^n$  的子集  $X$ ,  $x \in X$ . 称  $y$  为  $X$  在  $x$  处的切线, 如果  $y = 0$  或存在点列  $\{x_k\} \subset X$  使得  $x_k \neq x$ ,  $x_k \rightarrow x$ , 且  $\frac{x_k - x}{\|x_k - x\|} \rightarrow \frac{y}{\|y\|}$ .  $x$  处所有切线的集合称为  $X$  在  $x$  处的切锥, 记作  $T_X(x)$ .

**定义165.** 给定  $\mathbb{R}^n$  的子集  $X$ ,  $x \in X$ . 称  $z$  为  $X$  在  $x$  处的法线, 如果  $y = 0$  或存在点列  $\{x_k\} \subset X$  和  $\{z_k\}$  使得  $x_k \rightarrow x$ ,  $z_k \rightarrow z$ , 且  $z_k \in T_X(x_k)^*$ .  $x$  处所有切线的集合称为  $X$  在  $x$  处的法锥, 记作  $N_X(x)$ . 称  $x$  是  $X$  的正则 (regular) 点, 如果  $N_X(x) = T_X(x)^*$ .

**定理166.** 设  $C$  是凸集, 则  $\forall x \in C$ ,

$$N_C(x) = \{y \mid \langle y, z - x \rangle \leq 0, \forall z \in C\}, \quad (7.17)$$

$$T_C(x) = N_C(x)^*, \quad (7.18)$$

$$T_C(x)^* = N_C. \quad (7.19)$$

*Proof.*

□

**定理167.** 设  $C_1$ 、 $C_2$  是凸集,  $\text{ri}(C_1) \cap \text{ri}(C_2) \neq \emptyset$ , 则  $\forall x \in C_1 \cap C_2$ ,

$$T_{C_1 \cap C_2}(x) = T_{C_1}(x) \cap T_{C_2}(x), \quad (7.20)$$

$$N_{C_1 \cap C_2}(x) = N_{C_1}(x) + N_{C_2}(x). \quad (7.21)$$

*Proof.*

□

## 7.2 共轭函数(conjugate function)

**定义168.** 函数  $f(x)$  在  $C$  上的凸包络 (convex envelope) 是满足  $g(x) \leq f(x)$ ,  $\forall x \in C$  的最大的凸函数  $g$ .

**定义169.** 函数  $f^*(y) = \sup_{x \in C} \{\langle x, y \rangle - f(x)\}$  称为  $f$  的共轭函数。

**定理170.** 共轭函数有如下性质:

1.  $f^*$  是凸函数;

2.  $f(x) + f^*(y) \geq \langle x, y \rangle$  (*Fenchel不等式*), 它是如下不等式对非二次函数的推广:

$$\frac{1}{2}\|x\|^2 + \frac{1}{2}\|y\|^2 \geq \langle x, y \rangle.$$

3.  $f^{**}$ 是 $f^*$ 的凸包函数。特别地, 如果 $f$ 是凸函数, 则 $f^{**} = f$  (*Fenchel定理*);

4. 如果 $f$ 凸且闭, 则

$$y \in \partial f(x) \Leftrightarrow x \in \partial f^*(y) \Leftrightarrow f(x) + f^*y = \langle x, y \rangle.$$

共轭函数的例子:

$$1. f(x) = p^{-1}|x|^p, 1 < p < +\infty; f^*(y) = q^{-1}|y|^q, \frac{1}{p} + \frac{1}{q} = 1.$$

**命题171.** (pages 56-59 of [47]) *The convex envelope of  $\text{rank}(\mathbf{X})$  on  $C = \{\mathbf{X} | \|\mathbf{X}\|_2 \leq 1\}$  is  $\|\mathbf{X}\|_*$ .*

### 7.3 Envelope Function and Proximal Mapping

(page 160 of [14])

**定义172.** Let  $f : \mathbb{R}^n \rightarrow (-\infty, +\infty]$  be a closed proper function. For a scalar  $c > 0$ , define the corresponding envelope function  $E_c f$  and the proximal mapping  $P_c f$  by

$$\begin{aligned} E_c f(\mathbf{x}) &= \inf_{\mathbf{w}} \left\{ f(\mathbf{w}) + \frac{1}{2c} \|\mathbf{w} - \mathbf{x}\|^2 \right\}, \\ P_c f(\mathbf{x}) &= \underset{\mathbf{w}}{\operatorname{argmin}} \left\{ f(\mathbf{w}) + \frac{1}{2c} \|\mathbf{w} - \mathbf{x}\|^2 \right\}. \end{aligned} \quad (7.22)$$

Example:  $f(\mathbf{x}) = \chi_C(\mathbf{x})$ ,  $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \mathbf{A}\mathbf{x} + \mathbf{b}^T \mathbf{x} + \mathbf{c}$ ,  $f(\mathbf{x}) = \|\mathbf{x}\|_1$ ,  $f(\mathbf{x}) = \sum_{i=1}^n \log x_i$ ,  $f(\mathbf{X}) = \|\mathbf{X}\|_*$  (see [20]).

**备注173.** 1. The envelope function  $E_c f$  is an underestimate of the function  $f$ , i.e.,  $E_c f(\mathbf{x}) \leq f(\mathbf{x}), \forall \mathbf{x} \in \mathbb{R}^n$ . Furthermore,  $E_c f$  is a real-valued continuous function, whereas  $f$  itself may only be extended real-valued and lower semi-continuous.

2. It is obvious that  $\mathbf{u} = P_c f(\mathbf{x}) \Leftrightarrow \mathbf{x} - \mathbf{u} \in \partial f(\mathbf{u})$ .

**定理174.** For any function  $f$ ,  $P_c f(\mathbf{x})$  is a monotonic function of  $\mathbf{x}$  in the sense that:

$$\langle \mathbf{x}_1 - \mathbf{x}_2, \mathbf{y}_1 - \mathbf{y}_2 \rangle \geq 0, \quad \forall \mathbf{y}_i \in P_c f(\mathbf{x}_i), i = 1, 2. \quad (7.23)$$

*Proof.* By the definition of  $\mathbf{y}_i$ ,

$$\begin{aligned} f(\mathbf{y}_1) + \frac{1}{2c} \|\mathbf{y}_1 - \mathbf{x}_1\|^2 &\leq f(\mathbf{y}_2) + \frac{1}{2c} \|\mathbf{y}_2 - \mathbf{x}_1\|^2, \\ f(\mathbf{y}_2) + \frac{1}{2c} \|\mathbf{y}_2 - \mathbf{x}_2\|^2 &\leq f(\mathbf{y}_1) + \frac{1}{2c} \|\mathbf{y}_1 - \mathbf{x}_2\|^2. \end{aligned} \quad (7.24)$$

Adding them together yields (7.23).  $\square$

**命题175.** Let  $f : \mathbb{R}^n \rightarrow (-\infty, +\infty]$  be a closed proper convex function and  $c > 0$ . The envelope function  $E_c f$  is convex and smooth and its gradient is given by

$$\nabla E_c f(\mathbf{x}) = \frac{1}{c}(\mathbf{x} - P_c f(\mathbf{x})). \quad (7.25)$$

**备注176.** The envelope function  $E_c f$  is smooth, regardless of whether  $f$  is smooth.

**命题177.** Let  $f : \mathbb{R}^n \rightarrow (-\infty, +\infty]$  be a closed proper convex function and  $c > 0$ . The proximal mapping  $P_c f$  is single-valued and is continuous in the sense that  $P_c f(x) \rightarrow P_{c^*} f(\mathbf{x}^*)$  whenever  $(\mathbf{x}, c) \rightarrow (\mathbf{x}^*, c^*)$ , with  $c^* > 0$ .

**定理178.** (Moreau Decomposition)  $\mathbf{x} = P_c f(\mathbf{x}) + P_c f^*(\mathbf{x})$ ,  $\forall \mathbf{x}$ .

**练习179.**

1. 证明下列函数是凸的：

- $f_1(x_1, \dots, x_n) = \begin{cases} -(x_1 x_2 \cdots x_n)^{1/n}, & \text{if } x_1 > 0, \dots, x_n > 0, \\ +\infty, & \text{otherwise.} \end{cases}$
- $f_2(\mathbf{x}) = \ln(e^{x_1} + \cdots + e^{x_n})$ .
- $f_3(\mathbf{x}) = \|\mathbf{x}\|_2^p$ , 其中  $p \geq 1$ .
- $f_4(\mathbf{x}) = e^{\beta \mathbf{x}^T \mathbf{A} \mathbf{x}}$ , 其中  $\mathbf{A}$  为对称半正定矩阵,  $\beta > 0$ .

2. 证明：如果  $\alpha_i \geq 0$ ,  $x_i > 0$ ,  $\sum_{i=1}^n \alpha_i = 1$ , 那么

$$x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n} \leq \alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_n x_n,$$

且等号成立当且仅当  $x_1 = x_2 = \cdots = x_n$ .

3. 证明定理 158-2.

4. 证明：方向导数  $f'(x; y)$  关于  $y$  是凸函数。

5. 求向量 2-范数的次梯度。

6. 证明定理 170-1.

7. 设  $A \succ 0$ , 证明:  $\frac{1}{2}x^T Ax + b^T x + \frac{1}{2}(y - b)^T A^{-1}(y - b) \geq \langle x, y \rangle$ .

8. 求矩阵  $(2, 0)$  范数:  $\|(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)\|_{2,0} = \#\{j | \|\mathbf{x}_j\|_2 \neq 0\}$  在  $C = \{\mathbf{X} | \|\mathbf{X}\|_{2,1} \leq 1\}$  上的凸包络。

练习180.

1. Prove the last equalities in (7.14) and (7.15).
2. Find the envelope function and proximal mapping of  $f(\mathbf{x}) = \|\mathbf{x}\|_2$ .
3. Prove that if  $f(\mathbf{x}) = g(\lambda\mathbf{x} + \mathbf{a})$ , where  $\lambda \neq 0$ , then  $P_c f(\mathbf{x}) = \frac{1}{\lambda} [P_c(\lambda^2 g)(\lambda\mathbf{x} + \mathbf{a}) - \mathbf{a}]$ .
4. Prove that if  $f(\mathbf{x}) = \lambda g(\mathbf{x}/\lambda)$ , where  $\lambda > 0$ , then  $P_c f(\mathbf{x}) = \lambda P_c(\lambda^{-1} g)(\mathbf{x}/\lambda)$ .
5. Let  $f(\mathbf{X})$  be the characteristic function on the set  $S_+$  of psd matrices. Compute its proximal mapping.
6. Use Moreau decomposition to prove that  $\mathbf{x} = P_L(\mathbf{x}) + P_{L^\perp}(\mathbf{x})$ , where  $L$  is a subspace and  $L^\perp$  is its orthogonal complement.

## 第八章 一阶优化方法

(The materials are based on [1].)

我们先讨论目标函数是可微的情形，再讨论目标函数是凸的情形。

### 8.1 函数的强凸性与误差估计[18]

假定函数 $f$ 定义在 $S$ 上，二次可微且强凸，即存在 $m > 0$ 使得：

$$\text{grad}^2 f(x) \succeq lI, \quad \forall x \in S. \quad (8.1)$$

任给 $y, x \in S$ ，由带余量的Taylor定理，在线段 $xy$ 上存在 $z$ ，使得：

$$f(y) = f(x) + [\text{grad}f(x)]^T(y - x) + \frac{1}{2}(y - x)^T \text{grad}^2 f(z)(y - x). \quad (8.2)$$

于是

$$f(y) \geq f(x) + [\text{grad}f(x)]^T(y - x) + \frac{l}{2}\|y - x\|^2. \quad (8.3)$$

关于 $y$ 极小化上式两边得：

$$f^* \geq f(x) - \frac{1}{2l}\|\text{grad}f(x)\|^2.$$

于是

$$f(x) \leq f^* + \frac{1}{2l}\|\text{grad}f(x)\|^2. \quad (8.4)$$

在(8.3)里取 $y = x^*$ ，则

$$\begin{aligned} f^* &\geq f(x) + [\text{grad}f(x)]^T(x^* - x) + \frac{l}{2}\|x^* - x\|^2 \\ &\geq f(x) - \|\text{grad}f(x)\|\|x^* - x\| + \frac{l}{2}\|x^* - x\|^2 \end{aligned}$$

于是

$$\|x^* - x\| \leq \frac{2\|\text{grad}f(x)\|}{l}. \quad (8.5)$$

假设 $f$ 还满足：

$$\text{grad}^2 f(x) \preceq LI, \quad \forall x \in S, \quad (8.6)$$

则

$$f(y) \leq f(x) + [\text{grad}f(x)]^T(y - x) + \frac{L}{2}\|y - x\|^2. \quad (8.7)$$

同样，极小化上式两边可得

$$f(x) \geq f^* + \frac{1}{2L}\|\text{grad}f(x)\|^2. \quad (8.8)$$

值得一提的是，不等式(8.7)成立并不需要 $f$ 二阶可导。事实上我们有[13]：

**定理181.** 如果  $f$  Lipschitz一阶可导:

$$\|\text{grad}f(x) - \text{grad}f(y)\| \leq L\|x - y\|, \quad (8.9)$$

则不等式(8.7)成立。

*Proof.* 首先, 我们有

$$f(y) - f(x) = \int_0^1 \frac{df(x + t(y - x))}{dt} dt \quad (8.10)$$

$$= \int_0^1 [\text{grad}f(x + t(y - x))]^T (y - x) dt. \quad (8.11)$$

所以

$$f(y) - f(x) - [\text{grad}f(x)]^T (y - x) \quad (8.12)$$

$$= \int_0^1 [\text{grad}f(x + t(y - x)) - \text{grad}f(x)]^T (y - x) dt \quad (8.13)$$

$$\leq \int_0^1 \|\text{grad}f(x + t(y - x)) - \text{grad}f(x)\| \|y - x\| dt \quad (8.14)$$

$$\leq \int_0^1 Lt \|y - x\|^2 dt \quad (8.15)$$

$$= \frac{L}{2} \|y - x\|^2. \quad (8.16)$$

□

## 8.2 下降法 (目标函数可微情形)

---

### Algorithm 9 下降法 (一般形式)

- 
- 1: 初始化  $x_0$ ,  $k = 0$ .
  - 2: **while** 停止条件不满足 **do**
  - 3:   确定下降方向  $\Delta x_k$ ;
  - 4:   沿  $\Delta x_k$  作线搜索, 确定步长  $t_k > 0$ ;
  - 5:   更新:  $x_{k+1} = x_k + t_k \Delta x_k$ ,  $k \leftarrow k + 1$ .
  - 6: **end while**
- 

下降法的一般形式见算法9。下降法指的是每一步都得到更好的解:  $f(x_{k+1}) < f(x_k)$ , 除非  $x_k$  就是极小点。所谓“下降方向”就是任何满足  $[\text{grad}f(x_k)]^T \Delta x_k < 0$  的  $\Delta x_k$ 。由 Taylor 展开易见只要  $\Delta x_k$  是下降方向, 就必然存在步长  $t_k > 0$  使得  $f(x_{k+1}) < f(x_k)$ 。

常用的下降方向是负梯度方向:  $\Delta x_k = -\text{grad}f(x_k)$ 。选定下降方向  $\Delta x_k$  后, 线搜索也有很多策略。最自然的是精确线搜索, 即  $t_k = \arg \min_t f(x_k + t \Delta x_k)$ 。但实际上精

确线搜索毫无必要，它需要在下降方向上搜索多次，所以如果计算目标函数值的计算量不是显著地小于计算梯度的计算量，则建议使用非精确线搜索。常用的一种非精确线策略是回溯（backtracing）线搜索，见算法10。

---

**Algorithm 10** 回溯线搜索

- 
- 1: 给定  $\alpha \in (0, 1)$ ,  $\beta \in (0, 1)$ .
  - 2: 初始化:  $t = t_0$ .
  - 3: **while**  $f(x_k + t\Delta x_k) > f(x_k) + \alpha t[\text{grad}f(x_k)]^T \Delta x_k$  **do**
  - 4:      $t = \beta t$ ;
  - 5: **end while**
- 

### 8.2.1 精确线搜索时的收敛性分析

假设  $f$  满足(8.1)和(8.6)，则

$$f(x_k - t\text{grad}f(x_k)) \leq f(x_k) - t\|\text{grad}f(x_k)\|^2 + \frac{Lt^2}{2}\|\text{grad}f(x_k)\|^2. \quad (8.17)$$

右端项在  $t = 1/L$  时取极小。于是

$$f(x_{k+1}) \leq f(x_k - (1/L)\text{grad}f(x_k)) \leq f(x_k) - \frac{1}{2L}\|\text{grad}f(x_k)\|^2. \quad (8.18)$$

由(8.4)，我们进一步得到

$$f(x_{k+1}) - f^* \leq f(x_k) - f^* - \frac{l}{L}(f(x_k) - f^*). \quad (8.19)$$

所以

$$f(x_{k+1}) - f^* \leq \left(1 - \frac{l}{L}\right)(f(x_k) - f^*). \quad (8.20)$$

因此算法线性收敛。 $L/l$  称为问题的条件数。此时我们也可以看出，由于凸性强的函数一般来说条件数也低，所以有利于迭代收敛。另一方面，从证明中可以得到，如果对  $L$  有一个估计，则无须做线搜索：步长取  $t_k \equiv 1/L$  即可。

### 8.2.2 回溯线搜索时的收敛性分析

根据  $t_k$  的取法，我们得到：

$$f(x_k - t_k\text{grad}f(x_k)) \leq f(x_k) - \alpha t_k\|\text{grad}f(x_k)\|^2, \quad (8.21)$$

$$f(x_k - (t_k/\beta)\text{grad}f(x_k)) > f(x_k) - \alpha(t_k/\beta)\|\text{grad}f(x_k)\|^2. \quad (8.22)$$

由(8.17)，

$$f(x_k - (t_k/\beta)\text{grad}f(x_k)) \leq f(x_k) - (t_k/\beta)\|\text{grad}f(x_k)\|^2 + \frac{L}{2}(t_k/\beta)^2\|\text{grad}f(x_k)\|^2. \quad (8.23)$$

由上式和(8.22)得到

$$-\alpha(t_k/\beta)\|\text{grad}f(x_k)\|^2 \leq -(t_k/\beta)\|\text{grad}f(x_k)\|^2 + \frac{L}{2}(t_k/\beta)^2\|\text{grad}f(x_k)\|^2. \quad (8.24)$$

因此

$$t_k \geq \frac{2(1-\alpha)\beta}{L}. \quad (8.25)$$

代入(8.21)并结合(8.4)得:

$$f(x_{k+1}) - f^* \leq f(x_k) - f^* - \frac{4\alpha(1-\alpha)\beta l}{L}(f(x_k) - f^*). \quad (8.26)$$

所以

$$f(x_{k+1}) - f^* \leq \left[1 - \frac{4\alpha(1-\alpha)\beta l}{L}\right](f(x_k) - f^*). \quad (8.27)$$

由于系数小于1, 算法线性收敛。

由(8.25)可见,  $t_0$ 最好至少为 $2(1-\alpha)/L$ 。比较(8.20) 和(8.27)似乎精确线搜索收敛更快, 事实正是如此。另外, (8.27)似乎也暗示 $\alpha$ 最优取值为0.5,  $\beta$ 要尽量靠近1, 但事实并非如此。实践中经常取 $\alpha \in [0.01, 0.3]$ 、 $\beta \in [0.1, 0.8]$ 。

### 8.3 凸优化

所谓凸优化指的是求解如下问题:

$$\begin{aligned} & \min_x f(x), \\ & s.t. \quad x \in C, \end{aligned} \quad (8.28)$$

其中 $f$ 是凸函数,  $C$ 是闭凸集。 $C$ 经常由等式和不等式定义。此时我们可以把问题(8.28)明确地写成:

$$\begin{aligned} & \min_x f(x), \\ & s.t. \quad g_i(x) = 0, \quad i = 1, \dots, m, \\ & \quad h_j(x) \leq 0, \quad j = 1, \dots, n, \end{aligned} \quad (8.29)$$

其中 $g_i$ 、 $h_i$ 均为凸函数, 使得约束集是凸集。

#### 8.3.1 凸优化的最优化条件[14]

**定理182.**  $x^*$ 是问题(8.28)的极小点的充分必要条件是:

$$0 \in \partial f(x^*) + T_C(x^*)^*, \quad (8.30)$$

其中 $T_C(x)$ 是 $C$ 在 $x$ 处的切锥, 而 $X^*$ 代表 $X$ 的极锥。

*Proof.*

(8.31)

(8.32)

□

条件(8.30)也可以写成：存在 $g \in \partial f(x^*)$ 使得 $\langle g, x - x^* \rangle \geq 0, \forall x \in C$ 。当 $f$ 可微且 $C$ 是全空间时，该条件就退化成我们熟悉的 $\text{grad}f(x^*) = 0$ 。

#### 8.4 Karush-Kuhn-Tucker(KKT)条件[13]

$x^*$ 是问题(8.29)的极小点的必要条件是它是可行解：

$$\begin{aligned} g_i(x^*) &= 0, \quad i = 1, \dots, m; \\ h_j(x^*) &\leq 0, \quad j = 1, \dots, n; \end{aligned}$$

且存在实数 $\lambda_i^*$ 、 $\mu_j^*$ 使得：

$$0 \in \partial f(x^*) + \sum_{i=1}^m \lambda_i^* \partial g_i(x^*) + \sum_{j=1}^n \mu_j^* \partial h_i(x^*); \quad (8.33)$$

$$\mu_j^* = 0, \quad \text{if } h_j(x^*) < 0;$$

$$\mu_j^* \geq 0, \quad \text{if } h_j(x^*) = 0.$$

(8.34)

*Proof.* 由定理182和166<sup>1</sup>，我们只需验证

$$N_C(x^*) = \sum_{i=1}^m \lambda_i \partial g_i(x^*) + \sum_{j \in A(x^*)} \mu_j \partial h_i(x^*),$$

其中 $A(x^*) = \{j | h_j(x^*) = 0\}$ 且 $\mu_j \geq 0$ 。 $A(x^*)$ 称为 $x^*$ 处的起作用集。

记 $C_i = \{x | g_i(x) = 0\}$ ,  $D_j = \{x | h_i(x) \leq 0\}$ 。则 $C = (\cap_{i=1}^m C_i) \cap (\cap_{j=1}^n D_j)$ 。由定理167<sup>2</sup>,

$$N_C(x^*) = \sum_{i=1}^m N_{C_i}(x^*) + \sum_j N_{D_j}(x^*).$$

<sup>1</sup>对于凸集， $x^*$ 自然正则，于是 $T_C^*(x^*) = N_C(x^*)$ 。

<sup>2</sup>Slater条件保证凸集的相对内部之交非空。

当  $h_j(x^*) < 0$ , 即  $j \notin A(x^*)$  时,  $N_{D_j}(x^*) = \{0\}$ 。另外可以验证

$$\begin{aligned} N_{C_i}(x^*) &= \{\lambda_i \partial g_i(x^*)\}, \\ N_{D_j}(x^*) &= \{\mu_j \partial h_i(x^*) | \mu_j \geq 0\}, j \in A(x^*). \end{aligned}$$

于是得到KKT条件。  $\square$

满足以上条件的点  $x^*$  称为KKT点。

#### 8.4.1 凸优化的次梯度法[13]

光滑优化里很多直观自然的技术在凸优化里将不能使用。比如按坐标依次取极小和是精确线搜索（请举反例！）。对于一般的凸优化，要使用次梯度作为搜索方向，其算法描述见算法11，其中  $\mathcal{P}_C$  是往  $C$  上投影的算子。

---

##### Algorithm 11 次梯度法

---

- 1: 初始化  $x_0, k = 0$ .
  - 2: **while** 停止条件不满足 **do**
  - 3:   选定次梯度  $g_k \in \partial f(x_k)$ ;
  - 4:   选定步长  $t_k > 0$ ;
  - 5:   更新:  $x_{k+1} = \mathcal{P}_C(x_k - t_k g_k), k \leftarrow k + 1$ .
  - 6: **end while**
- 

次梯度法和前面光滑优化的重大区别在于它不能使用精确线搜索确定步长，因此不能保证目标函数值持续下降，导致其收敛速度一般都很慢。但是只要步长充分小，可以保证  $x_{k+1}$  比  $x_k$  更接近最优解  $x^*$ 。

**定理183.** 如果  $x_k$  不是最优解，则对任意  $0 < t_k < \frac{2(f(x_k) - f^*)}{\|g_k\|^2}$ , 有

$$\|x_{k+1} - x^*\| < \|x_k - x^*\|. \quad (8.35)$$

*Proof.*

$$\begin{aligned} \|x_k - t_k g_k - x^*\|^2 &= \|x_k - x^*\|^2 + t_k^2 \|g_k\|^2 - 2t_k \langle x_k - x^*, g_k \rangle \\ &\leq \|x_k - x^*\|^2 + t_k^2 \|g_k\|^2 + 2t_k (f^* - f(x_k)) \\ &< \|x_k - x^*\|^2. \end{aligned} \quad (8.36)$$

其中利用了  $f$  的凸性。于是由凸集投影的非扩张性，

$$\|x_{k+1} - x^*\| \leq \|(x_k - t_k g_k) - x^*\| < \|x_k - x^*\|. \quad (8.37)$$

$\square$

可以证明，只要步长满足：

$$t_k \rightarrow 0, \quad \sum_k t_k = +\infty,$$

则次梯度法收敛。

### 练习184.

1. 证明： $\exp(\mathbf{x}^T \Sigma \mathbf{x})$  在  $C = \{\mathbf{x} \mid \|\mathbf{x}\| \leq 1\}$  上是强凸的，并估计相应的  $l$  和  $L$ ，其中  $\Sigma = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}$ 。
2. 编写最速下降法（精确线搜索和回溯线搜索）的代码，并用于求解：

$$\min_{\mathbf{x}} \exp((\mathbf{x} - \mu_1)^T \Sigma_1 (\mathbf{x} - \mu_1)) + \exp((\mathbf{x} - \mu_2)^T \Sigma_2 (\mathbf{x} - \mu_2)),$$

其中  $\mu_1 = (0, 1)^T$ ,  $\mu_2 = (1, 0)^T$ ,  $\Sigma_1 = \begin{pmatrix} 1 & 2 \\ 2 & 2 \end{pmatrix}$ ,  $\Sigma_2 = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}$ . 交代码，报告参数、迭代次数和数值结果。

3. 编写次梯度下降法的代码，并用于求解：

$$\min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{b}\|_1, \quad s.t. \quad \mathbf{x} \geq \mathbf{0},$$

其中  $\mathbf{A} = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}$ ,  $\mathbf{b} = (-1, 2)^T$ . 交代码，报告步长选择、迭代次数和数值结果。

## 8.5 Nesterov技巧

### 8.5.1 $C^{1,1}$ 凸优化的复杂度[106]

考虑如下一类问题

$$\begin{aligned} \min_x & f(x), \\ s.t. & x \in \mathbb{R}^m, \end{aligned} \tag{8.38}$$

其中  $f$  是  $C^{1,1}$  光滑的凸函数，即  $f$  连续可导且导数 Lipschitz 连续：

$$\|\text{grad } f(x) - \text{grad } f(y)\| \leq L(f)\|x - y\|, \quad \forall x, y \in \mathbb{R}^m. \tag{8.39}$$

假设问题(8.38)可解，解集记为  $X$ 。记  $R(f)$  为原点到  $X$  的距离。不失一般性，可假定迭代初始点为原点。定义函数类

$$S_m(L, R) = \{f \mid f : \mathbb{R}^m \rightarrow \mathbb{R}, f \in C^{1,1}, L(f) \leq L, R(f) \leq R\}. \tag{8.40}$$

接下来我们定义一阶预测器(oracle) $\mathcal{O}$ ，它负责返回  $f$  在给定点  $x$  处的函数值和导数。

由于函数类  $S_m(L, R)$  里的函数不一定满足(8.3)，所以第8.2.1和8.2.2节里的线性收敛的结论不再成立。事实上，我们只有次线性收敛的结果。

**定理185.** 问题(8.38)在函数类 $S_m(L, R)$ 上的复杂度为:

$$O(1) \min \left\{ m, \sqrt{\frac{LR^2}{\varepsilon}} \right\} \leq Compl(\varepsilon) \leq \sqrt{\frac{4LR^2}{\varepsilon}}. \quad (8.41)$$

以上定理说明, 任一算法在函数类 $S_m(L, R)$ 上最慢的收敛速度只能为 $O(k^{-2})$ 。但是通常的梯度下降法只能达到 $O(k^{-1})$  的收敛速度, 令我们怀疑 $O(k^{-2})$ 的收敛速度是否能达到。

**定理186.** 设问题(8.38)用常数步长的梯度下降法来解:

$$x_0 = 0, \quad x_{k+1} = x_k - \gamma \text{grad}f(x_k), \quad (8.42)$$

其中 $\gamma \in (0, 2/L(f))$ , 则该迭代的收敛速度是 $O(K^{-1})$ :

$$f(x_K) - f^* \leq \frac{R^2(f)}{\gamma(2 - \gamma L(f))K}. \quad (8.43)$$

*Proof.* 首先由 $f$ 的凸性和定理181我们有:

$$f(x) + \langle \text{grad}f(x), y - x \rangle \leq f(y) \leq f(x) + \langle \text{grad}f(x), y - x \rangle + \frac{L}{2} \|y - x\|^2. \quad (8.44)$$

所以

$$f(x_{k+1}) \leq f(x_k) - \gamma(1 - \gamma L(f)/2) \|\text{grad}f(x_k)\|^2. \quad (8.45)$$

由于 $\gamma \in (0, 2/L(f))$ ,  $\{f(x_k)\}$ 是递减序列。把上式对 $k = 1, \dots$ 相加, 得

$$\sum_{k=1}^{+\infty} \|\text{grad}f(x_k)\|^2 \leq \frac{f(x_0) - f(x^*)}{\gamma(1 - \gamma L(f)/2)} \quad (8.46)$$

$$\leq \frac{\frac{L(f)}{2} \|x_0 - x^*\|^2}{\gamma(1 - \gamma L(f)/2)} \quad (8.47)$$

$$\leq \frac{L(f)R^2(f)}{\gamma(2 - \gamma L(f))}. \quad (8.48)$$

于是

$$\|x_{k+1} - x^*\|^2 = \|x_k - x^*\|^2 - 2\gamma \langle \text{grad}f(x_k), x_k - x^* \rangle + \gamma^2 \|\text{grad}f(x_k)\|^2 \quad (8.49)$$

$$\leq \|x_k - x^*\|^2 - 2\gamma(f(x_k) - f(x^*)) + \gamma^2 \|\text{grad}f(x_k)\|^2. \quad (8.50)$$

所以

$$\sum_{k=1}^K (f(x_k) - f(x^*)) \leq \frac{1}{2\gamma} \left[ \|x_1 - x^*\|^2 - \|x_{K+1} - x^*\|^2 + \gamma^2 \sum_{k=1}^K \|\text{grad}f(x_k)\|^2 \right] \quad (8.51)$$

$$\leq \frac{1}{2\gamma} \left[ R^2(f) + \gamma^2 \frac{L(f)R^2(f)}{\gamma(2 - \gamma L(f))} \right]. \quad (8.52)$$

最后, 由 $\{f(x_k)\}$ 递减, 得

$$f(x_K) - f^* \leq \frac{1}{K} \sum_{k=1}^K (f(x_k) - f(x^*)) \leq \frac{R^2(f)}{\gamma(2 - \gamma L(f))K}. \quad (8.53)$$

□

### 8.5.2 $C^{1,1}$ 凸优化的Nesterov方法

Y. Nesterov [106, 107]首先发现函数类 $S_m(L, R)$ 上最慢的收敛速度为 $O(k^{-2})$ 的算法是存在的。他设计了如下算法12来求解问题(8.38)。

---

#### Algorithm 12 Nesterov方法

---

- 1: 初始化 $x_0 = y_1 = q_0 = 0, t_0 = 1, k = 1.$
  - 2: **while** 停止条件不满足 **do**
  - 3:     更新:  $x_k = y_k - \frac{1}{t_{k-1}}(y_k + q_{k-1}), y_{k+1} = x_k - \frac{1}{L(f)}\text{grad}f(x_k), q_k = q_{k-1} + \frac{t_{k-1}}{L(f)}\text{grad}f(x_k), t_k = \frac{1 + \sqrt{1 + 4t_{k-1}^2}}{2};$
  - 4:      $k \leftarrow k + 1.$
  - 5: **end while**
- 

**定理187.** 算法12在函数类 $S_m(L, R)$ 上最慢的收敛速度是 $O(K^{-2})$ :

$$f(y_K) - f^* \leq \frac{2L(f)R^2(f)}{K^2}. \quad (8.54)$$

*Proof.* 我们首先证明:

$$f(x - l^{-1}\text{grad}f(x)) \leq f(x) - \frac{1}{2l}\|\text{grad}f(x)\|^2, \quad \forall l \geq L(f). \quad (8.55)$$

这由定理181立得。

其次我们证明:

$$f(y) \geq f(y_{k+1}) + \langle \text{grad}f(x_k), y - x_k \rangle + \frac{1}{2L(f)}\|\text{grad}f(x_k)\|^2. \quad (8.56)$$

事实上, 在(8.55)中令 $x = x_k$ , 再利用 $f(x_k) \leq f(y) - \langle \text{grad}f(x_k), y - x_k \rangle$ 和 $y_k$ 的定义, 重新整理一下即得。

定义 $\varepsilon_k = f(y_k) - f^*$ 。在不等式(8.56)中令 $y = x^*$ , 得

$$0 \geq \varepsilon_{k+1} + \langle \text{grad}f(x_k), x^* - x_k \rangle + \frac{1}{2L(f)}\|\text{grad}f(x_k)\|^2. \quad (8.57)$$

把 $x_k = y_k - \frac{1}{t_{k-1}}(y_k + q_{k-1})$ 改写成 $x_k = (t_{k-1} - 1)(y_k - x_k) - q_{k-1}$ 代入(8.57)得:

$$0 \geq \varepsilon_{k+1} + \langle \text{grad}f(x_k), x^* \rangle + (t_{k-1} - 1) \langle \text{grad}f(x_k), x_k - y_k \rangle + \langle \text{grad}f(x_k), q_{k-1} \rangle + \frac{1}{2L(f)} \|\text{grad}f(x_k)\|^2. \quad (8.58)$$

另一方面, 在不等式(8.56)中令 $y = y_k$ , 得

$$0 \geq \varepsilon_{k+1} - \varepsilon_k + \langle \text{grad}f(x_k), y_k - x_k \rangle + \frac{1}{2L(f)} \|\text{grad}f(x_k)\|^2. \quad (8.59)$$

因此

$$\langle \text{grad}f(x_k), x_k - y_k \rangle \geq \varepsilon_{k+1} - \varepsilon_k + \frac{1}{2L(f)} \|\text{grad}f(x_k)\|^2. \quad (8.60)$$

代入(8.58)得

$$0 \geq \varepsilon_{k+1} + \langle \text{grad}f(x_k), x^* \rangle + (t_{k-1} - 1) \left( \varepsilon_{k+1} - \varepsilon_k + \frac{1}{2L(f)} \|\text{grad}f(x_k)\|^2 \right) \quad (8.61)$$

$$+ \langle \text{grad}f(x_k), q_{k-1} \rangle + \frac{1}{2L(f)} \|\text{grad}f(x_k)\|^2. \quad (8.62)$$

整理后两边乘以 $t_{k-1}/L(f)$ 得

$$0 \geq \frac{t_{k-1}^2}{L(f)} \varepsilon_{k+1} - \frac{t_{k-1}^2 - t_{k-1}}{L(f)} \varepsilon_k \quad (8.63)$$

$$+ \left\langle \frac{t_{k-1}}{L(f)} \text{grad}f(x_k), x^* \right\rangle + \left\langle \frac{t_{k-1}}{L(f)} \text{grad}f(x_k), q_{k-1} \right\rangle + \frac{t_{k-1}^2}{2L^2(f)} \|\text{grad}f(x_k)\|^2. \quad (8.64)$$

利用等式 $\frac{t_{k-1}}{L(f)} \text{grad}f(x_k) = q_k - q_{k-1}$ 和 $t_{k-1}^2 - t_{k-1} = t_{k-2}^2$ , 上式可重写为

$$0 \geq \frac{t_{k-1}^2}{L(f)} \varepsilon_{k+1} - \frac{t_{k-2}^2}{L(f)} \varepsilon_k + \langle q_k - q_{k-1}, x^* \rangle \quad (8.65)$$

$$+ \langle q_k - q_{k-1}, q_{k-1} \rangle + \frac{1}{2} \|q_k - q_{k-1}\|^2 \quad (8.66)$$

$$= \frac{t_{k-1}^2}{L(f)} \varepsilon_{k+1} - \frac{t_{k-2}^2}{L(f)} \varepsilon_k + \langle q_k - q_{k-1}, x^* \rangle + \frac{1}{2} \|q_k\|^2 - \frac{1}{2} \|q_{k-1}\|^2. \quad (8.67)$$

将上式对 $k = 1, \dots, K-1$ 相加, 得

$$\frac{t_{K-2}^2}{L(f)} \varepsilon_K \leq -\langle q_{K-1}, x^* \rangle - \frac{1}{2} \|q_{K-1}\|^2 \leq \frac{1}{2} \|x^*\|^2 = \frac{1}{2} R^2(f). \quad (8.68)$$

最后, 由递推式 $t_k^2 - t_k = t_{k-1}^2$ 容易证得

$$t_k \geq 1 + k/2.$$

于是证得(8.54)。 □

### 8.5.3 Accelerated Proximal Gradient (APG) 方法

Nesterov的原方法是针对目标函数为 $C^{1,1}$ 光滑的函数的无约束优化的。在实际应用里，这些条件太苛刻了。Beck和Teboulle[8]对Nesterov方法做出了重要的推广，称为Accelerated Proximal Gradient (APG) 方法，使之能应用于广泛的实际凸优化问题中。Beck和Teboulle考虑的凸问题为：

$$\min_x f(x) + g(x), \quad (8.69)$$

其中 $f$ 和 $g$ 都是凸函数， $f$ 仍然满足(8.39)。这类问题经常是罚函数形式的带约束优化问题，比如问题

$$\begin{aligned} & \min_x g(x), \\ & \text{s.t. } h(x) = 0, \end{aligned} \quad (8.70)$$

经常转化成无约束问题

$$\min_x (h(x))^2 + \mu g(x) \quad (8.71)$$

来求解。

为方便我们把 $f(x) + g(x)$ 和 $L(f)$ 简写成 $F(x)$ 和 $L$ 。APG方法先对 $f(x)$ 在特殊点 $y$ 处进行一阶近似：

$$f(x) \leq f(y) + \langle \text{grad}f(y), x - y \rangle + \frac{L}{2} \|x - y\|^2, \quad (8.72)$$

再求解比原问题简单的子问题：

$$x_k = p_L(y_k) = \arg \min_x Q_L(x, y) \quad (8.73)$$

$$= \arg \min_x \left\{ g(x) + \frac{L}{2} \left\| x - \left( y - \frac{1}{L} \text{grad}f(y) \right) \right\|^2 \right\}, \quad (8.74)$$

其中 $Q_L(x, y) = f(y) + \langle \text{grad}f(y), x - y \rangle + \frac{L}{2} \|x - y\|^2 + g(x)$ 。然后和Nesterov方法一样进行适当的插值。具体描述见算法13。

---

#### Algorithm 13 Accelerated Proximal Gradient (APG)方法

---

- 1: 初始化 $x_0 = y_1$ ,  $t_1 = 1$ ,  $k = 1$ .
  - 2: **while** 停止条件不满足 **do**
  - 3:     更新:  $x_k = p_L(y_k)$ ,  $t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2}$ ,  $y_{k+1} = x_k + \frac{t_k - 1}{t_{k+1}}(x_k - x_{k-1})$ ;
  - 4:      $k \leftarrow k + 1$ .
  - 5: **end while**
- 

可以证明APG方法的收敛速度也至少是 $O(K^{-2})$ 。我们先证明三个引理。

引理188.

$$F(x) \geq F(p_L(y)) + L \langle y - x, p_L(y) - y \rangle + \frac{L}{2} \|p_L(y) - y\|^2, \quad \forall x, y. \quad (8.75)$$

*Proof.* 把  $Q_L(x, y)$  对  $x$  在  $x = p_L(y)$  处求次梯度可知存在  $\gamma(y) \in \partial g(p_L(y))$  使得：

$$\text{grad}f(y) + L(p_L(y) - y) + \gamma(y) = 0. \quad (8.76)$$

由  $f$  和  $g$  的凸性，得到

$$f(x) \geq f(y) + \langle \text{grad}f(y), x - y \rangle, \quad (8.77)$$

$$g(x) \geq g(p_L(y)) + \langle \gamma(y), x - p_L(y) \rangle. \quad (8.78)$$

两式相加得到

$$F(x) \geq f(y) + \langle \text{grad}f(y), x - y \rangle + g(p_L(y)) + \langle \gamma(y), x - p_L(y) \rangle. \quad (8.79)$$

另一方面，由于  $f$  满足不等式(8.72)，所以  $F(p_L(y)) \leq Q(p_L(y), y)$ 。于是

$$F(x) - F(p_L(y)) \geq f(y) + \langle \text{grad}f(y), x - y \rangle + g(p_L(y)) + \langle \gamma(y), x - p_L(y) \rangle \quad (8.80)$$

$$- Q(p_L(y), y) \quad (8.81)$$

$$= -\frac{L}{2} \|p_L(y) - y\|^2 + \langle \gamma(y) + \text{grad}f(y), x - p_L(y) \rangle \quad (8.82)$$

$$= -\frac{L}{2} \|p_L(y) - y\|^2 + L \langle y - p_L(y), x - p_L(y) \rangle \quad (8.83)$$

$$= \frac{L}{2} \|p_L(y) - y\|^2 + L \langle y - x, p_L(y) - y \rangle. \quad (8.84)$$

□

引理189.

$$\frac{2}{L} t_k^2 \varepsilon_k - \frac{2}{L} t_{k+1}^2 \varepsilon_{k+1} \geq \|u_{k+1}\|^2 - \|u_k\|^2, \quad (8.85)$$

其中  $\varepsilon_k = F(x_k) - F^*$ ,  $u_k = t_k x_k - (t_k - 1)x_{k-1} - x^*$ 。

*Proof.* 在不等式(8.75)中分别取  $x = x_k$ ,  $y = y_{k+1}$  和  $x = x^*$ ,  $y = y_{k+1}$ , 得

$$\frac{2}{L} (\varepsilon_k - \varepsilon_{k+1}) \geq \|x_{k+1} - y_{k+1}\|^2 + 2 \langle x_{k+1} - y_{k+1}, y_{k+1} - x_k \rangle, \quad (8.86)$$

$$-\frac{2}{L} \varepsilon_{k+1} \geq \|x_{k+1} - y_{k+1}\|^2 + 2 \langle x_{k+1} - y_{k+1}, y_{k+1} - x^* \rangle. \quad (8.87)$$

把第一个不等式乘以  $(t_{k+1} - 1)$  加到第二个不等式，得

$$\frac{2}{L} [(t_{k+1} - 1)\varepsilon_k - t_{k+1}\varepsilon_{k+1}] \geq t_{k+1} \|x_{k+1} - y_{k+1}\|^2 + 2 \langle x_{k+1} - y_{k+1}, t_{k+1}y_{k+1} - (t_{k+1} - 1)x_k - x^* \rangle. \quad (8.88)$$

两边乘以  $t_{k+1}$  并利用等式  $t_{k+1}^2 - t_k^2 = t_k^2$ , 得

$$\frac{2}{L}(t_k^2 \varepsilon_k - t_{k+1}^2 \varepsilon_{k+1}) \geq \|t_{k+1}(x_{k+1} - y_{k+1})\|^2 + 2t_{k+1} \langle x_{k+1} - y_{k+1}, t_{k+1}y_{k+1} - (t_{k+1} - 1)x_k - x^* \rangle. \quad (8.89)$$

利用恒等式

$$\|b - a\|^2 + 2 \langle b - a, a - c \rangle = \|b - c\|^2 - \|a - c\|^2,$$

上式可改写成

$$\frac{2}{L}(t_k^2 \varepsilon_k - t_{k+1}^2 \varepsilon_{k+1}) \geq \|t_{k+1}x_{k+1} - (t_{k+1} - 1)x_k - x^*\|^2 - \|t_{k+1}y_{k+1} - (t_{k+1} - 1)x_k - x^*\|^2. \quad (8.90)$$

由  $y_{k+1}$  和  $u_k$  的定义式, 上式实际上就是

$$\frac{2}{L}(t_k^2 \varepsilon_k - t_{k+1}^2 \varepsilon_{k+1}) \geq \|u_{k+1}\|^2 - \|u_k\|^2. \quad (8.91)$$

□

**引理190.**

$$\frac{2}{L} \varepsilon_1 \leq \|y_1 - x^*\|^2 - \|x_1 - x^*\|^2. \quad (8.92)$$

*Proof.* 在不等式(8.75)中取  $x = x^*$ 、 $y = y_1$ , 得

$$F(x^*) - F(p_L(y_1)) \geq \frac{L}{2} \|p_L(y_1) - y\|^2 + L \langle y_1 - x^*, p_L(y_1) - y_1 \rangle. \quad (8.93)$$

于是

$$-\varepsilon_1 = F(x^*) - F(x_1) = F(x^*) - F(p_L(y_1)) \quad (8.94)$$

$$\geq \frac{L}{2} \|p_L(y_1) - y\|^2 + L \langle y_1 - x^*, p_L(y_1) - y_1 \rangle \quad (8.95)$$

$$= \frac{L}{2} \|x_1 - y\|^2 + L \langle y_1 - x^*, x_1 - y_1 \rangle \quad (8.96)$$

$$= \frac{L}{2} (\|x_1 - x^*\|^2 - \|y_1 - x^*\|^2). \quad (8.97)$$

此即(8.92)。 □

**定理191.** 算法 13 的收敛速度至少是  $O(K^{-2})$ :

$$f(x_K) - f^* \leq \frac{2L\|x_0 - x^*\|^2}{(K+1)^2}. \quad (8.98)$$

*Proof.* 由引理189,  $\frac{2}{L}t_k^2 \varepsilon_k + \|u_k\|^2$  是递减序列, 所以

$$\frac{2}{L}t_K^2 \varepsilon_K \leq \frac{2}{L}t_1^2 \varepsilon_1 + \|x_1 - x^*\|^2. \quad (8.99)$$

由引理190, 上式右端  $\leq \|x_0 - x^*\|^2$ . 另一方面, 易证  $t_k \geq (k+1)/2$ , 所以

$$\varepsilon_K \leq \frac{2L\|x_0 - x^*\|^2}{(K+1)^2}. \quad (8.100)$$

□

如果不能准确估计 $L(f)$ , 则可以使用带回溯的APG方法, 见算法14, 其收敛速度也是 $O(K^{-2})$ 。

---

**Algorithm 14** 带回溯的Accelerated Proximal Gradient (APG)方法

- 1: 初始化 $L_0 > 0$ ,  $\eta > 1$ ,  $x_0 = y_1$ ,  $t_1 = 1$ ,  $k = 1$ .
- 2: **while** 停止条件不满足 **do**
- 3:     找出最小的非负整数 $i_k$ , 使得对于 $\hat{L} = \eta^{i_k} L_{k-1}$ ,

$$F(p_{\hat{L}}(y_k)) \leq Q_{\hat{L}}(p_{\hat{L}}(y_k), y_k)$$

成立。

- 4:     取 $L_k = \eta^{i_k} L_{k-1}$ , 更新:  $x_k = p_{L_k}(y_k)$ ,  $t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2}$ ,  $y_{k+1} = x_k + \frac{t_k - 1}{t_{k+1}}(x_k - x_{k-1})$ ;
  - 5:      $k \leftarrow k + 1$ .
  - 6: **end while**
- 

对于常见的带约束凸优化问题:

$$\min_{\mathbf{x}} f(\mathbf{x}), \quad s.t. \quad \mathbf{A}\mathbf{x} = \mathbf{b}, \quad (8.101)$$

可以通过罚函数的方式转化为无约束问题:

$$\min_{\mathbf{x}} \frac{\alpha}{2} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 + f(\mathbf{x}), \quad (8.102)$$

来求近似解。但是要注意的是, 如果一开始就让惩罚系数 $\alpha$ 很大, 则收敛会极其地慢。所以一般使用Continuation技巧[51], 即一开始取较小的值, 随着迭代进行再逐渐增大到一个较大的值, 这样简单处理后就会显著地加快收敛。

Zuo和Lin[168]对条件(8.39)做了放松以加速收敛。他们指出(8.39)实际上只是为了保证(8.72)成立。对APG算法收敛性起关键作用的只是(8.72)式。因此, Zuo和Lin[168]指出(8.39)可以放宽为

$$f(x) \leq f(y) + \langle \text{grad}f(y), y - x \rangle + \frac{1}{2} \|y - x\|_{\mathbf{L}_f}^2, \quad (8.103)$$

其中 $\mathbf{L}_f$ 为正定矩阵,  $\|\mathbf{x}\|_{\mathbf{L}_f} = \sqrt{\mathbf{x}^T \mathbf{L}_f \mathbf{x}}$ 。相应地, 算法13中的更新算法为:

$$\mathbf{x}_k = \arg \min_{\mathbf{x}} \left\{ g(x) + \langle \nabla f(\mathbf{y}_k), \mathbf{x} - \mathbf{y}_k \rangle + \frac{1}{2} \|\mathbf{x} - \mathbf{y}_k\|_{\mathbf{L}_f}^2 \right\}, \quad (8.104)$$

其余不变,  $O(k^{-2})$ 的收敛速度也得到保持。当 $g(\mathbf{x})$ 可分解时,  $\mathbf{L}_f$ 常常可以选成对角阵, 为不同的变量提供不同的一阶Lipschitz系数, 因此可以加速收敛。Zuo和Lin[168]把GAPG应用到图像恢复问题上得到了比APG快2倍以上的加速比。

## 8.6 Lagrange函数

问题(8.29)的Lagrange函数定义为:

$$L(x, \lambda, \mu) = f(x) + \sum_{i=1}^m \lambda_i g_i(x) + \sum_{j=1}^n \mu_j h_j(x). \quad (8.105)$$

$\lambda_i$ 、 $\mu_j$ 称为Lagrange乘子。则KKT条件的(8.33)条件可以很简洁地写成

$$0 \in \partial_x L(x, \lambda, \mu).$$

## 8.7 对偶问题

使用Lagrange函数可以对凸优化的对偶理论进行简洁地描述。问题(8.28)的对偶函数(dual function)为:

$$f^*(\lambda, \mu) = \inf_{x \in C} L(x, \lambda, \mu). \quad (8.106)$$

相应的对偶问题可以通过分析 $f^*(\lambda, \mu)$ 何时取有限值获得。比如 $f^*(\lambda, \mu) > -\infty$ 当且仅当 $(\lambda, \mu) \in D$ ，则问题(8.28)的对偶问题(dual problem)为

$$\begin{aligned} & \max_{\lambda, \mu} f^*(\lambda, \mu), \\ & \text{s.t. } (\lambda, \mu) \in D. \end{aligned} \quad (8.107)$$

而问题(8.28)称为原问题(primal problem)。

**定理192.** 对偶函数有如下性质:

1.  $D$ 是凸集;
2.  $f^*(\lambda, \mu) \leq f(x)$ ,  $\forall (\lambda, \mu) \in D, x \in C$ ;
3.  $f^*(\lambda, \mu)$ 是 $D$ 上的凹函数且具有超梯度(supergradient) $g_i(x_{\lambda, \mu}) \in \partial_{\lambda_i} f^*(\lambda, \mu)$ 、 $h_j(x_{\lambda, \mu}) \in \partial_{\mu_j} f^*(\lambda, \mu)$ , 其中 $x_{\lambda, \mu} = \arg \min_{x \in C} L(x, \lambda, \mu)$ ;
4. 如果对每一个 $(\lambda, \mu)$ ,  $x_{\lambda, \mu}$ 唯一, 则 $f^*(\lambda, \mu)$ 可微。特别地, 如果 $f(x)$ 严格凸, 则 $f^*(\lambda, \mu)$ 可微。进一步, 如果 $f(x)$ 强凸, 强凸系数为 $\nu$ , 则 $f^*(\lambda, \mu)$ 的梯度的Lipschitz系数为 $1/\nu$ 。

*Proof.* 3.

$$\begin{aligned} f^*(\lambda', \mu') &= \min_{x \in C} L(x, \lambda', \mu') \\ &\leq L(x_{\lambda, \mu}, \lambda', \mu') \\ &= L(x_{\lambda, \mu}, \lambda, \mu) + \sum_{i=1}^m (\lambda'_i - \lambda_i) g_i(x_{\lambda, \mu}) + \sum_{j=1}^n (\mu'_j - \mu_j) h_j(x_{\lambda, \mu}) \\ &= f^*(\lambda, \mu) + \sum_{i=1}^m (\lambda'_i - \lambda_i) g_i(x_{\lambda, \mu}) + \sum_{j=1}^n (\mu'_j - \mu_j) h_j(x_{\lambda, \mu}). \end{aligned} \quad (8.108)$$

所以 $f^*(\lambda, \mu)$ 具有超梯度，因此是凹函数。

4. 将Danskin定理（定理160）直接应用于 $-L(x, \lambda, \mu)$ 。  $\square$

求解对偶问题有如下优势：

1. 当约束少时，对偶问题维数就低，求解起来可能计算量少；
2. 原问题目标函数严格凸但不可微时，对偶函数可微，求解更方便，收敛速度更快。

### 练习193.

1. 证明定理192-2。
2. 根据伴随算子（adjoint operator）的定义： $\langle \mathcal{A}^*(\mathbf{x}), \mathbf{Y} \rangle = \langle \mathbf{x}, \mathcal{A}(\mathbf{Y}) \rangle, \forall \mathbf{x}, \mathbf{Y}$ ，求 $\pi_\Omega$ 的伴随算子，其中 $\pi_\Omega$ 是把矩阵下标在 $\Omega$ 里的元素取出来的线性算子。
3. 求 $\mathcal{A}$ 的伴随算子，其中 $\mathcal{A}(\mathbf{X}) = \mathbf{A}\mathbf{X}\mathbf{B}$ 。
4. 求问题： $\min_{\mathbf{X}} \|\mathbf{X}\|_F, \quad s.t. \quad \pi_\Omega(\mathbf{X}) = \mathbf{b}$ 的对偶问题。

## 8.8 增广Lagrange乘子法[13]

问题(8.29)只有等式约束时，问题(8.29)简化为：

$$\min_x f(x), \quad s.t. \quad g_i(x) = 0, \quad i = 1, 2, \dots, m, \quad (8.109)$$

其增广Lagrange函数（Augmented Lagrangian function）定义为：

$$\tilde{L}(x, \lambda, c) = f(x) + \sum_{i=1}^m \left( \lambda_i g_i(x) + \frac{c_i}{2} (g_i(x))^2 \right). \quad (8.110)$$

其优势在于 $\tilde{L}$ 和 $L$ 有相同的极小点和Lagrange乘子。由于增加了可能是强凸的项 $\sum_{i=1}^m \frac{c_i}{2} (g_i(x))^2$ ，使得问题（关于 $(x, \lambda, \mu)$ 的）条件数可能减小，因此算法收敛一般要比Lagrange乘子法快得多。另一方面，Lagrange乘子 $\lambda_i$ 的更新非常方便：

$$\lambda_i^{(k+1)} = \lambda_i^{(k)} + c_i g_i(x_{k+1}). \quad (8.111)$$

增广Lagrange函数其实就是问题

$$\min_x f(x) + \sum_{i=1}^m \frac{c_i}{2} (g_i(x))^2, \quad s.t. \quad g_i(x) = 0, \quad i = 1, 2, \dots, m, \quad (8.112)$$

的Lagrange函数。极小化它显然不改变问题的解。之所以Lagrange乘子的更新步长选作 $c_i$ 是因为如下原因。问题(8.109)的KKT条件为

$$g_i(x^*) = 0, \quad i = 1, 2, \dots, m,$$

$$0 \in \partial f(x^*) + \sum_{i=1}^m \lambda_i^* \partial g_i(x^*). \quad (8.113)$$

如果每次极小化增广Lagrange函数(8.110), 则

$$\begin{aligned} 0 &\in \partial f(x_{k+1}) + \sum_{i=1}^m \left( \lambda_i^{(k)} \partial g_i(x_{k+1}) + c_i g_i(x_{k+1}) \partial g_i(x_{k+1}) \right) \\ &= \partial f(x_{k+1}) + \sum_{i=1}^m \left( \lambda_i^{(k)} + c_i g_i(x_{k+1}) \right) \partial g_i(x_{k+1}). \end{aligned}$$

所以可以看出, 如果选择

$$\lambda_i^{(k+1)} = \lambda_i^{(k)} + c_i g_i(x_{k+1})$$

则可以保证每一次迭代得到的 $(x_{k+1}, \lambda_i^{(k+1)})$ 都满足对偶可行性条件(8.113)。

问题(8.29)含不等式约束时, 可以引入松弛变量 $s_i$ 把不等式写成等式:

$$h_i(x) + s_i^2 = 0.$$

于是可以套用上面针对等式约束的增广Lagrange函数, 得出问题(8.29)的增广Lagrange函数:

$$\tilde{L}(x, \lambda, \mu, c, d, s) = f(x) + \sum_{i=1}^m \left( \lambda_i g_i(x) + \frac{c_i}{2} (g_i(x))^2 \right) + \sum_{j=1}^n \left( \mu_j (g_j(x) + s_j^2) + \frac{d_j}{2} (g_j(x) + s_j^2)^2 \right). \quad (8.114)$$

对 $s$ 极小化得问题(8.29)的增广Lagrange函数的最终形式:

$$\tilde{L}(x, \lambda, \mu, c, d) = f(x) + \sum_{i=1}^m \left( \lambda_i g_i(x) + \frac{c_i}{2} (g_i(x))^2 \right) + \sum_{j=1}^n \left( \mu_j h_j^+(x) + \frac{d_j}{2} (h_j^+(x))^2 \right), \quad (8.115)$$

其中 $h_j^+(x) = \max(h_j, -\mu_j/d_j)$ 。上式也可以写成:

$$\tilde{L}(x, \lambda, \mu, c, d) = f(x) + \sum_{i=1}^m \left( \lambda_i g_i(x) + \frac{c_i}{2} (g_i(x))^2 \right) + \sum_{j=1}^n \frac{1}{2d_j} \left\{ [\max(0, \mu_j + d_j h_j(x))]^2 - \mu_j^2 \right\}. \quad (8.116)$$

增广Lagrange乘子法的完整描述见算法15。

当子问题

$$x_{k+1} = \arg \min_x \tilde{L}(x, \lambda^{(k)}, \mu^{(k)}, c^{(k)}, d^{(k)})$$

容易求解时, 增广Lagrange乘子法非常有优势, 因为可以证明当 $f$ 、 $g_i$ 、 $h_j$ 可微时, 如果惩罚系数 $c_i$ 、 $d_j$ 有界则 $\{x_k\}$ 线性收敛; 如果 $c_i$ 、 $d_j$ 无界则 $\{x_k\}$ 超线性收敛。但要注意的是, 当 $c_i$ 、 $d_j$ 增长时, 子问题的(关于 $x$ 的)条件数也在增大, 导致子问题求解的难度增大。因此 $c_i$ 、 $d_j$ 增长的速度需要仔细控制。

---

**Algorithm 15** 增广Lagrange乘子法

---

```

1: 初始化  $x_0 \in C$ ,  $\lambda_i^{(0)}, \mu_j^{(0)} \geq 0$ ,  $c_i^{(0)} > 0$ ,  $d_j^{(0)} > 0$ ,  $k = 0$ ,  $\rho > 1$ .
2: while KKT条件不满足 do
3:    $x_{k+1} = \arg \min_x \tilde{L}(x, \lambda^{(k)}, \mu^{(k)}, c^{(k)}, d^{(k)})$ ;
4:    $\lambda_i^{(k+1)} = \lambda_i^{(k)} + c_i^{(k)} g_i(x_{k+1})$ ,  $\mu_j^{(k+1)} = \max(0, \mu_j^{(k)} + d_j h_j(x_{k+1}))$ ;
5:    $c_i^{(k+1)} = \rho c_i^{(k)}$ ,  $d_j^{(k+1)} = \rho d_j^{(k)}$ ;
6:    $k \leftarrow k + 1$ .
7: end while

```

---

## 8.9 交错方向法（ADM）及其线性化（LADM）

在有约束情形，APG只能求得近似解，如果需要较高的数值精度，可以使用交错方向法（Alternating Direction Method, ADM）。ADM来源于增广Lagrange乘子法（Augmented Lagrange Multiplier, ALM），所以ADM又称为非精确增广Lagrange乘子法[78]。为简单起见，只考虑上面带等式约束的凸优化问题(8.101)。首先引入问题问题(8.101)的增广Lagrange函数：

$$L(\mathbf{x}, \lambda, \beta) = f(\mathbf{x}) + \langle \lambda, \mathbf{Ax} - \mathbf{b} \rangle + \frac{\beta}{2} \|\mathbf{Ax} - \mathbf{b}\|^2, \quad (8.117)$$

其中 $\lambda$ 为Lagrange乘子， $\beta$ 为惩罚系数。则有如下迭代：

---

**Algorithm 16** 增广Lagrange乘子法（ALM）

---

```

1: 初始化  $\mathbf{x}_0$ ,  $k = 0$ .
2: while 停止条件不满足 do
3:    $\mathbf{x}_{k+1} = \operatorname{argmin}_{\mathbf{x}} L(\mathbf{x}, \lambda_k, \beta)$ ;
4:    $\lambda_{k+1} = \lambda_k + \beta(\mathbf{Ax}_{k+1} - \mathbf{b})$ ;
5:    $k \leftarrow k + 1$ .
6: end while
7: 输出:  $\mathbf{x}_k$ .

```

---

ALM算法假定子问题：

$$\mathbf{x}_{k+1} = \operatorname{argmin}_{\mathbf{x}} L(\mathbf{x}, \lambda_k, \beta) \quad (8.118)$$

容易求解，这个假定并不一定成立。但是在应用中 $f(\mathbf{x})$ 经常有结构，比较常见的结构是可分解性，即：

$$f(\mathbf{x}) = f_1(\mathbf{y}) + f_2(\mathbf{z}), \quad \mathbf{x} = (\mathbf{y}^T, \mathbf{z}^T)^T. \quad (8.119)$$

此时约束 $\mathbf{Ax} = \mathbf{b}$ 可分解成

$$\mathbf{A}_1 \mathbf{y} + \mathbf{A}_2 \mathbf{z} = \mathbf{b}. \quad (8.120)$$

交错方向法就是利用了目标函数的可分解性。其迭代过程如下：

---

**Algorithm 17** 交错方向法 (ADM)

- 1: 初始化  $\mathbf{y}_0, \mathbf{z}_0, \lambda_0, k = 0$ .
  - 2: **while** 停止条件不满足 **do**
  - 3:  $\mathbf{y}_{k+1} = \underset{\mathbf{y}}{\operatorname{argmin}} L(\mathbf{y}, \mathbf{z}_k, \lambda_k, \beta) = \underset{\mathbf{y}}{\operatorname{argmin}} f_1(\mathbf{y}) + \frac{\beta}{2} \|\mathbf{A}_1 \mathbf{y} + \mathbf{A}_2 \mathbf{z}_k - \mathbf{b} + \lambda_k / \beta\|^2$ ;
  - 4:  $\mathbf{z}_{k+1} = \underset{\mathbf{z}}{\operatorname{argmin}} L(\mathbf{y}_{k+1}, \mathbf{z}_k, \lambda_k, \beta) = \underset{\mathbf{z}}{\operatorname{argmin}} f_2(\mathbf{z}) + \frac{\beta}{2} \|\mathbf{A}_1 \mathbf{y}_{k+1} + \mathbf{A}_2 \mathbf{z} - \mathbf{b} + \lambda_k / \beta\|^2$ ;
  - 5:  $\lambda_{k+1} = \lambda_k + \beta(\mathbf{A}_1 \mathbf{y}_{k+1} + \mathbf{A}_2 \mathbf{z}_{k+1} - \mathbf{b})$ ;
  - 6:  $k \leftarrow k + 1$ .
  - 7: **end while**
  - 8: 输出:  $(\mathbf{y}_k, \mathbf{z}, k)$ .
- 

ADM假定了如下形式的子问题：

$$\mathbf{y}_{k+1} = \underset{\mathbf{y}}{\operatorname{argmin}} f_1(\mathbf{y}) + \frac{\beta}{2} \|\mathbf{A}_1 \mathbf{y} - \mathbf{p}_k\|^2, \quad (8.121)$$

$$\mathbf{z}_{k+1} = \underset{\mathbf{z}}{\operatorname{argmin}} f_2(\mathbf{z}) + \frac{\beta}{2} \|\mathbf{A}_2 \mathbf{z} - \mathbf{q}_k\|^2, \quad (8.122)$$

是容易求解的，这在  $\mathbf{A}_1 = \mathbf{I}$ ,  $\mathbf{A}_2 = \mathbf{I}$  时是经常满足的，比如RPCA问题。但是对于更一般的问题， $\mathbf{A}_1, \mathbf{A}_2$  可能不是单位阵，导致上面的子问题不易求解。于是Lin等人[80]提出了线性化的ADM (Linearized ADM, LADM)，把(8.121)的二次项通过做一阶Taylor展开并加上一个近邻项 (proximal term) 来近似，把(8.121)改成：

$$\mathbf{y}_{k+1} = \underset{\mathbf{y}}{\operatorname{argmin}} f_1(\mathbf{y}) + \beta \langle \mathbf{A}_1^T (\mathbf{A}_1 \mathbf{y}_k - \mathbf{p}_k), \mathbf{y} - \mathbf{y}_k \rangle + \frac{\beta \eta_1}{2} \|\mathbf{y} - \mathbf{y}_k\|^2 \quad (8.123)$$

$$= \underset{\mathbf{y}}{\operatorname{argmin}} f_1(\mathbf{y}) + \frac{\beta \eta_1}{2} \|\mathbf{y} - \mathbf{y}_k + \mathbf{A}_1^T (\mathbf{A}_1 \mathbf{y}_k - \mathbf{p}_k) / \eta_1\|^2. \quad (8.124)$$

这样就可以很方便地得到  $\mathbf{y}_{k+1}$  了。对(8.122)也进行类似处理，可得：

$$\mathbf{z}_{k+1} = \underset{\mathbf{z}}{\operatorname{argmin}} f_2(\mathbf{z}) + \beta \langle \mathbf{A}_2^T (\mathbf{A}_2 \mathbf{z}_k - \mathbf{q}_k), \mathbf{z} - \mathbf{z}_k \rangle + \frac{\beta \eta_1}{2} \|\mathbf{z} - \mathbf{z}_k\|^2 \quad (8.125)$$

$$= \underset{\mathbf{z}}{\operatorname{argmin}} f_2(\mathbf{z}) + \frac{\beta \eta_2}{2} \|\mathbf{z} - \mathbf{z}_k + \mathbf{A}_2^T (\mathbf{A}_2 \mathbf{z}_k - \mathbf{q}_k) / \eta_2\|^2. \quad (8.126)$$

这样  $\mathbf{z}_{k+1}$  也可以很方便地得到了。如果采用动态调整的惩罚系数  $\beta$ :

$$\beta_{k+1} = \begin{cases} \rho \beta_k, & \text{if } \beta_k \max\{\|\mathbf{y}_{k+1} - \mathbf{y}_k\|, \|\mathbf{z}_{k+1} - \mathbf{z}_k\|\} / \|\mathbf{b}\| \leq \varepsilon_2, \\ \beta_k, & \text{otherwise,} \end{cases} \quad (8.127)$$

则有可能加速收敛，其中  $\rho \geq 1$ ,  $0 < \varepsilon_2 \ll 1$  为一阈值。 $\beta$  的初值的选择不要太大，应使得  $\beta_k$  在头几次迭代就增长。 $\rho$  的选择要适中，最好能使  $\beta_k$  随着迭代稳步增长。迭代在

下列条件满足时停止：

$$\begin{cases} \|\mathbf{A}_1\mathbf{y}_{k+1} + \mathbf{A}_2\mathbf{z}_{k+1} - \mathbf{b}\|/\|\mathbf{b}\| \leq \varepsilon_1, \\ \beta_k \max\{\|\mathbf{y}_{k+1} - \mathbf{y}_k\|, \|\mathbf{z}_{k+1} - \mathbf{z}_k\|\}/\|\mathbf{b}\| \leq \varepsilon_2. \end{cases} \quad (8.128)$$

Lin等人[80]证明了当 $\eta_i > \|\mathbf{A}_i\|_2^2$ 、序列 $\{\beta_k\}$ 不减且有上界时， $\{(\mathbf{y}_k, \mathbf{z}_k, \lambda_k)\}$ 收敛于问题

$$\min_{\mathbf{y}, \mathbf{z}} f_1(\mathbf{y}) + f_2(\mathbf{z}), \quad s.t. \quad \mathbf{A}_1\mathbf{y} + \mathbf{A}_2\mathbf{z} = \mathbf{b}, \quad (8.129)$$

的KKT点，即 $\{(\mathbf{y}_k, \mathbf{z}_k)\}$ 收敛到问题(8.129)的最优解。

## 8.10 Linearized ADM with Parallel Splitting and Adaptive Penalty (LADMPSAP)

The above LADM algorithm is only applicable for the two-block case. In practice we are very often faced with the multi-block case:

$$\min_{\mathbf{x}_1, \dots, \mathbf{x}_n} \sum_{i=1}^n f_i(\mathbf{x}_i), \quad s.t. \quad \sum_{i=1}^n \mathcal{A}_i(\mathbf{x}_i) = \mathbf{b}, \quad (8.130)$$

where  $\mathbf{x}_i$  and  $\mathbf{b}$  could be either vectors or matrices<sup>3</sup>,  $f_i$  is a closed proper convex function, and  $\mathcal{A}_i : \mathbb{R}^{d_i} \rightarrow \mathbb{R}^m$  is a linear mapping. Without loss of generality, we may assume that none of the  $\mathcal{A}_i$ 's is a zero mapping, the solution to  $\sum_{i=1}^n \mathcal{A}_i(\mathbf{x}_i) = \mathbf{b}$  is non-unique, and the mapping  $\mathcal{A}(\mathbf{x}_1, \dots, \mathbf{x}_n) \equiv \sum_{i=1}^n \mathcal{A}_i(\mathbf{x}_i)$  is onto<sup>4</sup>.

### 8.10.1 Exemplar Problems from Machine Learning

In this subsection, we present some examples of machine learning problems that can be formulated as the model problem (8.130).

#### 8.10.1.1 Latent Low-Rank Representation

Low-Rank Representation (LRR) [82, 86] is a recently proposed technique for robust subspace clustering and has been applied to many machine learning and computer vision problems. However, LRR works well only when the number of samples is more than the dimension of the samples, which may not be satisfied when the data dimension is high.

<sup>3</sup>In this paper we call each  $\mathbf{x}_i$  a “block” of variables because it may consist of multiple scalar variables. We will use bold capital letters if a block is known to be a matrix.

<sup>4</sup>The last two assumptions are equivalent to that the matrix  $\mathbf{A} \equiv (\mathbf{A}_1 \cdots \mathbf{A}_n)$  is not full column rank but full row rank, where  $\mathbf{A}_i$  is the matrix representation of  $\mathcal{A}_i$ .

So Liu et al. [85] proposed latent LRR to overcome this difficulty. The mathematical model of latent LRR is as follows:

$$\min_{\mathbf{Z}, \mathbf{L}, \mathbf{E}} \|\mathbf{Z}\|_* + \|\mathbf{L}\|_* + \mu \|\mathbf{E}\|_1, \quad s.t. \quad \mathbf{X} = \mathbf{XZ} + \mathbf{LX} + \mathbf{E}, \quad (8.131)$$

where  $\mathbf{X}$  is the data matrix, each column being a sample vector,  $\|\cdot\|_*$  is the nuclear norm [47], i.e., the sum of singular values, and  $\|\cdot\|_1$  is the  $\ell_1$  norm [22], i.e., the sum of absolute values of all entries. Latent LRR is to decompose data into principal feature  $\mathbf{XZ}$  and salient feature  $\mathbf{LX}$ , up to sparse noise  $\mathbf{E}$ .

### 8.10.1.2 Nonnegative Matrix Completion

Nonnegative matrix completion (NMC) [142] is a novel technique for dimensionality reduction, text mining, collaborative filtering, and clustering, etc. It can be formulated as:

$$\min_{\mathbf{X}, \mathbf{e}} \|\mathbf{X}\|_* + \frac{1}{2\mu} \|\mathbf{e}\|^2, \quad s.t. \quad \mathbf{b} = \mathcal{P}_\Omega(\mathbf{X}) + \mathbf{e}, \quad \mathbf{X} \geq 0, \quad (8.132)$$

where  $\mathbf{b}$  is the observed data in the matrix  $\mathbf{X}$  contaminated by noise  $\mathbf{e}$ ,  $\Omega$  is an index set,  $\mathcal{P}_\Omega$  is a linear mapping that selects those elements whose indices are in  $\Omega$ , and  $\|\cdot\|$  is the Frobenius norm. NMC is to recover the nonnegative low-rank matrix  $\mathbf{X}$  from the observed noisy data  $\mathbf{b}$ .

To see that the NMC problem can be reformulated as (8.130), we introduce an auxiliary variable  $\mathbf{y}$  and rewrite (8.132) as

$$\min_{\mathbf{X}, \mathbf{Y}, \mathbf{e}} \|\mathbf{X}\|_* + \chi_{\geq 0}(\mathbf{Y}) + \frac{1}{2\mu} \|\mathbf{e}\|^2, \quad s.t. \quad \begin{pmatrix} \mathcal{P}_\Omega(\mathbf{X}) \\ \mathbf{X} \end{pmatrix} - \begin{pmatrix} \mathbf{0} \\ \mathbf{Y} \end{pmatrix} + \begin{pmatrix} \mathbf{e} \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ \mathbf{0} \end{pmatrix}, \quad (8.133)$$

where  $\chi_{\geq 0}(\mathbf{Y}) = \begin{cases} 0, & \text{if } \mathbf{Y} \geq 0, \\ +\infty, & \text{otherwise,} \end{cases}$  is the characteristic function of the set of nonnegative matrices.

### 8.10.1.3 Group Sparse Logistic Regression with Overlap

Besides unsupervised learning models shown above, many supervised machine learning problems can also be written in the form of (8.130). For example, using logistic function as the loss function in the group LASSO with overlap [37, 64], one obtains the following model:

$$\min_{\mathbf{w}, b} \frac{1}{s} \sum_{i=1}^s \log(1 + \exp(-y_i(\mathbf{w}^T \mathbf{x}_i + b))) + \mu \sum_{j=1}^t \|\mathbf{s}_j \mathbf{w}\|, \quad (8.134)$$

where  $\mathbf{x}_i$  and  $y_i$ ,  $i = 1, \dots, s$ , are the training data and labels, respectively, and  $\mathbf{w}$  and  $b$  parameterize the linear classifier.  $\mathbf{s}_j$ ,  $j = 1, \dots, t$ , are the selection matrices, with only one 1 at each row and the rest entries are all zeros. The groups of entries,  $\mathbf{s}_j \mathbf{w}$ ,  $j = 1, \dots, t$ , may overlap each other. This model can also be considered as an extension of the group sparse logistic regression problem [100] to the case of overlapped groups.

Introducing  $\bar{\mathbf{w}} = (\mathbf{w}^T, b)^T$ ,  $\bar{\mathbf{x}}_i = (\mathbf{x}_i^T, 1)^T$ ,  $\mathbf{z} = (\mathbf{z}_1^T, \mathbf{z}_2^T, \dots, \mathbf{z}_t^T)^T$ , and  $\bar{\mathbf{s}} = (\mathbf{s}, \mathbf{0})$ , where  $\mathbf{s} = (\mathbf{s}_1^T, \dots, \mathbf{s}_t^T)^T$ , (8.134) can be rewritten as

$$\min_{\bar{\mathbf{w}}, \mathbf{z}} \frac{1}{s} \sum_{i=1}^s \log(1 + \exp(-y_i(\bar{\mathbf{w}}^T \bar{\mathbf{x}}_i))) + \mu \sum_{j=1}^t \|\mathbf{z}_j\|, \quad s.t. \quad \mathbf{z} = \bar{\mathbf{s}} \bar{\mathbf{w}}, \quad (8.135)$$

which is a special case of (8.130).

### 8.10.2 LADM with Parallel Splitting and Adaptive Penalty

Contrary to our intuition, the multi-block case is actually *fundamentally different* from the two-block one. A naive generalization of two-block LADM may not converge, e.g., for

$$\min_{\mathbf{x}_1, \dots, \mathbf{x}_n} \sum_{i=1}^n \|\mathbf{x}_i\|_1, \quad s.t. \quad \sum_{i=1}^n \mathbf{A}_i \mathbf{x}_i = \mathbf{b}, \quad (8.136)$$

where  $n \geq 5$ . So Lin et al. proposed Linearized ADM with Parallel Splitting and Adaptive Penalty (LADMPSAP) algorithm[81, 89] for solving (8.130).

### 8.10.3 LADMPSAP Algorithm

The modified algorithm for solving (8.130) consists of the following steps:

1. Update  $\mathbf{x}_i$ 's *in parallel*:

$$\mathbf{x}_i^{k+1} = \operatorname{argmin}_{\mathbf{x}_i} f_i(\mathbf{x}_i) + \frac{\sigma_i^{(k)}}{2} \left\| \mathbf{x}_i - \mathbf{x}_i^k + \mathcal{A}_i^*(\hat{\lambda}^k)/\sigma_i^{(k)} \right\|^2, \quad i = 1, \dots, n, \quad (8.137)$$

2. Update  $\lambda$ :

$$\lambda^{k+1} = \lambda^k + \beta_k \left( \sum_{i=1}^n \mathcal{A}_i(\mathbf{x}_i^{k+1}) - \mathbf{b} \right), \quad (8.138)$$

3. Update  $\beta$ :

$$\beta_{k+1} = \min(\beta_{\max}, \rho \beta_k), \quad (8.139)$$

where  $\sigma_i^{(k)} = \eta_i \beta_k$ ,  $\mathcal{A}_i^*$  is the adjoint operator of  $\mathcal{A}_i$ <sup>5</sup>,

$$\hat{\lambda}^k = \lambda^k + \beta_k \left( \sum_{i=1}^n \mathcal{A}_i(\mathbf{x}_i^k) - \mathbf{b} \right), \quad (8.140)$$

and

$$\rho = \begin{cases} \rho_0, & \text{if } \beta_k \max \left( \left\{ \sqrt{\eta_i} \|\mathbf{x}_i^{k+1} - \mathbf{x}_i^k\|, i = 1, \dots, n \right\} \right) / \|\mathbf{b}\| < \varepsilon_2, \\ 1, & \text{otherwise,} \end{cases} \quad (8.141)$$

with  $\rho_0 > 1$  being a constant and  $0 < \varepsilon_2 \ll 1$  being a threshold. Indeed, we replace

$$\tilde{\lambda}_i^k = \lambda^k + \beta_k \left( \sum_{j=1}^{i-1} \mathcal{A}_j(\mathbf{x}_j^{k+1}) + \sum_{j=i}^n \mathcal{A}_j(\mathbf{x}_j^k) - \mathbf{b} \right), \quad i = 1, \dots, n. \quad (8.142)$$

used in naively generalized LADM with  $\hat{\lambda}^k$  as (8.140), which is *independent* of  $i$ , and the rest procedures of the algorithm are all inherited, except that  $\eta_i$ 's have to be made larger (see Theorem 194). As now  $\mathbf{x}_i$ 's are updated in parallel and  $\beta_k$  changes adaptively, we call the new algorithm LADM with *parallel splitting* and *adaptive penalty* (LADMPSAP).

#### 8.10.4 Stopping Criteria

The iteration terminates when the following two conditions are met:

$$\left\| \sum_{i=1}^n \mathcal{A}_i(\mathbf{x}_i^{k+1}) - \mathbf{b} \right\| / \|\mathbf{b}\| < \varepsilon_1, \quad (8.143)$$

$$\beta_k \max \left( \left\{ \sqrt{\eta_i} \|\mathbf{x}_i^{k+1} - \mathbf{x}_i^k\|, i = 1, \dots, n \right\} \right) / \|\mathbf{b}\| < \varepsilon_2. \quad (8.144)$$

The above stopping criteria (8.143) and (8.144) are deduced from the KKT condition (i.e., the criteria (8.143) and (8.144) are for the feasibility and duality conditions, respectively). Indeed, the update rules (8.139) and (8.141) for  $\beta$  are hinted by the stopping criteria (8.143) and (8.144) such that the two errors are well balanced.

For better reference, we summarize the proposed LADMPSAP algorithm in Algorithm 18. For fast convergence, we suggest that  $\beta_0 = \alpha m \varepsilon_2$  and  $\alpha > 0$  and  $\rho_0 > 1$  should be chosen such that  $\beta_k$  increases steadily along with iteration.

#### 8.10.5 Global Convergence

For the global convergence of LADMPSAP, we have the following theorem, where we denote  $\{\mathbf{x}_i^k\} = \{\mathbf{x}_1^k, \dots, \mathbf{x}_n^k\}$  for simplicity.

---

<sup>5</sup>Defined as  $\langle \mathcal{A}_i^*(\mathbf{x}), \mathbf{y} \rangle = \langle \mathbf{x}, \mathcal{A}_i(\mathbf{y}) \rangle, \forall \mathbf{x}, \mathbf{y}$ .

---

**Algorithm 18** LADMPSAP for Solving (8.130)

**Initialize:** Set  $\varepsilon_1 > 0$ ,  $\varepsilon_2 > 0$ ,  $\beta_{\max} \gg 1 \gg \beta_0 > 0$ ,  $\eta_i > n\|\mathcal{A}_i\|^2$ ,  $\mathbf{x}_i^0$ ,  $i = 1, \dots, n$ ,  $\lambda^0$ .

**while** (8.143) or (8.144) is not satisfied **do**

**Step 1:** Compute  $\hat{\lambda}^k$  as (8.140).

**Step 2:** Update  $\mathbf{x}_i$ 's in parallel by solving

$$\mathbf{x}_i^{k+1} = \underset{\mathbf{x}_i}{\operatorname{argmin}} f_i(\mathbf{x}_i) + \frac{\eta_i \beta_k}{2} \left\| \mathbf{x}_i - \mathbf{x}_i^k + \mathcal{A}_i^*(\hat{\lambda}^k) / (\eta_i \beta_k) \right\|^2, \quad i = 1, \dots, n. \quad (8.145)$$

**Step 3:** Update  $\lambda$  by (8.138) and  $\beta$  by (8.139) and (8.141).

**end while**

---

**定理194. (Convergence of LADMPSAP)** If  $\{\beta_k\}$  is non-decreasing and upper bounded,  $\eta_i > n\|\mathcal{A}_i\|^2$ ,  $i = 1, \dots, n$ , then  $\{(\{\mathbf{x}_i^k\}, \lambda^k)\}$  generated by LADMPSAP converges to a KKT point of problem (8.130).

### 8.10.6 Enhanced Convergence Results

Theorem 194 is a convergence result for general convex programs (8.130), where  $f_i$ 's are general convex functions and hence  $\{\beta_k\}$  need to be bounded. Actually, almost all the existing theories on ADM and LADM even assumed a fixed  $\beta$ . For adaptive  $\beta_k$ , it will be more convenient if a user need not specify an upper bound on  $\{\beta_k\}$  because imposing a large upper bound essentially equals to allowing  $\{\beta_k\}$  to be unbounded. Since many machine learning problems choose  $f_i$ 's as matrix/vector norms, which result in bounded subgradients, we find that the boundedness assumption can be removed. Moreover, we can further prove the *sufficient and necessary* condition for global convergence.

**定理195. (Sufficient Condition for Global Convergence)** If  $\{\beta_k\}$  is non-decreasing and  $\sum_{k=1}^{+\infty} \beta_k^{-1} = +\infty$ ,  $\eta_i > n\|\mathcal{A}_i\|^2$ ,  $\partial f_i(\mathbf{x})$  is bounded,  $i = 1, \dots, n$ , then the sequence  $\{\mathbf{x}_i^k\}$  generated by LADMPSAP converges to an optimal solution to (8.130).

**定理196. (Necessary Condition for Global Convergence)** If  $\{\beta_k\}$  is non-decreasing,  $\eta_i > n\|\mathcal{A}_i\|^2$ ,  $\partial f_i(\mathbf{x})$  is bounded,  $i = 1, \dots, n$ , then  $\sum_{k=1}^{+\infty} \beta_k^{-1} = +\infty$  is also the necessary condition for the global convergence of  $\{\mathbf{x}_i^k\}$  generated by LADMPSAP to an optimal solution to (8.130).

### 8.10.7 Convergence Rate

In this subsection, based on a simple optimality measure we give a simple proof for the convergence rate of LADMPSAP. For simplicity, we define  $\mathbf{x} = (\mathbf{x}_1^T, \dots, \mathbf{x}_n^T)^T$ ,

$\mathbf{x}^* = ((\mathbf{x}_1^*)^T, \dots, (\mathbf{x}_2^*)^T)^T$  and  $f(\mathbf{x}) = \sum_{i=1}^n f_i(\mathbf{x}_i)$ , where  $(\mathbf{x}_1^*, \dots, \mathbf{x}_2^*, \lambda^*)$  is a KKT point of (8.130).

Then we have the following convergence rate theorem for LADMPSAP in an ergodic sense.

**定理197. (Convergence Rate of LADMPSAP)** Define  $\bar{\mathbf{x}}^K = \sum_{k=0}^K \gamma_k \mathbf{x}^{k+1}$ , where  $\gamma_k = \beta_k^{-1} / \sum_{j=0}^K \beta_j^{-1}$ . Then

$$f(\bar{\mathbf{x}}^K) - f(\mathbf{x}^*) + \sum_{i=1}^n \langle \mathcal{A}_i^*(\lambda^*), \bar{\mathbf{x}}_i^K - \mathbf{x}_i^* \rangle + \frac{\alpha \beta_0}{2} \left\| \sum_{i=1}^n \mathcal{A}_i(\bar{\mathbf{x}}_i^K) - \mathbf{b} \right\|^2 \leq C_0 / \left( 2 \sum_{k=0}^K \beta_k^{-1} \right), \quad (8.146)$$

where  $\alpha^{-1} = (n+1) \max \left( 1, \left\{ \frac{\|\mathcal{A}_i\|^2}{\eta_i - n\|\mathcal{A}_i\|^2}, i = 1, \dots, n \right\} \right)$  and  $C_0 = \sum_{i=1}^n \eta_i \|\mathbf{x}_i^0 - \mathbf{x}_i^*\|^2 + \beta_0^{-2} \|\lambda^0 - \lambda^*\|^2$ .

### 8.10.8 Practical LADMPSAP for Convex Programs with Convex Set Constraints

In real applications, we are often faced with convex programs with convex set constraints:

$$\min_{\mathbf{x}_1, \dots, \mathbf{x}_n} \sum_{i=1}^n f_i(\mathbf{x}_i), \quad s.t. \quad \sum_{i=1}^n \mathcal{A}_i(\mathbf{x}_i) = \mathbf{b}, \quad \mathbf{x}_i \in X_i, \quad i = 1, \dots, n, \quad (8.147)$$

where  $X_i \subseteq \mathbb{R}^{d_i}$  is a closed convex set. In this section, we assume that the projections onto  $X_i$ 's are all easily computable. For many convex sets used in machine learning, such an assumption is valid, e.g., when  $X_i$ 's are nonnegative cones or positive semi-definite cones. In the following, we discuss how to solve (8.147) efficiently. For simplicity, we assume  $X_i \neq \mathbb{R}^{d_i}, \forall i$ . Finally, we assume that  $\mathbf{b}$  is an interior point of  $\sum_{i=1}^n \mathcal{A}_i(X_i)$ .

We introduce auxiliary variables  $\mathbf{x}_{n+i}$  to convert  $\mathbf{x}_i \in X_i$  into  $\mathbf{x}_i = \mathbf{x}_{n+i}$  and  $\mathbf{x}_{n+i} \in X_i, i = 1, \dots, n$ . Then (8.147) can be reformulated as:

$$\min_{\mathbf{x}_1, \dots, \mathbf{x}_{2n}} \sum_{i=1}^{2n} f_i(\mathbf{x}_i), \quad s.t. \quad \sum_{i=1}^{2n} \hat{\mathcal{A}}_i(\mathbf{x}_i) = \hat{\mathbf{b}}, \quad (8.148)$$

where

$$f_{n+i}(\mathbf{x}) \equiv \chi_{X_i}(\mathbf{x}) = \begin{cases} 0, & \text{if } \mathbf{x} \in X_i, \\ +\infty, & \text{otherwise,} \end{cases}$$

is the characteristic function of  $X_i$ ,

$$\hat{\mathcal{A}}_i(\mathbf{x}_i) = \begin{pmatrix} \mathcal{A}_i(\mathbf{x}_i) \\ 0 \\ \vdots \\ \mathbf{x}_i \\ \vdots \\ 0 \end{pmatrix}, \hat{\mathcal{A}}_{n+i}(\mathbf{x}_{n+i}) = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ -\mathbf{x}_{n+i} \\ \vdots \\ 0 \end{pmatrix}, \text{ and } \hat{\mathbf{b}} = \begin{pmatrix} \mathbf{b} \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad (8.149)$$

where  $i = 1, \dots, n$ .

The adjoint operator  $\hat{\mathcal{A}}_i^*$  is

$$\hat{\mathcal{A}}_i^*(\mathbf{y}) = \mathcal{A}_i^*(\mathbf{y}_1) + \mathbf{y}_{i+1}, \quad \hat{\mathcal{A}}_{n+i}^*(\mathbf{y}) = -\mathbf{y}_{i+1}, \quad i = 1, \dots, n, \quad (8.150)$$

where  $\mathbf{y}_i$  is the  $i$ -th sub-vector of  $\mathbf{y}$ , partitioned according to the sizes of  $\mathbf{b}$  and  $\mathbf{x}_i$ ,  $i = 1, \dots, n$ .

Then LADMP SAP can be applied to solve problem (8.148). The Lagrange multiplier  $\lambda$  and the auxiliary multiplier  $\hat{\lambda}$  are respectively updated as

$$\lambda_1^{k+1} = \lambda_1^k + \beta_k \left( \sum_{i=1}^n \mathcal{A}_i(\mathbf{x}_i^{k+1}) - \mathbf{b} \right), \quad \lambda_{i+1}^{k+1} = \lambda_{i+1}^k + \beta_k (\mathbf{x}_i^{k+1} - \mathbf{x}_{n+i}^{k+1}), \quad (8.151)$$

$$\hat{\lambda}_1^k = \lambda_1^k + \beta_k \left( \sum_{i=1}^n \mathcal{A}_i(\mathbf{x}_i^k) - \mathbf{b} \right), \quad \hat{\lambda}_{i+1}^k = \lambda_{i+1}^k + \beta_k (\mathbf{x}_i^k - \mathbf{x}_{n+i}^k), \quad (8.152)$$

and  $\mathbf{x}_i$  is updated as (see (8.137))

$$\mathbf{x}_i^{k+1} = \underset{\mathbf{x}}{\operatorname{argmin}} f_i(\mathbf{x}) + \frac{\eta_i \beta_k}{2} \left\| \mathbf{x} - \mathbf{x}_i^k + [\mathcal{A}_i^*(\hat{\lambda}_1^k) + \hat{\lambda}_{i+1}^k]/(\eta_i \beta_k) \right\|^2, \quad (8.153)$$

$$\begin{aligned} \mathbf{x}_{n+i}^{k+1} &= \underset{\mathbf{x} \in X_i}{\operatorname{argmin}} \frac{\eta_{n+i} \beta_k}{2} \left\| \mathbf{x} - \mathbf{x}_{n+i}^k - \hat{\lambda}_{i+1}^k / (\eta_{n+i} \beta_k) \right\|^2 \\ &= \pi_{X_i} \left( \mathbf{x}_{n+i}^k + \hat{\lambda}_{i+1}^k / (\eta_{n+i} \beta_k) \right), \end{aligned} \quad (8.154)$$

where  $\pi_{X_i}$  is the projection onto  $X_i$  and  $i = 1, \dots, n$ .

**练习198.** Prove that the adjoint operator  $\hat{\mathcal{A}}_i^*$  is (8.150).

As for the choice of  $\eta_i$ 's, although we can simply apply Theorem 194 to assign their values as  $\eta_i > 2n(\|\mathcal{A}_i\|^2 + 1)$  and  $\eta_{n+i} > 2n$ ,  $i = 1, \dots, n$ , such choices are too pessimistic. As  $\eta_i$ 's are related to the magnitudes of the differences in  $\mathbf{x}_i^{k+1}$  from  $\mathbf{x}_i^k$ , we had better provide tighter estimate on  $\eta_i$ 's in order to achieve faster convergence. Actually, we have the following better result.

---

**Algorithm 19** LADMPSAP for (8.148), also a Practical Algorithm for (8.147).

**Initialize:** Set  $\varepsilon_1 > 0$ ,  $\varepsilon_2 > 0$ ,  $\beta_{\max} \gg 1 \gg \beta_0 > 0$ ,  $\eta_i > n\|\mathcal{A}_i\|^2 + 2$ ,  $\eta_{n+i} > 2$ ,  $\mathbf{x}_i^0$ ,  $\mathbf{x}_{n+i}^0 = \mathbf{x}_i^0$ ,  $i = 1, \dots, n$ ,  $\lambda^0 = ((\lambda_1^0)^T, \dots, (\lambda_{n+1}^0)^T)^T$ .

**while** (8.143) or (8.144) is not satisfied **do**

**Step 1:** Compute  $\hat{\lambda}^k$  as (8.152).

**Step 2:** Update  $\mathbf{x}_i$ ,  $i = 1, \dots, 2n$ , in parallel as (8.153)-(8.154).

**Step 3:** Update  $\lambda$  by (8.152) and  $\beta$  by (8.139) and (8.141).

**end while**

(Note that in (8.141), (8.143), and (8.144),  $n$  and  $\mathcal{A}_i$  should be replaced by  $2n$  and  $\hat{\mathcal{A}}_i$ , respectively.)

---

**定理199.** For problem (8.148), if  $\{\beta_k\}$  is non-decreasing and upper bounded and  $\eta_i$ 's are chosen as  $\eta_i > n\|\mathcal{A}_i\|^2 + 2$  and  $\eta_{n+i} > 2$ ,  $i = 1, \dots, n$ , then the sequence  $\{(\{\mathbf{x}_i^k\}, \lambda^k)\}$  generated by LADMPSAP converges to a KKT point of problem (8.148).

Finally, we summarize LADMPSAP for problem (8.148) in Algorithm 19, which is a practical algorithm for solving (8.147).

**备注200.** Analogs of Theorems 195 and 196 are also true for Algorithm 19 although  $\partial f_{n+i}$ 's are unbounded, thanks to our assumptions that all  $\partial f_i$ ,  $i = 1, \dots, n$ , are bounded and  $\mathbf{b}$  is an interior point of  $\sum_{i=1}^n \mathcal{A}_i(X_i)$ . Consequently,  $\beta_{\max}$  can also be removed if all  $\partial f_i$ ,  $i = 1, \dots, n$ , are bounded.

**备注201.** Since Algorithm 19 is an application of Algorithm 18 to problem (8.148), only with refined parameter estimation, its convergence rate in an ergodic sense is also  $O\left(1/\sum_{k=0}^K \beta_k^{-1}\right)$ , where  $K$  is the number of iterations.

**练习202.** Submit codes for the following problems. The parameters  $\beta_0$  and  $\rho_0$  have to be tuned to make the algorithms reasonably fast, where  $\varepsilon_1 = 1E - 4$  and  $\varepsilon_2 = 1E - 3$  are fixed.

1. Use LADMPSAP to solve the latent LRR problem:

$$\min_{\mathbf{L}, \mathbf{Z}, \mathbf{E}} \|\mathbf{L}\|_* + \|\mathbf{Z}\|_* + \lambda \|\mathbf{E}\|_1, \quad s.t. \quad \mathbf{X} = \mathbf{L}\mathbf{X} + \mathbf{Z}\mathbf{X} + \mathbf{E},$$

where  $\mathbf{X}$  is randomly generalized.

2. Use practical LADMPSAP to solve the nonnegative latent LRR problem, where  $\mathbf{L} \geq 0$ ,  $\mathbf{Z} \geq 0$ .

## 8.11 Proximal LADMPSAP for Even More General Convex Programs

In LADMPSAP we have assumed that the subproblems (8.137) are easily solvable. In many machine learning problems, the functions  $f_i$ 's are often matrix or vector norms or characteristic functions of convex sets. So this assumption often holds. Nonetheless, this assumption is not always true, e.g., when  $f_i$  is the logistic loss function (see (8.135)). So in this section we aim at generalizing LADMPSAP to solve even more general convex programs (8.130).

We are interested in the case that  $f_i$  can be decomposed into two components:

$$f_i(\mathbf{x}_i) = g_i(\mathbf{x}_i) + h_i(\mathbf{x}_i), \quad (8.155)$$

where both  $g_i$  and  $h_i$  are convex,  $g_i$  is  $C^{1,1}$ :

$$\|\nabla g_i(\mathbf{x}) - \nabla g_i(\mathbf{y})\| \leq L_i \|\mathbf{x} - \mathbf{y}\|, \quad \forall x, y \in \mathbb{R}^{d_i}, \quad (8.156)$$

and  $h_i$  may not be differentiable but its proximal operation is easily solvable. For brevity, we call  $L_i$  the Lipschitz constant of  $\nabla g_i$ .

**定理203** (Baillon-Haddad). *Let  $\mathcal{H}$  be a real Hilbert space,  $f : \mathcal{H} \rightarrow \mathbb{R}$  be convex, Fréchet differentiable on  $\mathcal{H}$ , and such that  $\nabla f$  is  $\beta$ -Lipschitz continuous for some  $\beta \in (0, +\infty)$ . Then  $\nabla f$  is  $1/\beta$ -coercive, i.e.*

$$\beta \langle x - y, \nabla f(x) - \nabla f(y) \rangle \geq \|\nabla f(x) - \nabla f(y)\|^2, \quad \forall x, y \in \mathcal{H}.$$

Recall that in each iteration of LADMPSAP, we have to solve subproblem (8.137). Since now we do not assume that the proximal operation of  $f_i$  is easily solvable, we may have difficulty in solving subproblem (8.137). By (8.155), we write down (8.137) as

$$\mathbf{x}_i^{k+1} = \underset{\mathbf{x}_i}{\operatorname{argmin}} h_i(\mathbf{x}_i) + g_i(\mathbf{x}_i) + \frac{\sigma_i^{(k)}}{2} \left\| \mathbf{x}_i - \mathbf{x}_i^k + \mathcal{A}_i^\dagger(\hat{\lambda}^k)/\sigma_i^{(k)} \right\|^2, \quad i = 1, \dots, n, \quad (8.157)$$

Since  $g_i(\mathbf{x}_i) + \frac{\sigma_i^{(k)}}{2} \left\| \mathbf{x}_i - \mathbf{x}_i^k + \mathcal{A}_i^\dagger(\hat{\lambda}^k)/\sigma_i^{(k)} \right\|^2$  is  $C^{1,1}$ , we may also linearize it at  $\mathbf{x}_i^k$  and add a proximal term. Such an idea leads to the following updating scheme of  $\mathbf{x}_i$ :

$$\begin{aligned} \mathbf{x}_i^{k+1} &= \underset{\mathbf{x}_i}{\operatorname{argmin}} h_i(\mathbf{x}_i) + g_i(\mathbf{x}_i^k) + \frac{\sigma_i^{(k)}}{2} \left\| \mathcal{A}_i^\dagger(\hat{\lambda}^k)/\sigma_i^{(k)} \right\|^2 \\ &\quad + \langle \nabla g_i(\mathbf{x}_i^k) + \mathcal{A}_i^\dagger(\hat{\lambda}^k), \mathbf{x}_i - \mathbf{x}_i^k \rangle + \frac{\tau_i^{(k)}}{2} \left\| \mathbf{x}_i - \mathbf{x}_i^k \right\|^2 \\ &= \underset{\mathbf{x}_i}{\operatorname{argmin}} h_i(\mathbf{x}_i) + \frac{\tau_i^{(k)}}{2} \left\| \mathbf{x}_i - \mathbf{x}_i^k + \frac{1}{\tau_i^{(k)}} [\mathcal{A}_i^\dagger(\hat{\lambda}^k) + \nabla g_i(\mathbf{x}_i^k)] \right\|^2, \end{aligned} \quad (8.158)$$

---

**Algorithm 20** Proximal LADMP SAP for Solving (8.130) with  $f_i$  Satisfying (8.155).

---

**Initialize:** Set  $\rho_0 > 1$ ,  $\beta_0 > 0$ ,  $\lambda^0$ ,  $T_i \geq L_i$ ,  $\eta_i > n\|\mathcal{A}_i\|^2$ ,  $\mathbf{x}_i^0$ ,  $i = 1, \dots, n$ .

**while** (8.160) or (8.161) is not satisfied **do**

**Step 1:** Compute  $\hat{\lambda}^k$  as (8.140).

**Step 2:** Update  $\mathbf{x}_i$ 's in parallel by solving

$$\mathbf{x}_i^{k+1} = \underset{\mathbf{x}_i}{\operatorname{argmin}} h_i(\mathbf{x}_i) + \frac{\tau_i^{(k)}}{2} \left\| \mathbf{x}_i - \mathbf{x}_i^k + \frac{1}{\tau_i^{(k)}} [\mathcal{A}_i^\dagger(\hat{\lambda}^k) + \nabla g_i(\mathbf{x}_i^k)] \right\|^2, \quad i = 1, \dots, n, \quad (8.162)$$

where  $\tau_i^{(k)} = T_i + \beta_k \eta_i$ .

**Step 3:** Update  $\lambda$  by (8.138) and  $\beta$  by (8.139) with  $\rho$  defined in (8.159).

**end while**

---

where  $i = 1, \dots, n$ . The choice of  $\tau_i^{(k)}$  is presented in Theorem 204, i.e.  $\tau_i^{(k)} = T_i + \beta_k \eta_i$ , where  $T_i \geq L_i$  and  $\eta_i > n\|\mathcal{A}_i\|^2$  are both positive constants.

By our assumption on  $h_i$ , the above subproblems are easily solvable. The update of Lagrange multiplier  $\lambda$  and  $\beta$  are still respectively goes as (8.138) and (8.139) but with

$$\rho = \begin{cases} \rho_0, & \text{if } \max \left( \left\{ \|\mathcal{A}_i\|^{-1} \left\| \nabla g_i(\mathbf{x}_i^{k+1}) - \nabla g_i(\mathbf{x}_i^k) - \tau_i^{(k)} (\mathbf{x}_i^{k+1} - \mathbf{x}_i^k) \right\|, \right. \right. \\ & \left. \left. i = 1, \dots, n \right\} \right) / \|\mathbf{b}\| < \varepsilon_2, \\ 1, & \text{otherwise.} \end{cases} \quad (8.159)$$

The iteration terminates when the following two conditions are met:

$$\left\| \sum_{i=1}^n \mathcal{A}_i(\mathbf{x}_i^{k+1}) - \mathbf{b} \right\| / \|\mathbf{b}\| < \varepsilon_1, \quad (8.160)$$

$$\max \left( \left\{ \|\mathcal{A}_i\|^{-1} \left\| \nabla g_i(\mathbf{x}_i^{k+1}) - \nabla g_i(\mathbf{x}_i^k) - \tau_i^{(k)} (\mathbf{x}_i^{k+1} - \mathbf{x}_i^k) \right\|, \right. \right. \\ \left. \left. i = 1, \dots, n \right\} \right) / \|\mathbf{b}\| < \varepsilon_2. \quad (8.161)$$

These two conditions are also deduced from the KKT conditions.

We call the above algorithm as proximal LADMP SAP and summarize it in Algorithm 20.

As for the convergence of proximal LADMP SAP, we have the following theorem.

**定理204. (Convergence of Proximal LADMP SAP)** If  $\beta_k$  is non-decreasing and upper bounded,  $\tau_i^{(k)} = T_i + \beta_k \eta_i$ , where  $T_i \geq L_i$  and  $\eta_i > n\|\mathcal{A}_i\|^2$  are both positive constants,  $i = 1, \dots, n$ , then  $\{(\{\mathbf{x}_i^k\}, \lambda^k)\}$  generated by proximal LADMP SAP converge to a KKT point of problem (8.130).

We further have the following convergence rate theorem for proximal LADMP SAP in an ergodic sense.

**定理205. (Convergence Rate of Proximal LADMP SAP)** Define  $\bar{\mathbf{x}}_i^K = \sum_{k=0}^K \gamma_k \mathbf{x}_i^{k+1}$ , where  $\gamma_k = \beta_k^{-1} / \sum_{j=0}^K \beta_j^{-1}$ . Then the following inequality holds for  $\bar{\mathbf{x}}_i^K$ :

$$\begin{aligned} & \sum_{i=1}^n \left( f_i(\bar{\mathbf{x}}_i^K) - f_i(\mathbf{x}_i^*) + \langle \mathcal{A}_i^\dagger(\lambda^*), \bar{\mathbf{x}}_i^K - \mathbf{x}_i^* \rangle \right) + \frac{\alpha\beta_0}{2} \left\| \sum_{i=1}^n \mathcal{A}_i(\bar{\mathbf{x}}_i^K) - \mathbf{b} \right\|^2 \\ & \leq C_0 / \sum_{k=0}^K 2\beta_k^{-1}, \end{aligned} \quad (8.163)$$

where  $\alpha^{-1} = (n+1) \max \left( 1, \left\{ \frac{\|\mathcal{A}_i\|^2}{\eta_i - n\|\mathcal{A}_i\|^2}, i = 1, \dots, n \right\} \right)$  and  $C_0 = \sum_{i=1}^n \beta_0^{-1} \tau_i^{(0)} \|\mathbf{x}_i^0 - \mathbf{x}_i^*\|^2 + \beta_0^{-2} \|\lambda^0 - \lambda^*\|^2$ .

When there are extra convex set constraints,  $\mathbf{x}_i \in X_i$ ,  $i = 1, \dots, n$ , we can also introduce auxiliary variables as in Section 8.10.8 and have an analogy of Theorems 199 and 197.

**定理206.** For problem (8.148), where  $f_i$  is described at the beginning of Section 8.11, if  $\beta_k$  is non-decreasing and upper bounded and  $\tau_i^{(k)} = T_i + \eta_i \beta_k$ , where  $T_i \geq L_i$ ,  $T_{n+i} = 0$ ,  $\eta_i > n\|\mathcal{A}_i\|^2 + 2$ , and  $\eta_{n+i} > 2$ ,  $i = 1, \dots, n$ , then  $\{\{\mathbf{x}_i^k\}, \lambda^k\}$  generated by proximal LADMP SAP converge to a KKT point of problem (8.148). The convergence rate in an ergodic sense is also  $O\left(1/\sum_{k=0}^K \beta_k^{-1}\right)$ , where  $K$  is the number of iterations.

## 8.12 Numerical Results

In this section, we test the performance of LADMP SAP on three specific examples of problem (8.130), i.e., Latent Low-Rank Representation (see (8.131)), Nonnegative Matrix Completion (see (8.132)), and Group Sparse Logistic Regression with Overlap (see (8.135)).

### 8.12.1 Solving Latent Low-Rank Representation

We first solve the latent LRR problem [85] (8.131). In order to test LADMP SAP and related algorithms with data whose characteristics are controllable, we follow [82] to generate synthetic data, which are parameterized as  $(s, p, d, \tilde{r})$ , where  $s$ ,  $p$ ,  $d$ , and  $\tilde{r}$  are the number of independent subspaces, points in each subspace, and ambient and intrinsic dimensions, respectively. The number of scale variables and constraints is  $(sp) \times d$ .

As first order methods are popular for solving convex programs in machine learning [17], here we compare LADMPSAP with several conceivable first order algorithms, including APG [8], naive ADM, naive LADM, LADMGB, and LADMPS. Naive ADM and naive LADM are generalizations of ADM and LADM, respectively, which are straightforwardly generalized from two variables to multiple variables, as discussed in Section 8.10.2. Naive ADM is applied to solve (8.131) after rewriting the constraint of (8.131) as  $\mathbf{X} = \mathbf{XP} + \mathbf{QX} + \mathbf{E}$ ,  $\mathbf{P} = \mathbf{Z}$ ,  $\mathbf{Q} = \mathbf{L}$ . For LADMPS,  $\beta_k$  is fixed in order to show the effectiveness of adaptive penalty. The parameters of APG and ADM are the same as those in [79] and [85], respectively. For LADM, we follow the suggestions in [144] to fix its penalty parameter  $\beta$  at  $2.5/\min(d, sp)$ , where  $d \times sp$  is the size of  $\mathbf{X}$ . For LADMGB, as there is no suggestion in [60] on how to choose a fixed  $\beta$ , we simply set it the same as that in LADM. The rest of the parameters are the same as those suggested in [59]. We fix  $\beta = \sigma_{\max}(\mathbf{X}) \min(d, sp) \varepsilon_2$  in LADMPS and set  $\beta_0 = \sigma_{\max}(\mathbf{X}) \min(d, sp) \varepsilon_2$  and  $\rho_0 = 10$  in LADMPSAP. For LADMPSAP, we also set  $\eta_Z = \eta_L = 1.02 \times 3\sigma_{\max}^2(\mathbf{X})$ , where  $\eta_Z$  and  $\eta_L$  are the parameters  $\eta_i$ 's in Algorithm 18 for  $\mathbf{Z}$  and  $\mathbf{L}$ , respectively. For the stopping criteria,  $\|\mathbf{XZ}^k + \mathbf{L}^k \mathbf{X} + \mathbf{E}^k - \mathbf{X}\|_F / \|\mathbf{X}\|_F \leq \varepsilon_1$  and  $\max(\|\mathbf{Z}^k - \mathbf{Z}^{k-1}\|_F, \|\mathbf{L}^k - \mathbf{L}^{k-1}\|_F, \|\mathbf{E}^k - \mathbf{E}^{k-1}\|_F) / \|\mathbf{X}\|_F \leq \varepsilon_2$ , with  $\varepsilon_1 = 10^{-3}$  and  $\varepsilon_2 = 10^{-4}$  are used for all the algorithms. For the parameter  $\mu$  in (8.131), we empirically set it as  $\mu = 0.01$ . To measure the relative errors in the solutions we run LADMPSAP 2000 iterations with  $\rho_0 = 1.01$  to obtain the estimated ground truth solution  $(\mathbf{Z}^*, \mathbf{L}^*, \mathbf{E}^*)$ . The experiments are run and timed on a notebook computer with an Intel Core i7 2.00 GHz CPU and 6GB memory, running Windows 7 and Matlab 7.13.

Table 8.1 shows the results of related algorithms. We can see that LADMPS and LADMPSAP are faster and more numerically accurate than LADMGB, and LADMPSAP is even faster than LADMPS thanks to the adaptive penalty. Moreover, naive ADM and naive LADM have relatively poorer numerical accuracy, possibly due to converging to wrong solutions. The numerical accuracy of APG is also worse than those of LADMPS and LADMPSAP because it only solves an approximate problem by adding the constraint to the objective function as penalty. Note that although we do not require  $\{\beta_k\}$  to be bounded, this does not imply that  $\beta_k$  will grow infinitely. As a matter of fact, when LADMPSAP terminates the final values of  $\beta_k$  are 21.1567, 42.2655, and 81.4227 for the three data settings, respectively.

We then test the performance of the above six algorithms on the Hopkins155 database [126], which consists of 156 sequences, each having 39 to 550 data vectors drawn from two or three motions. For computational efficiency, we preprocess the data by projecting them to be 5-dimensional using PCA. We test all algorithms with  $\mu = 2.4$ , which is

the best parameter for LRR on this database [82]. Table 8.2 shows the results on the Hopkins155 database. We can also see that LADMPSAP is faster than other methods in comparison. In particular, LADMPSAP is faster than LADMPS, which uses a fixed  $\beta$ . This testify to the advantage of using an adaptive penalty.

表 8.1: Comparisons of APG, naive ADM (nADM), naive LADM (nLADM), LADMGB, LADMPS, and LADMPSAP on the latent LRR problem (8.131). The quantities include computing time (in seconds), number of iterations, relative errors, and clustering accuracy (in percentage). They are averaged over 10 runs.

| $(s, p, d, \tilde{r})$ | Method   | Time          | #Iter.     | $\frac{\ \hat{\mathbf{Z}} - \mathbf{Z}^*\ }{\ \mathbf{Z}^*\ }$ | $\frac{\ \hat{\mathbf{L}} - \mathbf{L}^*\ }{\ \mathbf{L}^*\ }$ | $\frac{\ \hat{\mathbf{E}} - \mathbf{E}^*\ }{\ \mathbf{E}^*\ }$ | Acc. |
|------------------------|----------|---------------|------------|--|--|--|------|
| $(5, 50, 250, 5)$      | APG      | 18.20         | 236        | 0.3389   | 0.3167   | 0.4500   | 95.6 |
|                        | nADM     | 16.32         | 172        | 0.3993   | 0.3928   | 0.5592   | 95.6 |
|                        | nLADM    | 21.34         | 288        | 0.4553   | 0.4408   | 0.5607   | 95.6 |
|                        | LADMGB   | 24.10         | 290        | 0.4520   | 0.4355   | 0.5610   | 95.6 |
|                        | LADMPS   | 17.15         | 232        | 0.0163   | 0.0139   | <b>0.0446</b>  | 95.6 |
|                        | LADMPSAP | <b>8.04</b>   | <b>109</b> | <b>0.0089</b>  | <b>0.0083</b>  | 0.0464   | 95.6 |
| $(10, 50, 500, 5)$     | APG      | 85.03         | 234        | 0.1020   | 0.0844   | 0.7161   | 95.8 |
|                        | nADM     | 78.27         | 170        | 0.0928   | 0.1026   | 0.6636   | 95.8 |
|                        | nLADM    | 181.42        | 550        | 0.2077   | 0.2056   | 0.6623   | 95.8 |
|                        | LADMGB   | 214.94        | 550        | 0.1877   | 0.1848   | 0.6621   | 95.8 |
|                        | LADMPS   | 64.65         | 200        | 0.0167   | 0.0089   | 0.1059   | 95.8 |
|                        | LADMPSAP | <b>37.85</b>  | <b>117</b> | <b>0.0122</b>  | <b>0.0055</b>  | <b>0.0780</b>  | 95.8 |
| $(20, 50, 1000, 5)$    | APG      | 544.13        | 233        | 0.0319   | 0.0152   | 0.2126   | 95.2 |
|                        | nADM     | 466.78        | 166        | 0.0501   | 0.0433   | 0.2676   | 95.2 |
|                        | nLADM    | 1888.44       | 897        | 0.1783   | 0.1746   | 0.2433   | 95.2 |
|                        | LADMGB   | 2201.37       | 897        | 0.1774   | 0.1736   | 0.2434   | 95.2 |
|                        | LADMPS   | 367.68        | 177        | 0.0151   | 0.0105   | 0.0872   | 95.2 |
|                        | LADMPSAP | <b>260.22</b> | <b>125</b> | <b>0.0106</b>  | <b>0.0041</b>  | <b>0.0671</b>  | 95.2 |

### 8.12.2 Solving Nonnegative Matrix Completion

This subsection evaluates the performance of the practical LADMPSAP proposed in Section 8.10.8 for solving nonnegative matrix completion [142] (8.132).

表 8.2: Comparisons of APG, naive ADM (nADM), naive LADM (nLADM), LADMGB, LADMPS, and LADMPSAP on the Hopkins155 database. The quantities include average computing time, average number of iterations, and average classification errors on all 156 sequences.

| Method   | Time (seconds) | #Iteration | Error (%) |
|----------|----------------|------------|-----------|
| APG      | 10.37          | 67         | 8.33      |
| nADM     | 24.76          | 144        | 8.33      |
| nLADM    | 15.50          | 112        | 8.33      |
| LADMGB   | 16.05          | 113        | 8.36      |
| LADMPS   | 15.58          | 113        | 8.33      |
| LADMPSAP | <b>3.80</b>    | <b>26</b>  | 8.33      |

We first evaluate the numerical performance on synthetic data to demonstrate the superiority of practical LADMPSAP over the conventional LADM<sup>6</sup> [144]. The nonnegative low-rank matrix  $\mathbf{x}_0$  is generated by truncating the singular values of a randomly generated matrix. As LADM cannot handle the nonnegativity constraint, it actually solve the standard matrix completion problem, i.e., (8.132) without the nonnegativity constraint. For LADMPSAP, we follow the conditions in Theorem 199 to set  $\eta_i$ 's and set the rest of the parameters the same as those in Section 8.12.1. The stopping tolerances are set as  $\varepsilon_1 = \varepsilon_2 = 10^{-5}$ . The numerical comparison is shown in Table 8.3, where the relative nonnegative feasibility (FA) is defined as [142]:

$$\text{FA} := \|\min(\hat{\mathbf{X}}, 0)\| / \|\mathbf{X}_0\|,$$

in which  $\mathbf{X}_0$  is the ground truth and  $\hat{\mathbf{X}}$  is the computed solution. It can be seen that the numerical performance of LADMPSAP is much better than that of LADM, thus again verifies the efficiency of our proposed parallel splitting and adaptive penalty scheme for enhancing ADM/LADM type algorithms.

We then consider the image inpainting problem, which is to fill in the missing pixel values of a corrupted image. As the pixel values are nonnegative, the image inpainting problem can be formulated as the NMC problem. To prepare a low-rank image, we also truncate the singular values of a  $1024 \times 1024$  grayscale image “man”<sup>7</sup> to obtain an image of rank 40, shown in Fig. 8.1 (a)-(b). The corrupted image is generated from the original

<sup>6</sup>Code available at [http://math.nju.edu.cn/~jfyang/IADM\\_NNLS/index.html](http://math.nju.edu.cn/~jfyang/IADM_NNLS/index.html)

<sup>7</sup>Available at <http://sipi.usc.edu/database/>

表 8.3: Comparisons on the NMC problem (8.132) with synthetic data, averaged on 10 runs.  $q$ ,  $t$ , and  $d_r$  denote, respectively, the sample ratio, the number of measurements  $t = q(mn)$ , and the “degree of freedom” defined by  $d_r = r(m+n-r)$  for an  $m \times n$  matrix with rank  $r$  and  $q$ . Here we set  $m = n$  and fix  $r = 10$  in all the tests.

| <b>X</b> |     |         | LADM  |         |         |         | LADMPSAP |                |                |          |
|----------|-----|---------|-------|---------|---------|---------|----------|----------------|----------------|----------|
| $n$      | $q$ | $t/d_r$ | Iter. | Time(s) | RelErr  | FA      | Iter.    | Time(s)        | RelErr         | FA       |
| 1000     | 20% | 10.05   | 375   | 177.92  | 1.35E-5 | 6.21E-4 | 58       | <b>24.94</b>   | <b>9.67E-6</b> | <b>0</b> |
|          | 10% | 5.03    | 1000  | 459.70  | 4.60E-5 | 6.50E-4 | 109      | <b>42.68</b>   | <b>1.72E-5</b> | <b>0</b> |
| 5000     | 20% | 50.05   | 229   | 1613.68 | 1.08E-5 | 1.93E-4 | 49       | <b>369.96</b>  | <b>9.05E-6</b> | <b>0</b> |
|          | 10% | 25.03   | 539   | 2028.14 | 1.20E-5 | 7.70E-5 | 89       | <b>365.26</b>  | <b>9.76E-6</b> | <b>0</b> |
| 10000    | 10% | 50.03   | 463   | 6679.59 | 1.11E-5 | 4.18E-5 | 89       | <b>1584.39</b> | <b>1.03E-5</b> | <b>0</b> |

image (all pixels have been normalized in the range of  $[0, 1]$ ) by sampling 20% of the pixels uniformly at random and adding Gaussian noise with mean zero and standard deviation 0.1.

Besides LADM, here we also consider another recently proposed fixed point continuation with approximate SVD (FPCA [98]) on this problem. Similar to LADM, the code of FPCA<sup>8</sup> can only solve the standard matrix completion problem without the non-negativity constraint. This time we set  $\varepsilon_1 = 10^{-3}$  and  $\varepsilon_2 = 10^{-1}$  as the thresholds for stopping criteria. The recovered images are shown in Fig. 8.1 (c)-(e) and the quantitative results are in Table 8.4. One can see that on our test image both the qualitative and the quantitative results of LADMPSAP are better than those of FPCA and LADM. Note that LADMPSAP is faster than FPCA and LADM even though they do not handle the nonnegativity constraint.

### 8.12.3 Solving Group Sparse Logistic Regression with Overlap

In this subsection, we apply proximal LADMPSAP to solve the problem of group sparse logistic regression with overlap (8.134).

The Lipschitz constant of the gradient of logistic function with respect to  $\bar{\mathbf{w}}$  can be proven to be  $L_{\bar{\mathbf{w}}} \leq \frac{1}{4s} \|\bar{\mathbf{X}}\|_2^2$ , where  $\bar{\mathbf{X}} = (\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, \dots, \bar{\mathbf{x}}_s)$ . Thus (8.134) can be directly solved by Algorithm 20.

<sup>8</sup>Code available at <http://www1.se.cuhk.edu.hk/~sqma/softwares.html>

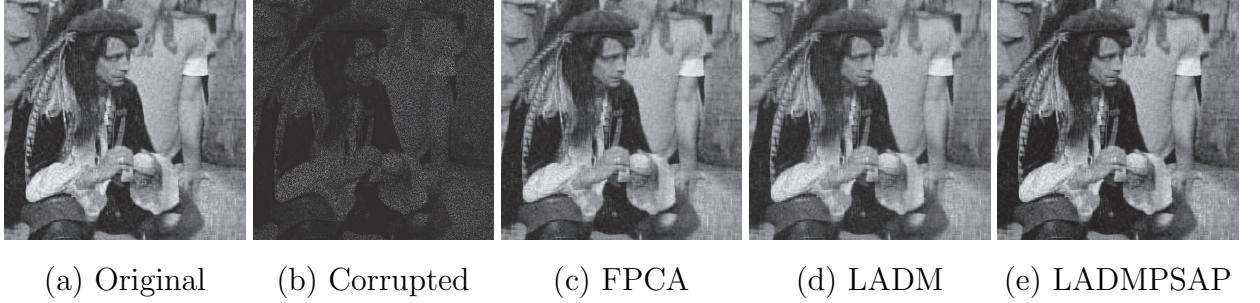


图 8.1: Image inpainting by FPCA, LADM and LADMPSP.

表 8.4: Comparisons on the image inpainting problem. “PSNR” stands for “Peak Signal to Noise Ratio” measured in decibel (dB).

| Method  | #Iter. | Time (s)      | PSNR (dB)    | FA       |
|---------|--------|---------------|--------------|----------|
| FPCA    | 179    | 228.99        | 27.77        | 9.41E-4  |
| LADM    | 228    | 207.95        | 26.98        | 2.92E-3  |
| LADMPSP | 143    | <b>134.89</b> | <b>31.39</b> | <b>0</b> |

### 8.12.3.1 Synthetic Data

To assess the performance of proximal LADMPSP, we simulate data with  $p = 9t+1$  variables, covered by  $t$  groups of ten variables with overlap of one variable between two successive groups:  $\{1, \dots, 10\}, \{10, \dots, 19\}, \dots, \{p-9, \dots, p\}$ . We randomly choose  $q$  groups to be the support of  $\mathbf{w}$ . If the chosen groups have overlapping variables with the unchosen groups, the overlapping variables are removed from the support of  $\mathbf{w}$ . So the support of  $\mathbf{w}$  may be less than  $10q$ .  $\mathbf{y} = (y_1, \dots, y_s)^T$  is chosen as  $(1, -1, 1, -1, \dots)^T$ .  $\mathbf{x} \in \mathbb{R}^{p \times s}$  is generated as follows. For  $\mathbf{x}_{i,j}$ , if  $i$  is in the support of  $\mathbf{w}$  and  $y_j = 1$ , then  $\mathbf{x}_{i,j}$  is generated uniformly on  $[0.5, 1.5]$ ; if  $i$  is in the support of  $\mathbf{w}$  and  $y_j = -1$ , then  $\mathbf{x}_{i,j}$  is generated uniformly on  $[-1.5, -0.5]$ ; if  $i$  is not in the support of  $\mathbf{w}$ , then  $\mathbf{x}_{i,j}$  is generated uniformly on  $[-0.5, 0.5]$ . Then the rows whose indices are in the support of  $\mathbf{w}$  are statistically different from the remaining rows in  $\mathbf{x}$ , hence can be considered as informative rows. We use model (8.135) to select the informative rows for classification, where  $\mu = 0.1$ . If the ground truth support of  $\mathbf{w}$  is recovered, then the two groups of data are linearly separable by considering only the coordinates in the support of  $\mathbf{w}$ .

We compare proximal LADMPSP with a series of ADM based methods, including ADM, LADM, LADMPS, and LADMPSP, where the subproblems for  $\mathbf{w}$  and  $\mathbf{b}$  have to be solved iteratively, e.g., by APG [8]. We terminate the inner loop by APG when the

表 8.5: Comparisons among ADM, LADM, LADMPS, LADMPSAP, and proximal LADMPSAP (pLADMPSAP) on the group sparse logistic regression with overlap problem. The quantities include the computing time (in seconds), number of outer iterations, and relative errors.

| $(s, p, t, q)$         | Method    | Time        | #Iter. | $\frac{\ \hat{\mathbf{w}} - \bar{\mathbf{w}}^*\ }{\ \bar{\mathbf{w}}^*\ }$ | $\frac{\ \hat{\mathbf{z}} - \mathbf{z}^*\ }{\ \mathbf{z}^*\ }$ |
|------------------------|-----------|-------------|--------|--|--|
| $(300, 901, 100, 10)$  | ADM       | 294.15      | 43     | 0.4800   | 0.4790   |
|                        | LADM      | 229.03      | 43     | 0.5331   | 0.5320   |
|                        | LADMPS    | 105.50      | 47     | 0.2088   | 0.2094   |
|                        | LADMPSAP  | 57.46       | 39     | 0.0371   | 0.0368   |
|                        | pLADMPSAP | <b>1.97</b> | 141    | <b>0.0112</b>  | <b>0.0112</b>  |
| $(450, 1351, 150, 15)$ | ADM       | 450.96      | 33     | 0.4337   | 0.4343   |
|                        | LADM      | 437.12      | 36     | 0.5126   | 0.5133   |
|                        | LADMPS    | 201.30      | 39     | 0.1938   | 0.1937   |
|                        | LADMPSAP  | 136.64      | 37     | 0.0321   | 0.0306   |
|                        | pLADMPSAP | <b>4.16</b> | 150    | <b>0.0131</b>  | <b>0.0131</b>  |
| $(600, 1801, 200, 20)$ | ADM       | 1617.09     | 62     | 1.4299   | 1.4365   |
|                        | LADM      | 1486.23     | 63     | 1.5200   | 1.5279   |
|                        | LADMPS    | 494.52      | 46     | 0.4915   | 0.4936   |
|                        | LADMPSAP  | 216.45      | 32     | 0.0787   | 0.0783   |
|                        | pLADMPSAP | <b>5.77</b> | 127    | <b>0.0276</b>  | <b>0.0277</b>  |

norm of gradient of the objective function of the subproblem is less than  $10^{-6}$ . As for the outer loop, we choose  $\varepsilon_1 = 2 \times 10^{-4}$  and  $\varepsilon_2 = 2 \times 10^{-3}$  as the thresholds to terminate the iterations.

For ADM, LADM, and LADMPS, which use a fixed penalty  $\beta$ , as we do not find any suggestion on its choice in the literature (the choice suggested in [144] is for nuclear norm regularized least square problem only) we try multiple choices of  $\beta$  and choose the one that results in the fastest convergence. For LADMPSAP, we set  $\beta_0 = 0.2$  and  $\rho_0 = 5$ . For proximal LADMPSAP we set  $T_1 = \frac{1}{4s}\|\bar{\mathbf{x}}\|_2^2$ ,  $\eta_1 = 2.01\|\bar{\mathbf{s}}\|_2^2$ ,  $T_2 = 0$ ,  $\eta_2 = 2.01$ ,  $\beta_0 = 1$ , and  $\rho_0 = 5$ . To measure the relative errors in the solutions we iterate proximal LADMPSAP for 2,000 times and regard its output as the ground truth solution  $(\bar{\mathbf{w}}^*, \mathbf{z}^*)$ .

Table 8.5 shows the comparison among related algorithms. The ground truth support of  $\mathbf{w}$  is recovered by all the compared algorithms. We can see that ADM, LADM, LADMPS, and LADMPSAP are much slower than proximal LADMPSAP because of the

time-consuming subproblem computation, although they have much smaller number of outer iterations. Their numerical accuracies are also inferior to that of proximal LADMP-SAP. We can also see that LADMP-SAP is faster and more numerically accurate than ADM, LADM, and LADMPS. This again testifies to the effectiveness of using adaptive penalty.

表 8.6: Comparisons among the Active Set method [64], LADM, LADMP-SAP, and proximal LADMP-SAP (pLADMP-SAP) on the pathway analysis. We present the CPU time (in seconds), classification error rate, and number of pathways. Results are estimated by three-fold cross validation. #Pathway gives the number of pathways that the selected genes belong to in each of the cross validation.

| Method     | Time       | Error             | #Pathway |
|------------|------------|-------------------|----------|
| Active Set | 2179       | $0.36 \pm 0.03$   | 6, 5, 78 |
| LADM       | 2433       | $0.315 \pm 0.049$ | 7, 9, 10 |
| LADMP-SAP  | 1593       | $0.329 \pm 0.011$ | 7, 9, 9  |
| pLADMP-SAP | <b>179</b> | $0.312 \pm 0.026$ | 4, 6, 6  |

### 8.12.3.2 Pathway Analysis on Breast Cancer Data

Then we consider the pathway analysis problem using the breast cancer gene expression data set [129], which consists of 8141 genes in 295 breast cancer tumors (78 metastatic and 217 non-metastatic). We follow (**author?**) [64] and use the canonical pathways from MSigDB [121] to generate the overlapping gene sets, which contains 639 groups of genes, 637 of which involve genes from our study. The statistics of the 637 gene groups are summarized as follows: the average number of genes in each group is 23.7, the largest gene group has 213 genes, and 3510 genes appear in these 637 groups with an average appearance frequency of about four. We follow (**author?**) [64] to restrict the analysis to the 3510 genes and balance the data set by using three replicates of each metastasis patient in the training set. We use model (8.135) to select genes, where  $\mu = 0.08$ . We want to predict whether a tumor is metastatic ( $y_i = 1$ ) or non-metastatic ( $y_i = -1$ ).

We compare proximal LADMP-SAP with the active set method, which was adopted in [64]<sup>9</sup>, LADM, and LADMP-SAP. In LADMP-SAP and proximal LADMP-SAP, we both set  $\beta_0 = 0.8$  and  $\rho_0 = 1.1$ . For LADM, we try multiple choices of  $\beta$  and choose the one that results in the fastest convergence. In LADM and LADMP-SAP, we terminate the

<sup>9</sup>Code available at <http://cbio.ensmp.fr/~ljacq/documents/overlasso-package.tgz>.

inner loop by APG when the norm of gradient of the objective function of the subproblem is less than  $10^{-6}$ . The thresholds for terminating the outer loop are all chosen as  $\varepsilon_1 = 10^{-3}$  and  $\varepsilon_2 = 6 \times 10^{-3}$ . For the three LADM based methods, we first solve (8.135) to select genes. Then we use the selected genes to re-train a traditional logistic regression model and use the model to predict the test samples. As in [64] we partition the whole data set into three subsets to do the experiment three times. Each time we select one subset as the test set and the other two as the training set (i.e., there are  $(78 + 217) \times 2/3 = 197$  samples for training). It is worth mentioning that (**author?**) [64] only kept the 300 genes that are the most correlated with the output in the pre-processing step. In contrast, we use all the 3510 genes in the training phase.

Table 8.6 shows that proximal LADMP SAP is more than ten times faster than the active set method used in [64], although it computes with a more than ten times larger training set. Proximal LADMP SAP is also much faster than LADM and LADMP SAP due to the lack of inner loop to solve subproblems. The prediction error and the sparseness at the pathway level by proximal LADMP SAP is also competitive with those of other methods in comparison.

**练习207.** Cancel  $q_k$  in Algorithm 12 and rewrite Algorithm 12.

**练习208.** Several algorithms can be applied to solve the fixed-rank problem:

$$\min_{\mathbf{X}} \|\mathcal{A}(\mathbf{X}) - \mathbf{b}\|^2, \quad s.t. \quad \text{rank}(\mathbf{X}) \leq r, \quad (8.164)$$

where  $\mathcal{A}$  is a linear mapping,  $\mathbf{b}$  is a known vector, and  $r$  is a known upper bound on the rank. Possible algorithms include:

1. Projected gradient [9].
2. Factorization method, where the problem is reformulated as:

$$\min_{\mathbf{P}, \mathbf{Q}} \|\mathcal{A}(\mathbf{P}\mathbf{Q}^T) - \mathbf{b}\|^2,$$

where both  $\mathbf{P}$  and  $\mathbf{Q}$  have  $r$  columns and they are minimized alternately.

3. APG with Nuclear Norm, where the problem is reformulated as:

$$\min_{\mathbf{X}} \|\mathcal{A}(\mathbf{X}) - \mathbf{b}\|^2 + \mu \|\mathbf{X}\|_*,$$

in which  $\mu > 0$  is appropriately chosen.

4. APG with truncated Nuclear Norm, where the problem is reformulated as:

$$\min_{\mathbf{X}} \|\mathcal{A}(\mathbf{X}) - \mathbf{b}\|^2 + \mu \|\mathbf{X}\|_{*,r},$$

in which  $\mu > 0$  is appropriately chosen and  $\|\mathbf{X}\|_{*,r} = \sum_{i>r} \sigma_i(\mathbf{X})$  is the  $r$ -th truncated nuclear norm. There are two choices of using this method. First, at the  $k$ -th iteration further approximate  $\|\mathbf{X}\|_{*,r}$  with  $\|\mathbf{X}\|_* - \text{tr}(\mathbf{U}_k^T \mathbf{X} \mathbf{V}_k)$ , where  $\mathbf{U}_k$  and  $\mathbf{V}^T$  consist of the first  $r$  left and right singular vectors of  $\mathbf{X}_k$ , respectively [63]. Second, no further approximation, where the proximal mapping of  $\|\mathbf{X}\|_{*,r}$  will be used (see Theorem 209).

5. Iteratively reweighted least squares (IRLS), where the problem is reformulated as:

$$\min_{\mathbf{X}} \|\mathcal{A}(\mathbf{X}) - \mathbf{b}\|^2 + \mu \|\mathbf{X}\|_{S_p}^p,$$

in which  $\mu > 0$  is appropriately chosen and  $p \in (0, 1)$ ,  $\|\mathbf{X}\|_{S_p} = [\text{tr}((\mathbf{X} \mathbf{X}^T)^{p/2})]^{1/p} = (\sum_i \sigma_i^p(\mathbf{X}))^{1/p}$  is called the Schatten  $p$ -norm of a matrix. To apply IRLS, at iteration  $k+1$ ,  $\|\mathbf{X}\|_{S_p}^p$  is approximated as:

$$\text{tr}((\mathbf{X} \mathbf{X}^T)(\mathbf{X}_k \mathbf{X}_k^T)^{p/2-1}).$$

6. ADM/LADM. For LADM, the problem is reformulated as:

$$\min_{\mathbf{X}, \mathbf{e}} \|\mathbf{e}\|^2, \quad \text{s.t.} \quad \mathcal{A}(\mathbf{X}) - \mathbf{e} = \mathbf{b}, \text{rank}(\mathbf{X}) \leq r.$$

For ADM, one more auxiliary variable is needed. ADM/LADM operates on the partial augmented Lagrangian function:

$$L(\mathbf{X}, \mathbf{e}, \mathbf{y}) = \|\mathbf{e}\|^2 + \langle \mathbf{y}, \mathcal{A}(\mathbf{X}) - \mathbf{e} - \mathbf{b} \rangle + \frac{\beta}{2} \|\mathcal{A}(\mathbf{X}) - \mathbf{e} - \mathbf{b}\|^2.$$

Only when updating  $\mathbf{X}$ , the constraint  $\text{rank}(\mathbf{X}) \leq r$  is enforced.

Apply the above algorithms to the image inpainting problem, i.e.,  $\mathbf{X}$  is a  $512 \times 512$  image and  $\mathcal{A}(\mathbf{X})$  is to uniformly randomly sample 20% pixel values and  $r$  is 40 (c.f. Figure 8.1). Show their results and report their PSNRs. The following theorem may be useful.

**定理209.** Let  $\mathbf{W}, \mathbf{Y} \in \mathbb{R}^{m \times n}$ . The solution  $\mathbf{W}^*$  to

$$\min_{\mathbf{W}} \varepsilon \|\mathbf{W}\|_{*,r} + \frac{1}{2} \|\mathbf{W} - \mathbf{Y}\|_F^2 \tag{8.165}$$

is  $\mathcal{D}_{r,\varepsilon}(\mathbf{Y}) = \mathbf{U} \tilde{\Sigma} \mathbf{V}^T$ , where  $\varepsilon > 0$  and  $\tilde{\sigma}_i = \sigma_i$ ,  $i = 1, \dots, r$ , and  $\tilde{\sigma}_i = \max(\sigma_i - \varepsilon, 0)$ ,  $i > r$ , in which  $\mathbf{U} \Sigma \mathbf{V}^T$  is the SVD of  $\mathbf{Y}$ .



## 第九章 二阶稀疏表示

(The materials are based on an extended version of [5]. An earlier version is [1].)

### 9.1 引言

前面讲的稀疏表示都是针对一阶信号的，如语音、一般的特征向量等。但在实际应用中，我们将面临着各种各样的数据，如图像、视频和基因微阵列（Microarray），它们天然就是矩阵甚至是张量。于是我们就自然面对着一个问题：如何度量矩阵和张量的稀疏性？

基于低秩的模型是近年涌现出来的鲁棒高效地处理高维数据的新工具。虽然秩在统计学中早已被用作矩阵的正则化子，如减秩回归（Reduced Rank Regression, RRR）；在三维立体视觉里，秩约束更是随处可见，但是近年低秩模型的兴起却是受取得了巨大成功的稀疏表示（Sparse Representation）和压缩传感（Compressed Sensing）理论的推动，由此系统地发展出了新的理论与应用。在此背景下，秩被阐释为二阶（即矩阵）稀疏性<sup>1</sup>的度量，而不仅仅是一个数学概念。为了说明这一点，我们举图像或视频压缩为例，要实现有效地压缩，必须充分利用图像或视频的时间或空间相关性；又如Netflix挑战<sup>2</sup>（图9.1），要推断未知的用户评价，需要充分考虑用户喜好的相关性和视频类别的相关性。矩阵行列间的相关性天然地和矩阵的秩关联在一起。因此，把秩定义为二阶稀疏性度量是很自然的。

下面以我们的工作为基础，简要介绍这方面的进展。我们先介绍线性模型，再介绍非线性模型，然后是常用求解算法，接下来是代表性应用，最后总结；其中线性模型分单子空间模型和多子空间模型，优化算法分凸优化和非凸优化。

### 9.2 线性模型

近年低秩模型的兴起大致开始于E. Candès 2008年提出的矩阵填充（Matrix Completion, MC）问题[24]。我们先介绍线性模型。虽然看上去比较简单，但是理论分析表明线性模型对强噪声和缺失数据非常鲁棒，在应用中其实也具有足够的数据表达能力。

<sup>1</sup>一阶稀疏性就是向量的稀疏性，其度量为非零元的个数，即 $\ell_0$ 范数 $\|\cdot\|_0$ 。

<sup>2</sup>Netflix是一家视频租赁公司，拥有很多用户对视频的评价，但这个用户/视频评价矩阵非常稀疏。该公司提供100万美元奖金希望能够把预测用户对视频的评价的准确率提高10%，以便有针对性地推荐，从而提高营收。见<http://www.netflixprize.com/>

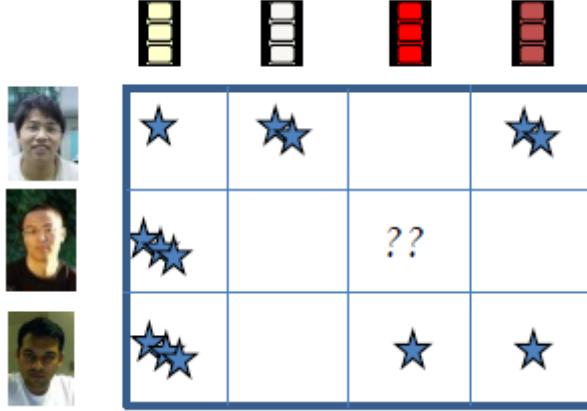


图 9.1: Netflix 挑战。需要预测用户对其未评价过的视频的喜好程度。

### 9.2.1 单子空间模型

E. Candès 2008年提出的MC问题[24]是：已知某矩阵 $\mathbf{D}$ 在某些位置的值，可否恢复出该矩阵？这是一个广泛的数学模型，如上面提到的Netflix挑战、基因微阵列测量等。显然这个问题的答案是不确定的，鉴于上面提到的需要考虑矩阵行列间的相关性，他建议选秩最小的那个解 $\mathbf{A}$ ：

$$\min_{\mathbf{A}} \text{rank}(\mathbf{A}), \quad s.t. \quad \pi_{\Omega}(\mathbf{A}) = \pi_{\Omega}(\mathbf{D}), \quad (9.1)$$

其中 $\Omega$ 是已知值的矩阵元素的位置的集合， $\pi_{\Omega}$ 是保留位置在 $\Omega$ 里的矩阵元素的值、其他位置填0的投影算子。他考虑的是数据缺失时如何恢复低秩结构的问题。稍后，E. Candès又进一步考虑了带噪声的MC问题[23]：

$$\min_{\mathbf{A}} \text{rank}(\mathbf{A}), \quad s.t. \quad \|\pi_{\Omega}(\mathbf{A}) - \pi_{\Omega}(\mathbf{D})\|_F^2 \leq \varepsilon, \quad (9.2)$$

以处理测量数据有噪声的情况。

如果考虑数据有强噪声时如何恢复低秩结构的问题，看似这个问题可以用传统的PCA解决，但实际上传统PCA只在噪声是高斯噪声时可以准确恢复潜在的低秩结构。对于非高斯噪声，如果噪声很强，即使是极少数的噪声，也会使传统的主元分析失败。由于主元分析在应用上的极端重要性，大量学者付出了很多努力在提高主元分析的鲁棒性上，提出了许多号称“鲁棒”的主元分析方法，但是没有一个方法被理论上严格证明是能够在一定条件下一定能够精确恢复出低秩结构的。2009年，Chandrasekaran等人[128]和Wright等人[139]同时提出了鲁棒主元分析（Robust PCA, RPCA）。他们考虑的是数据中有稀疏大噪声时如何恢复数据的低秩结构：

$$\min_{\mathbf{A}, \mathbf{E}} \text{rank}(\mathbf{A}) + \lambda \|\mathbf{E}\|_0, \quad s.t. \quad \mathbf{A} + \mathbf{E} = \mathbf{D}, \quad (9.3)$$

其中 $\|\mathbf{E}\|_0$ 表示 $\mathbf{E}$  中非零元的个数。J. Wright的工作后来得到E. Candès的加入，获得了更强的结果，即观测矩阵可以只在部分位置知道值。推广后的模型为[22]：

$$\min_{\mathbf{A}, \mathbf{E}} \text{rank}(\mathbf{A}) + \lambda \|\mathbf{E}\|_0, \quad s.t. \quad \pi_\Omega(\mathbf{A} + \mathbf{E}) = \pi_\Omega(\mathbf{D}). \quad (9.4)$$

在他们的论文里，也讨论了带稠密高斯噪声的广义RPCA模型[22]：

$$\min_{\mathbf{A}, \mathbf{E}} \text{rank}(\mathbf{A}) + \lambda \|\mathbf{E}\|_0, \quad s.t. \quad \|\pi_\Omega(\mathbf{A} + \mathbf{E}) - \pi_\Omega(\mathbf{D})\|_F^2 \leq \varepsilon. \quad (9.5)$$

Chen等人考虑了噪声集中在若干列的情况，提出了Outlier Pursuit模型[30]，它把RPCA模型中的 $\|\mathbf{E}\|_0$ 换成 $\|\mathbf{E}\|_{2,0}$ ，即计算 $\mathbf{E}$ 的列向量的 $\ell_2$ 范数有多少个为0。

Liu等人把矩阵填充推广到了张量填充[88]。虽然张量有基于CP分解定义的秩，但是它不可计算，所以Liu等人提出了一种折衷的张量秩，定义为张量按不同模式(mode) 展开(unfolding) 后得到的不同矩阵的秩的和。他们提出的张量填充模型就是：在已知张量某些位置的值时，通过极小化折衷的张量秩来恢复缺失的值。同样基于折衷的张量秩，Tan等人把RPCA推广到了张量恢复[122]，即把给定张量分解为两个张量之和，一个具有较低的折衷的张量秩，另一个稀疏。

### 9.2.2 多子空间模型

RPCA只能从数据中提取一个子空间，它对数据在此子空间中的精细结构无法刻画。精细结构的最简单情形是多子空间模型，即数据分布在若干子空间附近，我们需要找到这些子空间。这个问题马毅等人称为Generalized PCA (GPCA)问题[132]，之前已有很多算法，如代数法、RANSAC等，但都没有理论保障。稀疏表示的出现为这个问题提供了新的思路。E. Elhamifar和R. Vidal 2009年利用样本间相互表达，在表达系数矩阵稀疏的目标下提出了Sparse Subspace Clustering (SSC)模型[45] ((9.6)中 $\text{rank}(\mathbf{Z})$ 换成 $\|\mathbf{Z}\|_0$ ,  $\|\mathbf{E}\|_{2,0}$ 换成 $\|\mathbf{E}\|_0$ ，同时添加约束 $\text{diag}(\mathbf{Z}) = 0$ 以防止只用样本本身表达自己)。受此启发，刘光灿等人提出了Low-Rank Representation (LRR)模型[82, 86]：

$$\min_{\mathbf{Z}, \mathbf{E}} \text{rank}(\mathbf{Z}) + \lambda \|\mathbf{E}\|_{2,0}, \quad s.t. \quad \mathbf{D} = \mathbf{DZ} + \mathbf{E}. \quad (9.6)$$

之所以要求表达系数矩阵 $\mathbf{Z}$ 低秩，是为了增强 $\mathbf{Z}$ 各列之间的相关性以提高对噪声的抵抗能力。SSC 和LRR的最优表达系数矩阵 $\mathbf{Z}^*$ 可以作为样本间的相似性度量，用 $(|\mathbf{Z}^*| + |\mathbf{Z}^{*,T}|)/2$ <sup>3</sup>定义样本间的权重（ $|\mathbf{Z}^*|$ 表示把 $\mathbf{Z}^*$ 的元素都取绝对值得到的矩阵），再通过谱聚类就可以把数据聚类成若干线性子空间。由于最优表达系数矩阵 $\mathbf{Z}^*$  可以作为样本间的相似性度量，庄连生等人进一步要求系数矩阵稀疏、非负，以应用于半监督学习[167]。

<sup>3</sup>在后来的TPAMI2013论文[86]中，刘光灿等人改用 $|\mathbf{U}_{Z^*} \mathbf{U}_{Z^*}^T|$ 构建邻接矩阵，其中 $\mathbf{U}_{Z^*}$ 是 $Z^*$ 的瘦型奇异值分解的左奇异向量集合。原因见第9.2.3.2节。

LRR需要假定数据充足。在样本不足情形，刘光灿和颜水成[85]提出了Latent LRR模型：

$$\min_{\mathbf{Z}, \mathbf{L}, \mathbf{E}} \text{rank}(\mathbf{Z}) + \text{rank}(\mathbf{L}) + \lambda \|\mathbf{E}\|_0, \quad s.t. \quad \mathbf{D} = \mathbf{DZ} + \mathbf{LD} + \mathbf{E}. \quad (9.7)$$

他们称 $\mathbf{DZ}$ 为Principal Feature， $\mathbf{LD}$ 为Salient Feature， $\mathbf{Z}$ 用于子空间聚类， $\mathbf{L}$ 则可用于提取数据的鉴别信息以识别。而刘日升等人[91]提出了Fixed Rank Representation (FRR) 模型：

$$\min_{\mathbf{Z}, \tilde{\mathbf{Z}}, \mathbf{E}} \|\mathbf{Z} - \tilde{\mathbf{Z}}\|_F^2 + \lambda \|\mathbf{E}\|_{2,0}, \quad s.t. \quad \mathbf{D} = \mathbf{DZ} + \mathbf{E}, \text{rank}(\tilde{\mathbf{Z}}) \leq r, \quad (9.8)$$

其中 $\tilde{\mathbf{Z}}$ 用于度量样本间的相似性。

为了进一步提高子空间聚类的精度，卢参义等人提出使用Trace Lasso来约束表达系数[93]：

$$\min_{\mathbf{Z}, \mathbf{E}} \|\mathbf{D}\text{diag}(\mathbf{Z}_i)\|_* + \lambda \|\mathbf{E}_i\|_0, \quad s.t. \quad \mathbf{D}_i = \mathbf{DZ}_i + \mathbf{E}_i, i = 1, \dots, n, \quad (9.9)$$

其中 $\mathbf{Z}_i$ 是矩阵 $\mathbf{Z}$ 的第*i*列， $\|\mathbf{D}\text{diag}(\mathbf{Z}_i)\|_*$ 称为向量 $\mathbf{Z}_i$ 的Trace Lasso， $\|\cdot\|_*$ 为矩阵的核范数 (Nuclear Norm，即矩阵奇异值之和)。当 $\mathbf{D}$ 的列是 $\ell_2$ -范数归一化的时，Trace Lasso具有优美的插值性质：

$$\|\mathbf{Z}_i\|_2 \leq \|\mathbf{D}\text{diag}(\mathbf{Z}_i)\|_* \leq \|\mathbf{Z}_i\|_1,$$

而且左端等式在数据是完全相关时（每列相同或方向相反）达到，右端在数据是完全无关时（列之间正交）达到。因此，Trace Lasso具有自适应于数据相关性的特性。该模型称为Correlation Adaptive Subspace Segmentation (CASS)。

为了更有效地对张量数据进行聚类，傅逸凡等人提出了Tensor LRR模型[50]，以便综合张量在各模式下的信息。

### 9.2.3 理论分析

对低秩模型的理论分析是比较丰富的，可以归纳为以下三个部分。

#### 9.2.3.1 精确恢复

上面介绍的低秩模型都是离散优化问题，很多都是NP-难，这对求解它们造成了很大的困难。为了克服这个困难，一个常用的办法是把它们近似成凸优化问题。粗略地说，和 $\ell_0$ 范数 $\|\cdot\|_0$ “最接近”的凸函数是 $\ell_1$  范数 $\|\cdot\|_1$ ，即所有元素绝对值的和；和秩“最接近”的凸函数是核范数 $\|\cdot\|_*$ 。这样，上面的离散优化问题都可以转化为凸优化问题，可以比较方便地求解。但是会自然产生一个问题：求解凸优化问题是否能够得到真实的解？大部分针对单子空间的低秩模型，如MC[24]、RPCA[22]、带缺失数据

的RPCA[22]、Outlier Pursuit[30, 150]，都有相应的肯定回答，基本上都可以归纳为：如果Outlier是稀疏的且随机分布的、真实解矩阵是低秩的，则真实解矩阵是可以精确恢复的。需要指出的神奇之处是：精确恢复性与Outlier的大小无关，而是与其稀疏程度有关。这就保证了大部分针对单子空间的低秩模型是具有很强的鲁棒性的。这个特性和传统的PCA的差别是非常显著的。而对于针对多子空间的低秩模型，只有LRR有较为详细的理论分析[83]，但刘光灿等人仅证明了当Outlier比例不超过某阈值时， $\mathbf{Z}_0$ 的行空间和哪些样本是Outlier是可以精确恢复的， $\mathbf{Z}_0$ 的行空间可以从 $\mathbf{U}_{Z^*}\mathbf{U}_{Z^*}^T$ 得到，其中 $\mathbf{U}_{Z^*}\Sigma_{Z^*}\mathbf{V}_{Z^*}^T$ 为最优解 $\mathbf{Z}^*$ 的瘦型奇异值分解。该分析没有回答 $\mathbf{Z}_0$ 和 $\mathbf{E}_0$ 本身是否可以精确恢复，但是幸运的是，将LRR应用于子空间聚类时，我们只需要 $\mathbf{Z}_0$ 的行空间就够了。

### 9.2.3.2 闭解

低秩模型一个非常令人振奋的性质在于它在无噪声情形可能有闭解，这是稀疏模型所不具备的性质。魏嗣明和林宙辰[136]分析了LRR的数学性质，首先发现无噪声的LRR模型：

$$\min_{\mathbf{Z}} \|\mathbf{Z}\|_*, \quad s.t. \quad \mathbf{D} = \mathbf{D}\mathbf{Z}, \quad (9.10)$$

有唯一解，而且该解可以很方便地表达出来：设 $\mathbf{D}$ 的瘦型奇异值分解为 $\mathbf{U}_D\Sigma_D\mathbf{V}_D^T$ ，则该解为 $\mathbf{V}_D\mathbf{V}_D^T$ 。矩阵 $\mathbf{V}_D\mathbf{V}_D^T$ 在立体视觉里被称为Shape Interaction Matrix。刘光灿等人[86]进一步发现使用一般字典的LRR模型：

$$\min_{\mathbf{Z}} \|\mathbf{Z}\|_*, \quad s.t. \quad \mathbf{D} = \mathbf{B}\mathbf{Z}, \quad (9.11)$$

也有唯一解，而且该解也可以很方便地表达出来： $\mathbf{Z}^* = \mathbf{B}^+\mathbf{D}$ ，其中 $\mathbf{B}^+$ 为 $\mathbf{B}$ 的Moore-Penrose 伪逆。这个结论被Yu和Schuurmans推广到一般的正交不变范数[148]，在该文中作者找到了更多的具有闭解的低秩问题。Favaro等人也找到了若干和子空间聚类有关的带闭解的低秩问题[46]。张宏扬等人进一步发现无噪声的Latent LRR问题（离散形式和凸形式）的解不唯一，并给出了所有闭解[149]，文中还发现离散形式的无噪声LRR问题（即(9.6)中 $\mathbf{E}$ 为0）事实上不是NP-难的，并给出了所有闭解。张宏扬等人因此提出从Latent LRR解集中寻找最稀疏解的算法来弥补Latent LRR的这个缺陷[151]。

### 9.2.3.3 块对角结构

面向多子空间的低秩模型都会算得一个表达系数矩阵 $\mathbf{Z}$ 。对于SRC和LRR，都可以证明：在理想情况下，即当样本无噪声、子空间相互独立（即任何子空间都不能用其他子空间来表达）时，最优的表达系数矩阵 $\mathbf{Z}^*$ 是块对角的。由于每个对角块对应于一个子空间， $\mathbf{Z}^*$ 的块对角结构对子空间聚类至关重要。令人惊讶的是，卢参义等人证明了：如果 $\mathbf{Z}$ 使用Frobenius范数的平方来正则化（相应模型称为LSR），则在

理想情况下最优表达系数矩阵 $\mathbf{Z}^*$ 也是块对角的[94]。为此，卢参义等人提出了强化块对角条件（Enforced Block-Diagonal Conditions），只要对 $\mathbf{Z}$ 的正则化项满足强化块对角条件，则在理想情况下最优解就是块对角的[94]，这就大大拓宽了 $\mathbf{Z}$ 的正则化项的选择范围，即可以不限是稀疏或低秩约束。对于 $\mathbf{Z}$ 按列求解的子空间聚类模型，如基于Trace Lasso的CASS 模型(9.9)，卢参义等人也相应提出了强化块稀疏条件（Enforced Block-Sparse Conditions），只要对 $\mathbf{Z}$ 的列的正则化项满足强化块稀疏条件，则在理想情况下最优的就是块对角的[93]。但是以上结论都是在理想情况下分析的，在有噪声或子空间之间不独立情形，最优的 $\mathbf{Z}$ 会偏离块对角，对后续的寻找子空间造成一定的困难。为此，冯佳时等人根据谱图理论关于Laplacian矩阵0特征值重数和权重矩阵对角块个数之间的对应关系提出了块对角先验[48]，把它添加到前面的子空间聚类模型中，就可以保证在任何情况下得到的都是严格块对角的表达系数矩阵 $\mathbf{Z}$ ，从而显著提高了对噪声的抵抗能力。

表达系数之间的群组效应（Grouping Effect），即当样本相近时其表达系数向量也相近，对在噪声情况下保持表达系数矩阵 $\mathbf{Z}$ 的块对角结构有一定作用。SSC、LRR、LSR、CASS等都证明了具有群组效应。胡瀚等人为此提出了一般性的强化群组效应条件（Enforced Grouping Effect Conditions）[62]，可以很方便地判定正则化函数具有群组效应。

### 9.3 非线性模型

用于分割非线性流形的低秩模型比较少。一个自然的想法是利用Kernel技巧，由Wang等人提出[134]。其思想是：假设通过非线性映射 $\phi$ ，样本集 $\mathbf{X}$ 在高维空间中分布在线性子空间上，则可以对映射后的样本集运用LRR模型。假定噪声是高斯的，则模型为：

$$\min_{\mathbf{Z}} \|\phi(\mathbf{X}) - \phi(\mathbf{X})\mathbf{Z}\|_F^2 + \lambda \|\mathbf{Z}\|_*.$$

由于 $\|\phi(\mathbf{X}) - \phi(\mathbf{X})\mathbf{Z}\|_F^2 = \text{tr} [(\phi(\mathbf{X}) - \phi(\mathbf{X})\mathbf{Z})^T(\phi(\mathbf{X}) - \phi(\mathbf{X})\mathbf{Z})]$ ，就得到了 $\phi(\mathbf{X})^T\phi(\mathbf{X})$ ，就可以引入核函数 $K(\mathbf{x}, \mathbf{y})$ ，使得 $K(\mathbf{x}, \mathbf{y}) = \phi^T(\mathbf{x})\phi(\mathbf{y})$ 。因此，上面的模型可以写成核化的形式而不需显示引入非线性映射 $\phi$ 。但是，当噪声不是高斯时，上述核技巧不适用。

### 9.4 优化算法

第9.2节中所列离散低秩模型一般都是NP-难的，只能近似求解。一般的做法是转化成连续优化问题。这时有两种转化方式，一种是转化为凸优化问题，如上面提到的，把 $\ell_0$ 范数 $\|\cdot\|_0$ 换成 $\ell_1$ 范数 $\|\cdot\|_1$ 、秩换成核范数 $\|\cdot\|_*$ 。另一种是转化为转化为非凸优化问题，就是用非凸的连续函数来近似 $\ell_0$ 范数 $\|\cdot\|_0$ （比如用 $\ell_p$  范数 $\|\cdot\|_p$  ( $0 < p < 1$ )）和秩（比如用Schatten- $p$ 范数（奇异值的 $\ell_p$ 范数））。还有一种做法是把需要约束为低秩的矩阵

直接表达成两个矩阵的乘积，第一个矩阵的列数和第二个矩阵的行数均为期望的秩，然后交替更新第一个和第二个矩阵直至不变为止。这种做法是在稀疏模型中所没有的。凸优化的好处是能够得到修改后的模型的全局最优解，但是解有可能不够低秩或稀疏；非凸优化的好处是往往能得到更低秩和更稀疏的解，但是不能得到修改后的模型的全局最优解，解的质量可能依赖于初值。所以凸优化和非凸优化互为补充。针对问题的特点，还有可能设计随机算法，大大降低求解的复杂度。

#### 9.4.0.4 凸优化算法

凸优化已经是比较成熟的领域，一般都有多项式时间复杂度的求解算法（比如内点法）。对于大规模/高维数据，一般需要 $O(n \text{polylog}(n))$ 的复杂度，连平方复杂度都是不可容忍的。为了帮助理解低秩模型所面临的计算复杂度的障碍，我们举以下例子说明。对于RPCA问题，如果矩阵大小是 $n \times n$ ，则该问题有 $2n^2$ 个未知数，即使 $n = 1000$ ，它对应于一个不大的矩阵，未知数个数也达到了一百万。如果使用内点法来求解，比如使用斯坦福大学的CVX软件包，则每迭代一次的时间复杂度是 $O(n^6)$ ，而空间复杂度则是 $O(n^4)$ ，导致一般内存为4GB的PC机只能处理 $80 \times 80$ 的矩阵。因此要使得低秩模型实用，必须要设计高效的优化算法。

目前面向大规模计算的优化算法都是一阶算法。代表性的算法包括加速近邻梯度法（Accelerated Proximal Gradient, APG）[8, 107]和交错方向法（Alternating Direction Method, ADM）[78, 80, 81]。APG主要针对无约束问题。在目标函数为 $C^{1,1}$ ，即可微且梯度Lipschitz连续时，一般的梯度下降法的收敛速度只能达到 $O(k^{-1})$ ，其中 $k$ 为迭代次数，但是Nesterov构造了一个算法[107]，其收敛速度可以达到 $O(k^{-2})$ 。后来Beck和Teboulle把Nesterov的算法推广到目标函数为一个凸函数和一个 $C^{1,1}$ 函数和的情形[8]，大大扩展了Nesterov算法的应用范围。另外，APG需要估计Lipschitz系数，如果Lipschitz系数估计得过于保守（太大），则会影响收敛速度，所以Beck和Teboulle还进一步提出了使用动态估计的Lipschitz系数，以加速收敛[8]。在一些有特殊结构的问题上，APG可以被推广（Generalized APG, GAPG）[168]，使得不同变量对应的Lipschitz系数不同，从而加快收敛。对于带线性约束的优化问题，可以把约束的平方放到目标函数里作为惩罚项，再应用APG求近似解。为加快收敛速度，惩罚系数要逐渐增加到较大的值，这一重要技巧称为Continuation[51]。ADM面向目标函数可分解（即为两个不同变量凸函数的和）、带线性约束和凸集约束的问题，它是Lagrange乘子法的一种变形。它先构造问题的增广Lagrangian函数，然后通过交替极小化增广Lagrangian函数来更新两个变量，最后更新Lagrange乘子，如此迭代[78]。其好处是更新变量子问题要比原问题简单，甚至有闭解。在子问题不容易求解的情形，可以考虑线性化增广Lagrangian函数的增广项，把问题进一步简化，这一技术称为线性化交错方向法（Linearized Alternating Direction Method, LADM）[78]。如果线性化增广Lagrangian函数的增广项还不足以导致足够简单的子问题，还可以进一步把目标函数

的光滑成分也线性化[81]。对于多变量（变量数大于2）的凸规划问题，简单地套用两个变量的ADM法不能保证收敛。但是如果把串行更新改成并行更新，通过适当的参数选取，则可以保证收敛，带线性化的情形也是如此[81]。在以上介绍的ADM系列算法中，增广项的惩罚参数可以动态增长，以加速收敛[78, 80, 81]。

无论是用何种凸优化算法求解低秩模型，都会碰到如下形式的子问题：

$$\min_{\mathbf{X}} \|\mathbf{X}\|_* + \frac{\alpha}{2} \|\mathbf{X} - \mathbf{W}\|_F^2.$$

它有闭解[20]：设 $\mathbf{W}$ 的奇异值分解为 $\mathbf{W} = \mathbf{U}\Sigma\mathbf{V}^T$ ，则最优解为 $\mathbf{X} = \mathbf{U}\Theta_{\alpha^{-1}}(\Sigma)\mathbf{V}^T$ ，其中

$$\Theta_\varepsilon(x) = \begin{cases} x - \varepsilon, & \text{如果 } x > \varepsilon, \\ x + \varepsilon, & \text{如果 } x < -\varepsilon, \\ 0 & \text{如果 } -\varepsilon \leq x \leq \varepsilon. \end{cases} \quad (9.12)$$

因此求解低秩模型往往离不开奇异值分解，而维度为 $m \times n$ 的矩阵的奇异值分解的复杂度为 $O(mn \min(m, n))$ ，所以一般来说求解低秩模型计算量都很大，在面临大矩阵时问题尤为突出。所幸从(9.12)可以看出小于 $\alpha^{-1}$ 的奇异值及其奇异向量无需求出，因为这些奇异值将被收缩为0，从而对 $\mathbf{X}$ 没有贡献。于是就可以只计算大于 $\alpha^{-1}$ 的奇异值及其奇异向量，这可以使用PROPACK来实现，计算量相应地下降到 $O(rmn)$ ，其中 $r$ 为最优 $\mathbf{Z}$ 的秩。值得一提的是，PROPACK只能提供指定个数的最大奇异值及其奇异向量，所以调用PROPACK时要动态地预测 $r$ 的值[78]。当解不是低秩的时，比如在图像处理和计算机视觉里已经有广泛应用的Transform Invariant Low-Rank Textures (TILT) [157] (见(9.13)及第9.5.4节)，可以使用增量奇异值分解[115]进行加速。

#### 9.4.1 非凸优化算法

对于用Schatten- $p$ 范数近似秩、 $\ell_p$ 范数近似 $\ell_0$ 范数的无约束问题，一个有效做法是迭代重加权最小二乘法 (Iteratively Reweighted Least Squares) [95]，即把 $\text{tr}((\mathbf{X}\mathbf{X}^T)^{p/2})$ 近似为 $\text{tr}((\mathbf{X}_k\mathbf{X}_k^T)^{(p/2)-1}(\mathbf{X}\mathbf{X}^T))$ 、 $|\mathbf{x}_i|^p$ 近似为 $|\mathbf{x}_i^{(k)}|^{p-2}\mathbf{x}_i^2$ ，其中 $\mathbf{X}_k$ 为低秩矩阵 $\mathbf{X}$ 在第 $k$ 次迭代的值， $\mathbf{x}_i^{(k)}$ 为稀疏向量 $\mathbf{x}$ 的第 $i$ 个分量在第 $k$ 次迭代的值。这样每次更新 $\mathbf{X}$ 需要解矩阵方程，更新 $\mathbf{x}$ 需要求解线性方程组。另一种方法是借用APG的思想，把目标函数的 $C^{1,1}$ 成分也线性化，这样每次只需求解如下形式的子问题：

$$\min_{\mathbf{X}} \sum_{i=1}^n g(\sigma_i(\mathbf{X})) + \frac{\alpha}{2} \|\mathbf{X} - \mathbf{W}\|_F^2,$$

其中 $g$ 是单调递增的凹函数，比如 $x^p$ ( $0 < p < 1$ )。卢参义等人给出了求解上面子问题的算法[96]。另一种近似秩的函数是截断核范数 (Truncated Nuclear Norm, TNN) [63]: $\|\mathbf{X}\|_r = \sum_{i=r+1}^{\min(m,n)} \sigma_i(\mathbf{X})$ ，它不包含前 $r$ 个奇异值，因此不是凸函数。可以很容易

看出来，极小化TNN会促使后面的奇异值尽可能小，而前 $r$ 个奇异值取值多少没有影响，因此可以促使解更接近一个秩 $r$ 的矩阵。TNN可以更进一步，给越往后的奇异值加越大的权重，得到加权核范数（Weighted Nuclear Norm, WNN）[57]：

$$\|\mathbf{X}\|_{w,*} = \sum_{i=1}^{\min(m,n)} w_i \sigma_i(\mathbf{X})。$$

但此时求解子问题：

$$\min_{\mathbf{X}} \sum_{i=1}^n \|\mathbf{X}\|_{w,*} + \frac{\alpha}{2} \|\mathbf{X} - \mathbf{W}\|_F^2,$$

一般没有闭解，需要解一个小型的优化问题。

第三种方法是直接把待求的低秩矩阵 $\mathbf{X}$ 表达成 $\mathbf{X} = \mathbf{AB}^T$ ，其中 $\mathbf{A}$ 和 $\mathbf{B}$ 都是列数为 $r$ 的矩阵，则可以交替更新 $\mathbf{A}$ 和 $\mathbf{B}$ 直至不变[137]。这种方法的优势在于简单，缺点在于需要较好地先验估计最优解的秩。

#### 9.4.2 随机优化算法

以上方法，无论凸或是非凸，计算复杂度都至少是 $O(rmn)$ ，其中 $m, n$ 是待求低秩矩阵的尺寸，这在 $m, n$ 都很大时还是不够快。要突破这个复杂度瓶颈，需要使用随机算法。但是随机算法不能简单地把确定型算法的每一步随机化，因为有些步骤的随机化会导致很大的误差。所以随机算法需要根据低秩模型的特点来设计，导致目前这方面工作还不多。对RPCA，刘日升等人提出了 $\ell_1$ -滤波方法[90]，先随机采样数据矩阵 $\mathbf{D}$ 的适当大小的子矩阵 $\mathbf{D}^s$ ，在 $\mathbf{D}^s$ 上求解小型RPCA问题得到低秩的 $\mathbf{A}^s$ 和稀疏的 $\mathbf{E}^s$ ，然后 $\mathbf{D}^s$ 所在的其他行和列用求得的 $\mathbf{A}^s$ 来处理，最后原数据矩阵 $\mathbf{D}$ 对应的低秩矩阵 $\mathbf{A}$ 可以通过Nystrom技巧来表出，整个算法的复杂度为 $O(r^3) + O(r^2(m+n))$ ，为关于矩阵大小线性。对LRR和Latent LRR，张宏扬等人发现如果模型略作改动，其解和RPCA的解可以互为表出，因此，都可以通过先求解RPCA来大大加速[152]。

### 9.5 代表性应用

低秩模型在信号处理和机器学习等领域里已经获得了广泛的应用，NIPS2011上曾出现了大量的讨论低秩模型的论文。由于专业所限，本节只简要介绍笔者及其合作者在图像处理和计算机视觉领域里找到的典型应用。

#### 9.5.1 视频去噪[67]

由于同一视频中各帧之间非常相似，同一帧中的不同图像区域之间也有很大的相似程度，我们很自然地可以假定由这些图像块排列而成的矩阵是低秩的，而根据某一像素值是否背离同一位置处所有像素的“均值”判定该点是否可靠，进而用矩阵填充模型(9.2)来恢复那些被噪声污染的像素。部分结果如图9.2所示。

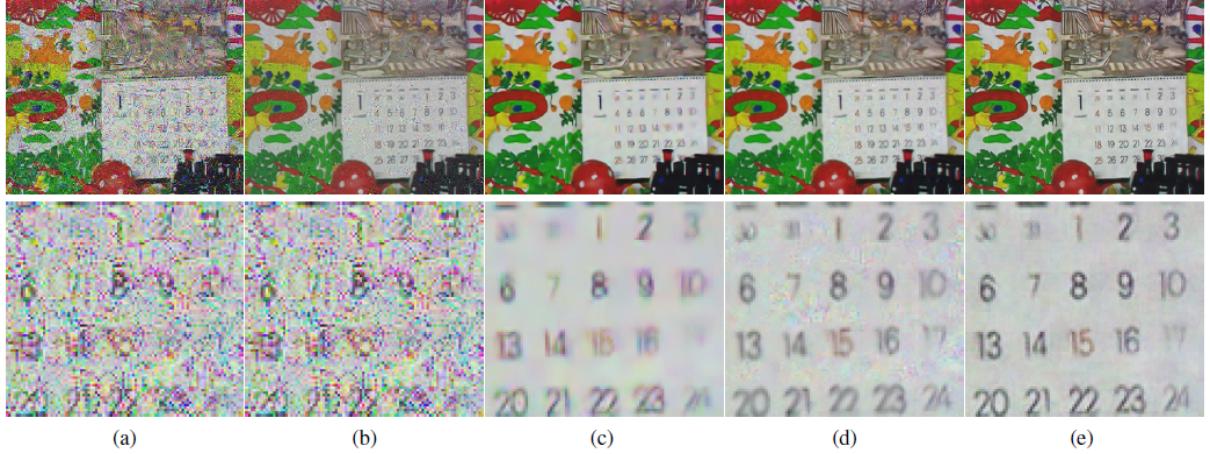


图 9.2: 视频去噪结果。(a) VBM3D的结果, 无冲击噪声预处理。(b) PCA的结果, 无冲击噪声预处理。(c) VBM3D的结果, 做了冲击噪声预处理。(d) PCA的结果, 做了冲击噪声预处理。(e) Matrix Completion的结果。

### 9.5.2 背景建模[22]

背景建模的最简单情形是从固定摄相机拍摄的视频中分离背景和前景。此时很容易想到背景是基本不变的, 所以如果把背景的每一帧作为矩阵的一列, 则该矩阵低秩。同时由于前景是移动的物体, 占据像素比例较低, 所以前景对应于视频中的稀疏“噪声”部分。由此得到做背景建模的RPCA模型(9.3), 其中 $\mathbf{D}$ 的每一列是视频的每一帧拉直后得到的向量,  $\mathbf{A}$ 的每一列对应于背景的每一帧拉直后得到的向量,  $\mathbf{E}$ 的每一列对应于前景的每一帧拉直后得到的向量。部分结果如图9.3所示。

### 9.5.3 图像批量对齐 (RASL) [113]

背景建模需要假定背景已经对齐, 这样才能得到低秩的背景视频。在没有对齐的情况下, 可以考虑把每一帧/每一幅图像作适当的几何变形使它们对齐。此时数学模型为:

$$\min_{\tau, \mathbf{A}, \mathbf{E}} \|\mathbf{A}\|_* + \lambda \|\mathbf{E}\|_1, \quad \mathbf{D} \circ \tau = \mathbf{A} + \mathbf{E}, \quad (9.13)$$

其中 $\mathbf{D} \circ \tau$ 是把每一帧/每一幅图像作适当的几何变形 $\tau$ 的形式写法 (每一帧/每一幅图像的几何变形不同)。此时, (9.13)是一个非凸优化问题。为了有效求解, Peng等人[113]提出了把 $\tau$ 局部线性化的迭代算法, 即做如下循环直至 $\Delta\tau_k$ 足够小:

$$\begin{cases} \min_{\Delta\tau_k, \mathbf{A}, \mathbf{E}} \|\mathbf{A}\|_* + \lambda \|\mathbf{E}\|_1, & \mathbf{D} \circ \tau_k + \mathbf{J} \Delta\tau_k = \mathbf{A} + \mathbf{E}, \\ \tau_{k+1} \leftarrow \tau_k + \Delta\tau_k, \\ k \leftarrow k + 1, \end{cases} \quad (9.14)$$

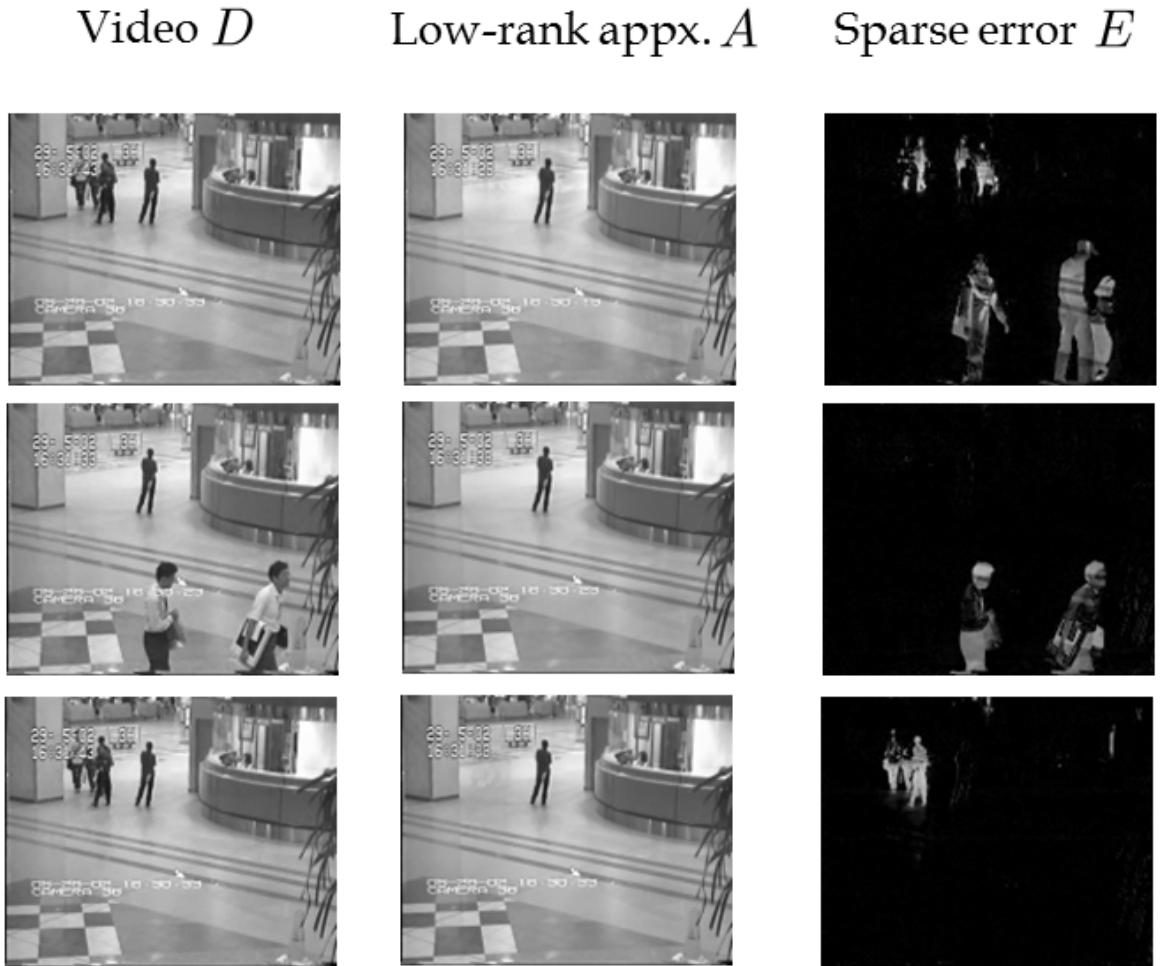


图 9.3: 背景建模。第一列是监控视频, 第二列是背景视频, 第三列是前景视频(值取绝对值)。

其中  $\mathbf{J}$  是  $\mathbf{D} \circ \tau$  对  $\tau$  的参数的 Jacobi 矩阵。在仿射变换模型下, 人脸图像的部分对齐结果如图9.4所示。

#### 9.5.4 变换不变低秩纹理 (TILT) [157]

变换不变低秩纹理 (Transform Invariant Low-rank Textures, TILT) 的数学模型形式上和RASL的(9.13)完全一样, 其求解过程也完全一样, 但是这里  $\mathbf{D}$  是图像的一个长方形图像块。TILT的思想是通过几何变换  $\tau$  把  $\mathbf{D}$  所代表的图像区域校正成正则的区域, 如具有横平竖直、对称等特性, 这些特性可以通过低秩性来进行刻画。图9.5是射影变换下图像校正的例子。

TILT原则上对任何参数变换都适用。Zhang等人[158]还考虑了基于广义柱体变换

的TILT，可用于人造建筑物表面纹理的提取。一些例子如图9.6所示。

TILT还被广泛应用于建筑物几何建摸[157]、相机自动标定和镜头畸变自动校正[159]。由于其应用上的重要性，Ren 和Lin[115]特别研究了它的快速算法，把求解速度提高了5倍以上。

### 9.5.5 运动分割[82, 87]

LRR(9.6)被认为是做刚体运动分割最好的算法之一。所谓刚体运动分割，就是把视频里做刚体运动的物体上的特征点进行聚类，使得每一类对应于一个独立运动的物体，这样就可以得到物体运动的轨迹。部分例子如图9.7所示。

### 9.5.6 图像分割[31]

图像分割是特殊的聚类问题。首先把图像过分割（Over Segment）成超像素（Super-pixel），然后在超像素上提取适当的特征，通过改进的LRR模型综合多种特征（基本上每一种特征对应于一个LRR模型），求出整体表示矩阵 $\mathbf{Z}^*$ ，然后对用 $(|\mathbf{Z}^*| + |\mathbf{Z}^{*T}|)/2$ 表出的相似性矩阵进行正则化割（Normalized Cut），得出超像素的聚类关系，每一类就对应于一个图像区域。部分例子见图9.8.

### 9.5.7 图像显著区域检测[74]

运动分割和图像分割都是利用了LRR的表示矩阵 $\mathbf{Z}$ ，而图像显著区域检测则是利用LRR里的稀疏“噪声” $\mathbf{E}$ 。图像显著区域一般就是图像中“与众不同”的区域，因此如果用其他区域进行“预测”则会产生较大的误差。因此，如果把图像分解成小块，在其上提取适当特征，则图像显著区域对应于LRR里的稀疏“噪声” $\mathbf{E}$ 较大的部分。部分例子如图9.9所示。

### 9.5.8 其他应用

低秩模型的其他应用，如部分重复图像检索[146]、人脸识别[114]、结构化纹理修复、街景全景图拼接、人造物体定向（Upright Orientation）、基于光度学的立体视觉[140]、图像标签的改进[166]、视觉域适应（Visual Domain Adaption） [66]、鲁棒视觉跟踪[154]、三维人脸特征提取[103]、CT重建[52]、图像半监督分类[167]、图像集的协同分割（co-segmentation）、视点无关步态识别、文档关键词提取、甚至音频分析[111] [104]、蛋白质-基因相关性分析、网络流量异常检测、鲁棒滤波与系统辨识等等，限于篇幅就不再一一介绍了。

## 9.6 结束语

低秩模型近年来信号处理、机器学习、计算机视觉等领域获得了广泛的应用，短短几年从理论、算法到应用各方面都得到了快速的发展，本文只是基于我们的工作做了非常粗浅的介绍。许多具体问题，如果结合问题的特性适当地引入低秩性约束，很多情况下都能得到更好的结果。有些问题的数据本身可能没有低秩性，此时可以引入适当变换增强其低秩性（如RASL/TILT对RPCA的改进）。有些学者没有检查数据是否具有低秩性或对数据进行适当的预处理就声称使用低秩约束效果不好，这是不太严谨的。从上面的介绍可以看出，低秩模型在如下几个方面还需要更多的研究：矩阵往张量的推广、非线性流形聚类、近线性复杂度的随机算法等。希望本节的介绍能起到抛砖引玉的作用。



人脸的初始姿态



人脸的中间结果



人脸的最终结果

图 9.4: 人脸对齐的迭代过程。



图 9.5: 利用TILT模型进行图像校正的例子。第一行表示原始的图像块(长方形框)及相应校正变换(四边形框, 校正变换就是四边形框相对于长方形框的几何变换), 第二行是校正后的图像块。

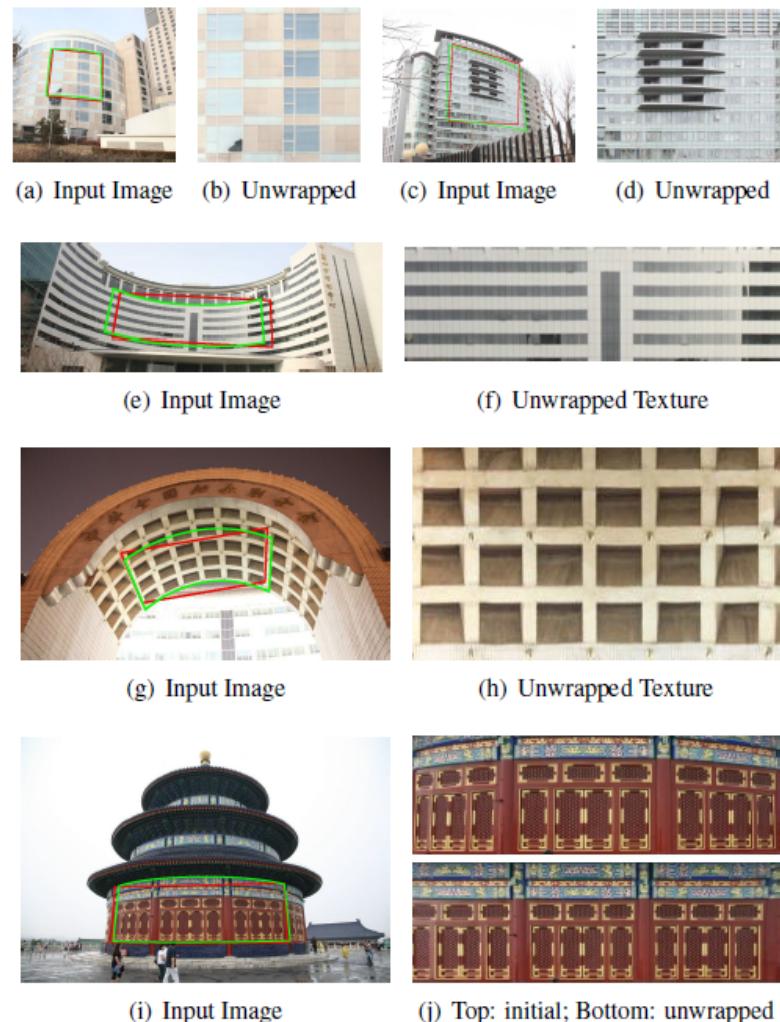


图 9.6: 基于广义柱体变换的TILT的建筑物纹理展开。



图 9.7: 运动分割的例子。

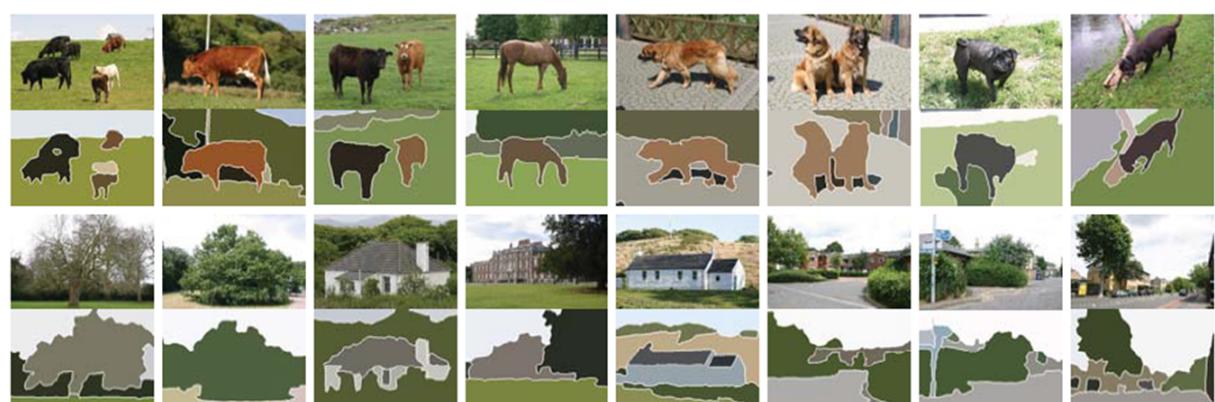


图 9.8: 图像分割的例子。

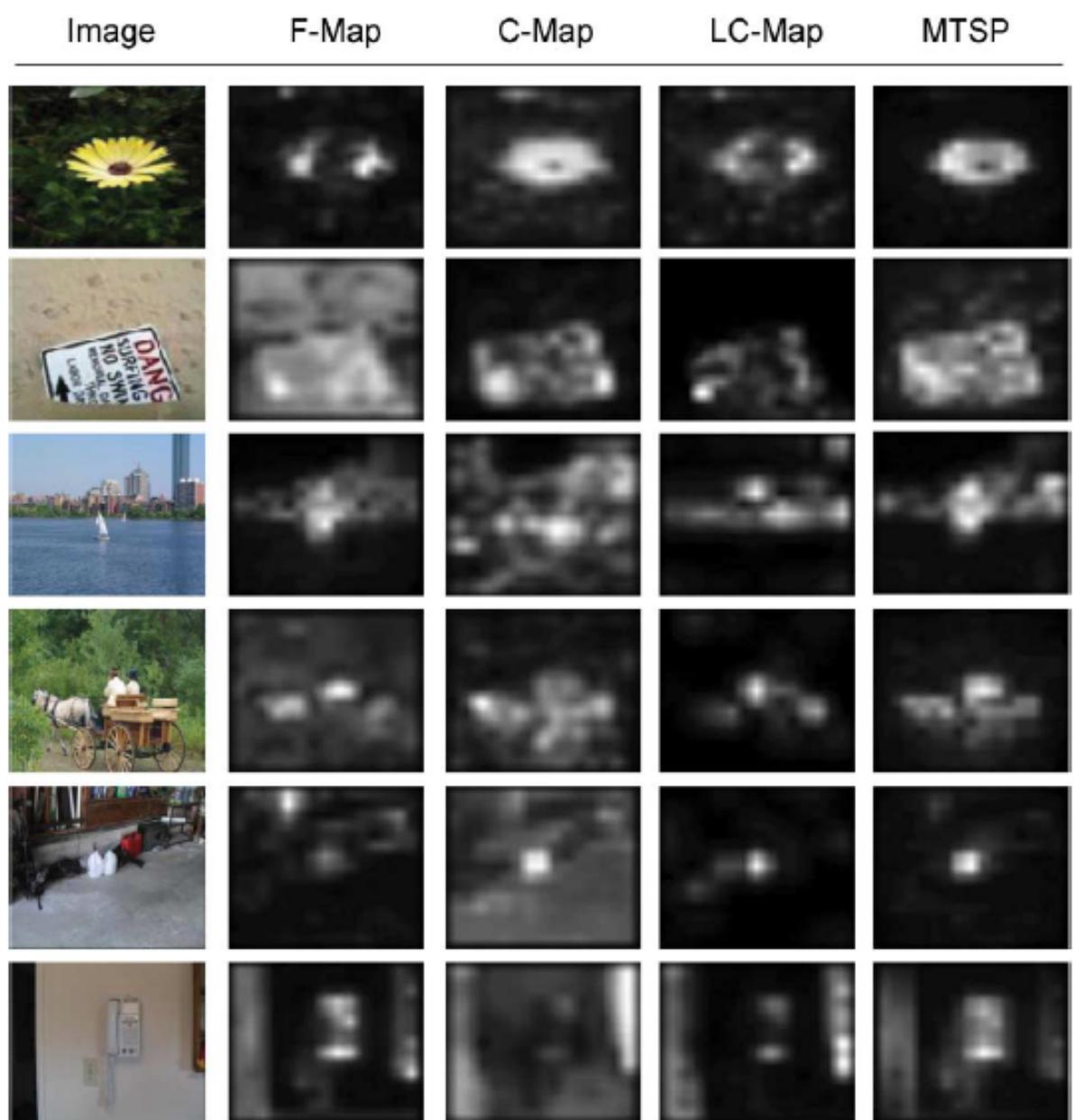


图 9.9: 图像显著区域检测的例子。第一列是输入图像，第二至五列是不同方法的检测结果，其中最后一列是基于LRR的方法。

## 第十章 Fast Algorithms for DR Approximation

(Chapter 15 of [133])

In nonlinear dimensionality reduction, the kernel dimension is the square of the vector number in the data set. In many applications, the number of data vectors is very large. The spectral decomposition of a large dimensional kernel encounters difficulties in at least three aspects: large memory usage, high computational complexity, and computational instability. Although the kernels in some nonlinear DR methods are sparse matrices, which enable us to overcome the difficulties in memory usage and computational complexity partially, yet it is not clear if the instability issue can be settled. In this chapter, we study some fast algorithms that avoid the spectral decomposition of large dimensional kernels in DR processing, dramatically reducing memory usage and computational complexity, as well as increasing numerical stability.

### 10.1 Low-rank Approximation and Rank-revealing Factorization

In the PCA algorithm, the key step is to make the SVD decomposition of the data matrix. In nonlinear DR algorithms, the DR date is derived from the spectral decomposition of DR kernels. When the dimension of a DR kernel is very large, the spectral decomposition of the kernel is a challenging problem. Therefore, an efficient low-rank approximation of the decomposition is a ubiquitous task. In this section we introduce the concept of rank-revealing factorization, which directly leads to low-rank approximation of matrices. In the context, we often use the term *matrix decomposition* for *matrix factorization*.

#### 10.1.1 Rank-revealing Factorization

As an example of matrix decomposition, we review the SVD factorization. Assume that  $\mathbf{A} \in \mathcal{M}_{m,n}(k)$  is an  $m \times n$  matrix with rank  $k$ . The SVD factorization of  $\mathbf{A}$  is

$$\mathbf{A} = \sum_{j=1}^k \sigma_j \mathbf{u}_j \mathbf{v}_j^T = \mathbf{U} \Sigma \mathbf{V}^T, \quad (10.1)$$

where  $\mathcal{U} = \{\mathbf{u}_1, \dots, \mathbf{u}_k\}$  is an o.n. basis of the  $k$ -dimensional subspace spanned by the columns of  $\mathbf{A}$ , and  $\mathcal{V} = \{\mathbf{v}_1, \dots, \mathbf{v}_k\}$  is the o.n. basis of the functionals on  $\text{Col}(\mathbf{A})$ . Let

$d \leq k$ . Write

$$\mathbf{A}_d = \sum_{j=1}^d \sigma_j \mathbf{u}_j \mathbf{v}_j^T = \mathbf{U}_d \boldsymbol{\Sigma}_d \mathbf{V}_d^T. \quad (10.2)$$

Then

$$\|\mathbf{A} - \mathbf{A}_d\|_2 = \sigma_{d+1}, \quad \|\mathbf{A} - \mathbf{A}_d\|_F^2 = \sum_{j=d+1}^k \sigma_j^2.$$

Hence,  $\mathbf{A}_d$  is the best approximation of  $\mathbf{A}$  among all  $d$ -rank  $m \times n$  matrices under the spectral norm (and the Frobenius norm as well). We call  $\mathbf{A}_d$  the best  $d$  low-rank approximation of  $\mathbf{A}$ , or the best  $d$ -rank-revealing approximation of  $\mathbf{A}$ . Since SVD is computationally expensive when  $m$  and  $n$  are large, people try to find some approximative algorithms that are fast but do not lose too much accuracy. This idea leads to:

**定义210.** Let  $\mathbf{A} \in \mathcal{M}_{m,n}(k)$  have singular values  $\sigma_1 \geq \dots \geq \sigma_k > 0$  and  $d \leq k$ . A matrix  $\tilde{\mathbf{A}}_d \in \mathcal{M}_{m,n}(d)$  is called a  $d$ -rank approximation of  $\mathbf{A}$  if

$$\|\mathbf{A} - \tilde{\mathbf{A}}_d\|_2 \leq c\sigma_{d+1}, \quad (10.3)$$

where  $c \geq 1$  is a constant independent of  $\mathbf{A}$ .

When  $c = 1$ ,  $\tilde{\mathbf{A}}_d$  is uniquely given by  $\mathbf{A}_d$  in (10.2). The norm in (10.3) can also be replaced by the Frobenius norm and  $\sigma_{d+1}$  is replaced by  $\sqrt{\sum_{j=d+1}^k \sigma_j^2}$ . When  $c > 1$ , the approximation matrix  $\tilde{\mathbf{A}}_d$  is not unique. The algorithm that realizes the low-rank approximation is called a low-rank approximation algorithm. For example, we have

**定义211.** An algorithm is called a  $d$ -rank approximation SVD algorithm, if for each  $\mathbf{A} \in \mathcal{M}_{m,n}(k)$ , the algorithm produces three matrices  $\mathbf{U} \in \mathcal{O}_{m,d}$ ,  $\boldsymbol{\Sigma} \in \mathcal{D}_d$ , and  $\mathbf{V} \in \mathcal{O}_{n,d}$ , such that

$$\|\mathbf{A} - \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T\|_2 \leq c\sigma_{d+1}, \quad (10.4)$$

where  $c \geq 1$  is a constant independent of  $\mathbf{A}$ .

Note that if the rank of  $\mathbf{A}$  is  $d$ , then  $d$ -rank approximation SVD algorithms produce the exact SVD decomposition of  $\mathbf{A}$ . When  $c > 1$  and  $\sigma_{d+1} \neq 0$ , the algorithms do not provide the best approximation as given in (10.2). However, as a consequence of relaxing the constraint  $c = 1$ , we may develop algorithms that reduce computational complexity. In approximative algorithms, we often need to make trade-offs between computational complexity and precision. The use of approximate matrices appears promising, as the degradation in performance is often marginal compared to the gain in processing speed.

Hence, low-rank approximation algorithms are very attractive in the applications, which are involved in large dimensional matrices.

The rank-revealing factorization provides an approach to low-rank approximation algorithms, which are directly derived from a truncated rank-revealing factorization. Because of the close relation between them, we do not distinguish *rank-revealing approximation* from *low-rank approximation* in terminology.

In general, a rank-revealing factorization is defined as follows.

**定义212.** For  $\mathbf{A} \in \mathcal{M}_{m,n}(k)$ , the factorization

$$\mathbf{A} = \mathbf{X}\mathbf{D}\mathbf{Y}^T \quad (10.5)$$

is called a rank-revealing factorization of  $\mathbf{A}$  if  $\mathbf{X} \in \mathcal{M}_{m,k}$ ,  $\mathbf{D} \in \mathcal{D}_k$ ,  $\mathbf{Y} \in \mathcal{M}_{n,k}$ , where  $\mathbf{D}$  is a diagonal matrix with nonnegative, non-increasing diagonal entries, and  $\mathbf{X}$  and  $\mathbf{Y}$  are well conditioned.

Let  $d \ll \min(m, n)$ ,  $\mathbf{X}_d$  be the submatrix of  $\mathbf{X}$  consisting of the first  $d$  columns of  $\mathbf{X}$ ,  $\mathbf{Y}_d$  be the submatrix of  $\mathbf{Y}$  consisting of the first  $d$  columns of  $\mathbf{Y}$ , and  $\mathbf{D}_d$  be the  $d \times d$  leading submatrix of  $\mathbf{D}$ . Then the truncated factorization

$$\tilde{\mathbf{A}}_d = \mathbf{X}_d \mathbf{D}_d \mathbf{Y}_d^T$$

is a  $d$ -rank approximation of  $\mathbf{A}$  provided  $\mathbf{X}$  and  $\mathbf{Y}$  both are “nearly” orthogonal. Here we use the word “nearly” orthogonal for a matrix with the conditional number near 1. Let  $\mathbf{A} \in \mathcal{M}_{m,n}$  have all singular values  $\sigma_1 \geq \dots \geq \sigma_k > 0$ . When  $\mathbf{X}$  and  $\mathbf{Y}$  are nearly o.g., the diagonal entries of  $\mathbf{D}$  are close to  $\sigma_j$ ,  $1 \leq j \leq k$ . Therefore,  $\tilde{\mathbf{A}}_d$  approximates  $\mathbf{A}_d$  in (10.2).

There are various rank-revealing factorizations. For example,

- $\mathbf{A} = \mathbf{Q}\mathbf{R}\mathbf{P}^T$  is called a rank-revealing QR factorization at index  $k$  if  $\mathbf{Q} \in \mathcal{O}_m$ ,  $\mathbf{R} = \begin{pmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \mathbf{0} & \mathbf{R}_{22} \end{pmatrix}$  is upper triangular with  $\mathbf{R}_{11} \in \mathcal{M}_{k,k}$ ,  $\mathbf{P} \in \mathcal{M}_n$  is a permutation matrix,  $\|\mathbf{R}_{22}\|_2 = O(\sigma_{k+1})$ , and  $\sigma_k(\mathbf{R}_{11}) = O(\sigma_k)$ .
- $\mathbf{A} = \mathbf{Q}^T \mathbf{L} \mathbf{U} \mathbf{P}^T$  is called a rank-revealing LU factorization at index  $k$  if  $\mathbf{P}$  and  $\mathbf{Q}$  are permutation matrices,  $\mathbf{L} = \begin{pmatrix} \mathbf{L}_{11} & \mathbf{0} \\ \mathbf{L}_{21} & \mathbf{L}_{22} \end{pmatrix}$  is lower triangular,  $\mathbf{U} = \begin{pmatrix} \mathbf{U}_{11} & \mathbf{U}_{12} \\ \mathbf{0} & \mathbf{U}_{22} \end{pmatrix}$  is upper triangular with  $\mathbf{U}_{11}, \mathbf{L}_{11} \in \mathcal{M}_k$ ,  $\|\mathbf{L}_{22}\mathbf{U}_{22}\|_2 = O(\sigma_{k+1})$ , and  $\sigma_k(\mathbf{L}_{11}\mathbf{U}_{11}) = O(\sigma_k)$ .

A rank-revealing factorization at index  $k$  is also called  $k$ -rank revealing factorization. An algorithm realizing a rank-revealing factorization is called rank-revealing algorithm.

### 10.1.2 Fast Rank-revealing Algorithms

The role of the permutation matrices in the factorizations above is to search the principal  $k$ -subspace of  $\text{Col}(\mathbf{A})$ . Since the columns of the best  $k$ -rank approximation of  $\mathbf{A}$  spans the principal  $k$ -subspace, a general  $k$ -rank approximation of  $\mathbf{A}$  ought to span a subspace close to the principal one. A rank-revealing algorithm is fast if it can quickly find the approximatively principal  $k$ -subspace. The existence of fast rank-revealing algorithms is confirmed by many theorems. The following is one of them.

**定理213.** Suppose  $\mathbf{A} \in \mathcal{M}_{m,n}$  and  $k \leq \min(m, n)$ . There is a matrix  $\mathbf{P} \in \mathcal{M}_{k,n}$  and a matrix  $\mathbf{B} \in \mathcal{M}_{m,k}$  whose columns constitute a subset of the columns of  $\mathbf{A}$  such that

1.  $\mathbf{P}$  has a  $k \times k$  submatrix, which is identity,
2. no entry of  $\mathbf{P}$  has an absolute value greater than 1,
3.  $\|\mathbf{P}\|_2 \leq \sqrt{k(n-k)+1}$ , and the least singular value of  $\mathbf{P}$  is  $\geq 1$ ,
4.  $\mathbf{BP} = \mathbf{A}$ , when  $k = m$  or  $k = n$ , and  $\|\mathbf{BP} - \mathbf{A}\| \leq \sqrt{k(n-k)+1}\sigma_{k+1}$ , when  $k < \min(m, n)$ , where  $\sigma_{k+1}$  is the  $(k+1)$ -th greatest singular value of  $\mathbf{A}$ .

It is also confirmed that reaching a nearly principal  $k$  column of  $\mathbf{A}$  only needs  $Ckmn \log(n)$  operations.

Rank-revealing algorithms, such as rank-revealing QR algorithms, rank-revealing LU algorithms, and rank-revealing Cholesky decomposition algorithms, are well studied in the name of low-rank approximation in the numerical matrix theory. There are books on rank-revealing decompositions. Many deterministic low-rank approximation algorithms already exist in literature. Fierro and Hansen developed two Matlab toolboxes, UTV Tools and UTV Expansion Pack, consisting of the rank-revealing algorithms, which are based on UTV factorization. Rank-revealing UTV factorization is a modification of rank revealing SVD factorization. It can also be considered as a generalization of Schur decomposition. In UTV factorization the middle diagonal matrix  $\Sigma$  in SVD decomposition (10.1) is relaxed to a block-triangular matrix  $\mathbf{T}$ . If the middle matrix is upper triangular, then the decomposition is called URT factorization. Let the middle matrix in a URT decomposition be denoted by  $\mathbf{R}$ . For  $m \leq n$ , the factorization takes the form at index  $d$  as follows.

$$\mathbf{A} = \mathbf{U}_R \begin{pmatrix} \mathbf{R} \\ \mathbf{0} \end{pmatrix} \mathbf{V}_R^T = ((\mathbf{U}_R)_d, (\mathbf{U}_R)_0, (\mathbf{U}_R)_{\perp}) \begin{pmatrix} \mathbf{R}_d & \mathbf{F} \\ \mathbf{0} & \mathbf{G} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} (\mathbf{V}_R)_d \\ (\mathbf{V}_R)_0 \end{pmatrix} \quad (10.6)$$

where  $\mathbf{R}_d \in \mathcal{M}_d$  is non-singular,  $\mathbf{G}$  is an  $(n-d) \times (n-d)$  matrix. The URV decomposition is said to be  $d$ -rank revealing if

$$\sigma_d(\mathbf{R}_d) = O(\sigma_d), \quad \|(\mathbf{F}^T, \mathbf{G}^T)\|_2 = O(\sigma_{d+1}). \quad (10.7)$$

The truncated  $d$ -rank-revealing URV factorization

$$(\mathbf{A}_R)_d = (\mathbf{U}_R)_d \mathbf{R}_d (\mathbf{V}_R)_d^T \quad (10.8)$$

is a  $d$ -rank-revealing approximation of  $\mathbf{A}$ . Let the SVD of  $\mathbf{R}_d$  be

$$\mathbf{R}_d = \mathbf{U}_d \mathbf{D}_d \mathbf{V}_d^T.$$

Then

$$(\mathbf{A}_R)_d = [(\mathbf{U}_R)_d \mathbf{U}_d] \mathbf{D}_d [(\mathbf{V}_R)_d \mathbf{V}_d]^T \triangleq (\mathbf{U}_A)_d \mathbf{D}_d (\mathbf{V}_A)_d$$

is a  $d$ -rank-revealing SVD approximation of  $\mathbf{A}$ . Since both  $\mathbf{U}_d$  and  $\mathbf{V}_d$  are o.g.,  $(\mathbf{U}_R)_d$  is a rotation of  $(\mathbf{U}_A)_d$  and  $(\mathbf{V}_R)_d$  is a rotation of  $(\mathbf{V}_A)_d$ . If the middle upper triangular matrix  $\mathbf{R}$  in the URV decomposition is replaced by a lower triangular matrix  $\mathbf{L}$ , then the decomposition is called a ULV one. For  $m \geq n$ , at index  $d$ , the ULV decomposition takes the form

$$\mathbf{A} = \mathbf{U}_L \begin{pmatrix} \mathbf{L} \\ \mathbf{0} \end{pmatrix} \mathbf{V}_L^T = ((\mathbf{U}_L)_d, (\mathbf{U}_L)_0, (\mathbf{U}_L)_{\perp}) \begin{pmatrix} \mathbf{L}_d & \mathbf{0} \\ \mathbf{H} & \mathbf{E} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} (\mathbf{V}_L)_d \\ (\mathbf{V}_L)_0 \end{pmatrix}, \quad (10.9)$$

where  $\mathbf{L}_d \in \mathcal{M}_d$  is non-singular,  $\mathbf{E}$  is an  $(n-d) \times (n-d)$  matrix. The ULV decomposition is said to be  $d$ -rank-revealing if

$$\sigma_d(\mathbf{L}_d) = O(\sigma_d), \quad \|(\mathbf{H}, \mathbf{E})\|_2 = O(\sigma_{d+1}). \quad (10.10)$$

The truncated  $d$ -rank-revealing ULV factorization

$$(\mathbf{A}_L)_d = (\mathbf{U}_L)_d \mathbf{L}_d (\mathbf{V}_L)_d^T \quad (10.11)$$

is a  $d$ -rank-revealing approximation of  $\mathbf{A}$ . Let the SVD of  $\mathbf{L}_d$  be

$$\mathbf{L}_d = \tilde{\mathbf{U}}_d \tilde{\mathbf{D}}_d \tilde{\mathbf{V}}_d^T.$$

Then

$$(\mathbf{A}_L)_d = [(\mathbf{U}_L)_d \tilde{\mathbf{U}}_d] \tilde{\mathbf{D}}_d [(\mathbf{V}_L)_d \tilde{\mathbf{V}}_d]^T \triangleq (\tilde{\mathbf{U}}_A)_d \tilde{\mathbf{D}}_d (\tilde{\mathbf{V}}_A)_d$$

is a  $d$ -rank-revealing SVD approximation of  $\mathbf{A}$ . When the gap between  $\sigma_d$  and  $\sigma_{d+1}$  is large, i.e.,  $\sigma_d \gg \sigma_{d+1}$ , the  $d$ -rank-revealing SVD approximation in (10.2) is very close to

the  $d$ -rank-revealing URV approximation in (10.8) and the  $d$ -rank-revealing ULV approximation in (10.11). To estimate the approximation errors, we need the definition of the distance between two subspaces with the same dimension. Assume that  $S_1$  and  $S_2$  are two subspaces of  $\mathbb{R}^m$  with the same dimension. Let  $\Theta(S_1, S_2)$  denote the angle between them. Then the distance between  $S_1$  and  $S_2$  can be defined by

$$d(S_1, S_2) = \sin(\Theta(S_1, S_2)).$$

For a matrix  $\mathbf{M}$ , we denote by  $\sigma_d(\mathbf{M})$  the  $d$ th singular value of  $\mathbf{M}$ .

**定理214.** *Let the  $d$ -rank-revealing UTV decomposition of  $\mathbf{A}$  be given in (10.8) and (10.11), and the SVD decomposition of  $\mathbf{A}$  be given in (10.1). Assume  $\sigma_d(\mathbf{R}_d) > \|\mathbf{G}\|_2$ . Then*

$$d(Col(\mathbf{U}_d), Col((\mathbf{U}_R)_d)) \leq \frac{\|\mathbf{F}\|_2 \|\mathbf{G}\|_2}{\sigma_d^2(\mathbf{R}_d) - \|\mathbf{G}\|_2^2}$$

and

$$\frac{\|\mathbf{F}\|_2}{2\|\mathbf{R}\|_2} d(Col(\mathbf{V}_d), Col((\mathbf{V}_R)_d)) \leq \frac{\sigma_d(\mathbf{R}_d) \|\mathbf{F}\|_2}{\sigma_d^2(\mathbf{R}_d) - \|\mathbf{G}\|_2^2}.$$

Similarly, assume  $\sigma_d(\mathbf{L}_d) > \|\mathbf{E}\|_2$ . Then

$$d(Col(\mathbf{V}_d), Col((\mathbf{V}_L)_d)) \leq \frac{\|\mathbf{H}\|_2 \|\mathbf{E}\|_2}{\sigma_d^2(\mathbf{L}_d) - \|\mathbf{E}\|_2^2}$$

and

$$\frac{\|\mathbf{H}\|_2}{2\|\mathbf{L}\|_2} d(Col(\mathbf{U}_d), Col((\mathbf{U}_L)_d)) \leq \frac{\sigma_d(\mathbf{L}_d) \|\mathbf{H}\|_2}{\sigma_d^2(\mathbf{L}_d) - \|\mathbf{E}\|_2^2}.$$

We skip the proof of Theorem 214.

The following are two UTL algorithms. The first low-rank algorithm L-ULV(L) is a natural “translation” of Stewart’s high-rank algorithm to the low-rank case so that it is “warm-started” with an initial ULV decomposition  $\mathbf{A} = \mathbf{U}\mathbf{L}\mathbf{V}^T$ . Hence, the algorithm is named with the letter “L”. Usually, the initial matrix  $\mathbf{V}$  is chosen to equal the identity. When the algorithm terminates, the matrices  $\mathbf{H}$  and  $\mathbf{E}$  are of the forms  $\mathbf{H} = [\hat{\mathbf{H}}, \mathbf{0}]^T$  and  $\mathbf{E} = [\hat{\mathbf{E}}, \mathbf{0}]^T$ , where  $\hat{\mathbf{H}}$  and  $\hat{\mathbf{E}}$  have  $(n - d)$  rows. The following is the pseudo-code of the algorithm.

ALGORITHM L-ULV(L).

Input: An  $m \times n$  matrix  $\mathbf{A}$  and rank tolerance  $\tau$ .

Output: Numerical rank  $d$ , orthogonal matrices  $\mathbf{U}$  and  $\mathbf{V}$ , and  $n \times n$  lower block-triangular matrix  $\mathbf{L}$ .

- Compute an initial ULV decomposition  $\mathbf{A} = \mathbf{U}\mathbf{L}\mathbf{V}^T$ , let  $i \leftarrow 1$ .

- Compute estimates  $\sigma_{est}^{(1)}$  and  $\mathbf{u}_{est}^{(1)}$  of the largest singular value and corresponding left singular vector of the matrix  $\mathbf{L}$ .
- While  $\sigma_{est}^{(i)} > \tau$  and  $i < n$  do
 

Determine a sequence of Givens rotations  $(\mathbf{P}_{n-i,n-i+1})_{i=1}^{n-1}$  such that  $\mathbf{P}_{n-i,n-i+1} \cdots \mathbf{P}_{1,2} \mathbf{u}_{est}^{(i)} = \mathbf{e}_1$ , where  $\mathbf{e}_1 = [1, 0, \dots, 0]^T$ .

For  $j = n - i : -1 : 2$ ,

set  $\hat{\mathbf{L}} \leftarrow \begin{pmatrix} \mathbf{I}_{i-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_{j-1,j} \end{pmatrix} \mathbf{L}$  and  $\mathbf{U} \leftarrow \mathbf{U} \begin{pmatrix} \mathbf{I}_{i-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_{j-1,j} \end{pmatrix}$ .

Determine a Givens rotation  $\mathbf{Q}_{j-1,j}$  such that  $\hat{\mathbf{L}} \begin{pmatrix} \mathbf{I}_{i-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_{j-1,j} \end{pmatrix}$  is lower triangular.

Set  $\mathbf{L} \leftarrow \hat{\mathbf{L}} \begin{pmatrix} \mathbf{I}_{i-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_{j-1,j} \end{pmatrix}$  and  $\mathbf{V} \leftarrow \mathbf{V} \begin{pmatrix} \mathbf{I}_{i-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_{j-1,j} \end{pmatrix}$ .
- Deflate by setting  $i \leftarrow i + 1$  and compute estimates  $\sigma_{est}^{(i)}$  and  $\mathbf{u}_{est}^{(i)}$  of the largest singular value and corresponding singular vector of  $\mathbf{L}(i:n, i:n)$ .

The second “cold-started” low-rank algorithm avoids the initial factorization and starts directly from the matrix  $\mathbf{A}$ . Hence, the algorithm is named with the letter “A”. This algorithm is sometimes more efficient than the previous algorithm. When ALGORITHM L-ULV(A) terminates, both  $\mathbf{H}$  and  $\mathbf{E}$  are dense matrices with  $m - d$  rows.

#### ALGORITHM L-ULV(A).

Input: An  $m \times n$  matrix  $\mathbf{A}$  and rank tolerance  $\tau$ .

Output: Numerical rank  $d$ , orthogonal matrices  $\mathbf{U}$  and  $\mathbf{V}$ , and  $n \times n$  lower block-triangular matrix  $\mathbf{L}$ .

- Initialize  $i \leftarrow 1$ ,  $\mathbf{L} \leftarrow \mathbf{A}$ ,  $\mathbf{U} \leftarrow \mathbf{I}_m$ , and  $\mathbf{V} \leftarrow \mathbf{I}_n$ .
- Compute estimates  $\sigma_{est}^{(1)}$  and  $\mathbf{u}_{est}^{(1)}$  of the largest singular value and corresponding left singular vector of the matrix  $\mathbf{L}$ .
- While  $\sigma_{est}^{(i)} > \tau$  and  $i < n$  do
 

Determine a Householder matrix  $\mathbf{P}^i$  such that  $\mathbf{P}^i \mathbf{u}_{est}^{(i)} = \mathbf{e}_1$ , where  $\mathbf{e}_1 = [1, 0, \dots, 0]^T$ .

Set  $\hat{\mathbf{L}} \leftarrow \begin{pmatrix} \mathbf{I}_{i-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}^i \end{pmatrix} \mathbf{L}$  and  $\mathbf{U} \leftarrow \mathbf{U} \begin{pmatrix} \mathbf{I}_{i-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}^i \end{pmatrix}$ .

Determine a Householder matrix  $\mathbf{Q}^i$  that annihilates elements  $i + 1$  through  $n$  of the  $i$ th row of  $\hat{\mathbf{L}}$ . Set  $\mathbf{L} \leftarrow \hat{\mathbf{L}} \begin{pmatrix} \mathbf{I}_{i-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}^i \end{pmatrix}$  and  $\mathbf{V} \leftarrow \mathbf{V} \begin{pmatrix} \mathbf{I}_{i-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}^i \end{pmatrix}$ .

- Deflate by setting  $i \leftarrow i + 1$  and compute estimates  $\sigma_{est}^{(i)}$  and  $\mathbf{u}_{est}^{(i)}$  of the largest singular value and corresponding singular vector of  $\mathbf{L}(i : n, i : n)$ .

The authors of UTV developed UTV Expansion Pack as the supplement of the package UTV Tools. In UTV Expansion Pack, rank-revealing decompositions for positive semidefinite and indefinite matrices are included. Recall that the spectral decomposition of a symmetric  $n \times n$  matrix  $\mathbf{A}$  with rank  $r$  is given as follows:

$$\mathbf{A} = \mathbf{V}\Lambda\mathbf{V}^T = (\mathbf{V}_r, \mathbf{V}_\perp) \begin{pmatrix} \Lambda_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} (\mathbf{V}_r, \mathbf{V}_\perp)^T = \mathbf{V}_r \Lambda_r \mathbf{V}_r^T, \quad (10.12)$$

where  $\mathbf{V}$  is o.g. and  $\Lambda_r$  is diagonal. We assume that the diagonal entries of  $\Lambda_r$  are descended arranged so that  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_r$ . Then the best  $d$ -rank approximation of  $\mathbf{A}$  is

$$\mathbf{A}_d = \mathbf{V}_d \Lambda_d \mathbf{V}_d^T, \quad (10.13)$$

where  $\mathbf{V}_d$  consists of the first  $d$  columns of  $\mathbf{V}$  and  $\Lambda_d = \text{diag}(\lambda_1, \dots, \lambda_d)$ .

Symmetric rank-revealing VSV decompositions provide the algorithms that take into account the symmetry of the matrix. Compared to the general UTV decompositions, the VSV decompositions lead to savings in computer time as well as advantages in the approximation properties of low-rank matrix approximations derived from the symmetric decompositions. Assume that the positive demidefinite matrix  $\mathbf{A}$  has numerical rank  $d$ , i.e.,  $\lambda_d \gg \lambda_{d+1}$  and  $\lambda_{d+1} \approx 0$ . Then the rank-revealing VSV decomposition of  $\mathbf{A}$  takes the form

$$\mathbf{A} = \mathbf{V}\mathbf{S}\mathbf{V}^T, \quad \mathbf{S} = \begin{pmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \\ \mathbf{S}_{21} & \mathbf{S}_{22} \end{pmatrix}, \quad \mathbf{V} = (\mathbf{V}_1, \mathbf{V}_2), \quad (10.14)$$

where the symmetric matrix  $\mathbf{S}$  is partitioned such that it reveals the numerical rank of  $\mathbf{A}$ , i.e., the eigenvalues of the  $d \times d$  leading submatrix  $\mathbf{S}_{11}$  approximate the first  $d$  eigenvalues of  $\mathbf{A}$ , while the norms  $\mathbf{S}_{12}$  and  $\mathbf{S}_{22}$  are both of the order  $\lambda_{d+1}$ . The matrix  $\mathbf{V}$  is orthogonal and partitioned such that the column spaces of the two blocks  $\mathbf{V}_1$  and  $\mathbf{V}_2$  approximate the subspaces spanned by the first  $d$  and the last  $n - d$  right singular vectors of  $\mathbf{A}$ , respectively.

In the case of  $d \ll n$ , UTV Expansion Pack provides the low-rank VSV algorithms `1vsvid` and `1vsvsd`, which represent the middle matrix  $\mathbf{S}$  in (10.14) with the form

$$\mathbf{S} = \mathbf{T}^T \boldsymbol{\Omega} \mathbf{T},$$

where  $\mathbf{T}$  is an upper or lower triangular matrix and  $\boldsymbol{\Omega}$  is a diagonal matrix with  $\pm 1$  on the diagonal, such that the inertia of  $\mathbf{A}$  is preserved in the inertia of  $\boldsymbol{\Omega}$ . In `1vsvsd`,  $\mathbf{A}$

is positive semi-definite with the rank  $\gg d$ , then  $\Omega$  is the identity matrix that can be omitted. The following is the pseudo-code of `1vsvid`.

Low-Rank VSV ALGORITHM.

Input: An  $n \times n$  symmetric data matrix  $\mathbf{A}$  and rank tolerance  $\tau$ .

Output: Numerical rank  $d$ , orthogonal matrices  $\mathbf{V}$ , lower triangular matrix  $\mathbf{L}$ , and signature matrix  $\mathbf{b}$ .

- Initialize  $i \leftarrow 1$ ,  $\mathbf{A} = \mathbf{L}^T \Omega \mathbf{L}$ ,  $\mathbf{V} \leftarrow \mathbf{I}_n$ .
- Compute estimates  $\sigma_d = \|\mathbf{L}(d : n, d : n)^T \Omega(d : n, d : n) \mathbf{L}(d : n, d : n)\|_2$  and the corresponding right singular vector  $\mathbf{w}_d$  associated with  $\sigma_d$ .
- If  $(\sigma_d < \tau)$  then  $k \leftarrow k - 1$ , exit.
- Revealment: determine o.g. matrix  $\mathbf{P}_d$  such that  $\mathbf{P}_d \mathbf{w}_d = \mathbf{e}_1$ ; update  $\mathbf{L}(d : n, d : n) \leftarrow L(d : n, d : n) \mathbf{P}_d$  and  $\mathbf{V}(:, d : n) \leftarrow \mathbf{V}(:, d : n) \mathbf{P}_d$ ;  
make QL factorization of  $\mathbf{L}(d : n, d : n)$ :  $\mathbf{L}(d : n, d : n) = \mathbf{Q}_d \mathbf{L}_d$ .  
update  $\mathbf{L}(d : n, d : n) \leftarrow \mathbf{L}_d$  and  $\Omega(d : n, d : n) \leftarrow \mathbf{Q}_d^T \Omega(d : n, d : n) \mathbf{Q}_d$ ;
- Deflate by setting  $d \leftarrow d + 1$ .
- Go to step 2.

The pseudo-code of `1vsvid` is similar. The Matlab source code of UTV Tools and UTV Expansion Pack can be downloaded from the web. Unfortunately, on the source code there are a few bugs that have to be fixed.

### 10.1.3 Nyström Approximation

Belabbas and Wolfe proposed a fast low-rank approximation for covariance matrices. The method used in the approximation is called Nyström approximation. The Nyström method is a technique for finding numerical approximations to eigenvectors of positive semi-definite matrices. To better understand the nature of the Nyström approximation, it is instructive to examine it from the standpoint of matrix completion. Let  $\mathbf{G}$  be an  $N \times N$  positive semidefinite matrix, of which a partition is given by

$$\mathbf{G} = \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{C} \end{pmatrix} \quad (10.15)$$

with  $\mathbf{A} \in \mathcal{M}_n$ ,  $\mathbf{B} \in \mathcal{M}_{(N-n),n}$ , and  $\mathbf{C} \in \mathcal{M}_{N-n}$ . In the case of interest,  $n \ll N$ , so  $\mathbf{C}$  is huge.  $\mathbf{G}$  is psd, so is  $\mathbf{A}$ . Furthermore, the matrix  $\mathbf{A}$  most likely is positive definite so

that  $\mathbf{A}$  has a spectral decomposition

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{U}^T, \quad (10.16)$$

where  $\Sigma$  is invertible. The Nyström extension of  $\mathbf{U}$  is defined by

$$\bar{\mathbf{U}} = \begin{pmatrix} \mathbf{U} \\ \mathbf{B}^T\mathbf{U}\Sigma^{-1} \end{pmatrix}, \quad (10.17)$$

whose columns approximate eigenvectors of  $\mathbf{G}$ . The matrix  $\tilde{\mathbf{G}} = \bar{\mathbf{U}}\Sigma\bar{\mathbf{U}}^T$  is an approximation of  $\mathbf{G}$ . It also takes the form

$$\begin{aligned} \tilde{\mathbf{G}} &= \bar{\mathbf{U}}\Sigma\bar{\mathbf{U}}^T = \begin{pmatrix} \mathbf{U} \\ \mathbf{B}^T\mathbf{U}\Sigma^{-1} \end{pmatrix} \Sigma (\mathbf{U}^T, \Sigma^{-1}\mathbf{U}^T\mathbf{B}) \\ &= \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T\mathbf{B}^T\mathbf{A}^{-1}\mathbf{B} & \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{A}^T \\ \mathbf{B}^T \end{pmatrix} \mathbf{A}^{-1}(\mathbf{A}, \mathbf{B}). \end{aligned} \quad (10.18)$$

Thus, the Nyström extension implicitly approximates  $\mathbf{C}$  by  $\mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}$ . The quality of the approximation can be quantified by the norm of the Schur complement  $\|\mathbf{C} - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}\|_2$ .

Note that the columns of  $\bar{\mathbf{U}}$  in (10.17) are not orthogonal. To find an o.g. approximation of  $\mathbf{U}$ , we apply the following.

**定理215.** Let  $\mathbf{G}$  be a psd matrix having the partition (10.16), in which  $\mathbf{A}$  is pd. Define

$$\mathbf{S} = \mathbf{A} + \mathbf{A}^{-1/2}\mathbf{B}\mathbf{B}^T\mathbf{A}^{-1/2}$$

and decompose it as  $\mathbf{S} = \mathbf{U}\Lambda\mathbf{U}^T$ . Let

$$\mathbf{V} = \begin{pmatrix} \mathbf{A} \\ \mathbf{B}^T \end{pmatrix} \mathbf{A}^{-1/2}\mathbf{U}\Lambda^{-1/2}. \quad (10.19)$$

Then the columns of  $\mathbf{V}$  form an o.n. system, i.e.,  $\mathbf{V}^T\mathbf{V} = \mathbf{I}$ , and

$$\|\mathbf{G} - \mathbf{V}\Lambda\mathbf{V}^T\|_2 = \|\mathbf{C} - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}\|_2. \quad (10.20)$$

*Proof.* We have

$$\begin{aligned} \mathbf{V}^T\mathbf{V} &= (\Lambda^{-1/2}\mathbf{U}^T\mathbf{A}^{-1/2}(\mathbf{A}, \mathbf{B})) \left( \begin{pmatrix} \mathbf{A} \\ \mathbf{B}^T \end{pmatrix} \mathbf{A}^{-1/2}\mathbf{U}\Lambda^{-1/2} \right) \\ &= (\Lambda^{-1/2}\mathbf{U}^T\mathbf{A}^{-1/2})(\mathbf{A}^2 + \mathbf{B}\mathbf{B}^T)(\mathbf{A}^{-1/2}\mathbf{U}\Lambda^{-1/2}) \\ &= (\Lambda^{-1/2}\mathbf{U}^T)\mathbf{S}(\mathbf{U}\Lambda^{-1/2}) \\ &= \mathbf{I} \end{aligned} \quad (10.21)$$

and

$$\begin{aligned}
 \mathbf{V}\Lambda\mathbf{V}^T &= \begin{pmatrix} \mathbf{A} \\ \mathbf{B}^T \end{pmatrix} \mathbf{A}^{-1/2} \mathbf{U} (\mathbf{U}^T \mathbf{A}^{-1/2}) (\mathbf{A}, \mathbf{B}) \\
 &= \begin{pmatrix} \mathbf{A} \\ \mathbf{B}^T \end{pmatrix} \mathbf{A}^{-1} (\mathbf{A}, \mathbf{B}) \\
 &= \tilde{\mathbf{G}}.
 \end{aligned} \tag{10.22}$$

The proof is completed.  $\square$

#### 10.1.4 Greedy Low-rank Approximation

The analysis above tells us that it is possible to select some columns of  $\mathbf{G}$  to construct an approximation of  $\mathbf{G}$ . Without loss of generality, we assume that the first  $k$  columns of  $\mathbf{G}$  are the best selection for the construction of the approximation. Let  $\mathbf{A} = [\mathbf{a}_1, \dots, \mathbf{a}_n]$  be the matrix such that  $\mathbf{G} = \mathbf{A}\mathbf{A}^T$ . Write

$$\mathbf{G} = \sum_{j=1}^n \mathbf{a}_j \mathbf{a}_j^T.$$

Then the best approximation of  $\mathbf{G}$  constructed by its first  $k$  columns is formulated so as to find a weight vector  $\mathbf{w} = [w_1, \dots, w_k]^T$  such that

$$\mathbf{w} = \underset{\mathbf{w} \in \mathbb{R}^k}{\operatorname{argmin}} \left\| \mathbf{G} - \sum_{j=1}^k w_j \mathbf{a}_j \mathbf{a}_j^T \right\|_F. \tag{10.23}$$

Let  $\mathbf{Q}$  be the Hadamard square of  $\mathbf{G}$  such that

$$\mathbf{Q} = \mathbf{G} \boxtimes \mathbf{G} = [\mathbf{Q}_{i,j}] = [\mathbf{G}_{i,j}^2], \tag{10.24}$$

which is partitioned into

$$\mathbf{Q} = \begin{pmatrix} \mathbf{Q}_k & \mathbf{Y} \\ \mathbf{Y}^T & \mathbf{Z} \end{pmatrix}, \tag{10.25}$$

where  $\mathbf{Q}_k \in \mathcal{S}_k$  is the first major principal submatrix of  $\mathbf{Q}$ . Define the vector  $\mathbf{r} \in \mathbb{R}^k$  by

$$\mathbf{r} = [\mathbf{Q}_k, \mathbf{Y}] \mathbf{1}. \tag{10.26}$$

Applying

$$\|\mathbf{G}\|_F^2 = \sum_{i,j=1}^n \mathbf{Q}_{i,j}, \quad \operatorname{tr}(\mathbf{G} \mathbf{a}_i \mathbf{a}_i^T) = \sum_{j=1}^n \mathbf{Q}_{i,j},$$

we have

$$\left\| \mathbf{G} - \sum_{j=1}^k w_j \mathbf{a}_j \mathbf{a}_j^T \right\|_F^2 = \mathbf{w}^T \mathbf{Q}_k \mathbf{w} - 2 \langle \mathbf{w}, \mathbf{r} \rangle + \sum_{i,j=1}^n \mathbf{Q}_{i,j},$$

which achieves the minimum at

$$\mathbf{w} = \mathbf{Q}_k^{-1}\mathbf{r}.$$

Let

$$\tilde{\mathbf{G}} = \sum_{j=1}^k w_j \mathbf{a}_j \mathbf{a}_j^T \quad (10.27)$$

and

$$\tilde{\mathbf{Q}} = (\mathbf{Q}_k, \mathbf{Y})^T \mathbf{Q}_k^{-1} (\mathbf{Q}_k, \mathbf{Y}).$$

We have

$$\|\mathbf{G} - \tilde{\mathbf{G}}\|_F^2 = \mathbf{1}^T (\mathbf{Q} - \tilde{\mathbf{Q}}) \mathbf{1},$$

which provides the error estimate of the approximation. We state the obtained result in the following theorem.

**定理216.** Let  $\mathbf{G}$  be a positive semi-definite matrix and  $\tilde{\mathbf{G}}$  be constructed as in (10.27). Let  $\mathbf{Q}$ ,  $\mathbf{Q}_k$ ,  $\mathbf{Y}$ , and  $\mathbf{Z}$  be defined as in (10.24) and (10.25). Let  $\mathbf{E}$  be the  $(n-k) \times (n-k)$  matrix with all entries equal to 1. Then

$$\|\mathbf{G} - \tilde{\mathbf{G}}\|_F^2 = \text{tr}((\mathbf{Z} - \mathbf{Y}^T \mathbf{Q}_k \mathbf{Y}) \mathbf{E}). \quad (10.28)$$

By Theorem 216, the greedy low-rank matrix approximation is designed as follows.

Assume that the positive semidefinite  $n \times n$  matrix  $\mathbf{G}$  and the dimension  $k$  are given ( $k < n$ ).

Step 1. Select the greedy columns. Write  $\mathbf{G} = [\mathbf{g}_1, \dots, \mathbf{g}_n]$ . Compute the norm vector  $\mathbf{T} = [\|\mathbf{g}_i\|^2]_{i=1}^n$  and find the index vector  $J = [j_1, \dots, j_k]$ , where the columns have  $k$  largest norms ( $k < n$ ).

Step 2. Construct  $\mathbf{Q}$  and  $\mathbf{Q}_k$ . Set  $\mathbf{Q}_{i,j} = \mathbf{Q}_{i,j}^2$ ,  $1 \leq i, j \leq n$  and  $\mathbf{Q}_k = [\mathbf{Q}_{j,l}]_{j,l \in J}$ .

Step 3. Compute vector  $\mathbf{r}$ . Set  $\mathbf{r}_l = \sum_{j=1}^n \mathbf{Q}_{l,j}$ ,  $l \in J$ .

Step 4. Compute weight vector  $\mathbf{w}$ . Set  $\mathbf{w} = \mathbf{Q}_k^{-1}\mathbf{r}$ .

Step 5. Compute  $\mathbf{F}$  so that  $\mathbf{FF}^T$  approximates  $\mathbf{G}$ . Set  $\mathbf{D} = \text{diag}(\mathbf{w})$ ,  $\mathbf{G}_k = [\mathbf{g}_{j_1}, \dots, \mathbf{g}_{j_k}]$ , and  $\mathbf{F} = \mathbf{G}_k \mathbf{D}$ . Compute the SVD of  $\mathbf{F} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T$ , where  $\mathbf{U} \in \mathcal{O}_{n,k}$ .

Step 6. Construct the approximation  $\tilde{\mathbf{G}}$ . Set  $\tilde{\mathbf{G}} = \mathbf{U} \mathbf{U}^T \mathbf{G} \mathbf{U} \mathbf{U}^T$ .

## 10.2 Randomized Algorithm for Matrix Approximation

### 10.2.1 Randomized Low-rank Approximation

The low-rank approximation algorithms introduced in the last section are deterministic. There exist also randomized algorithms for matrix approximation. Randomized algorithms usually are faster than the deterministic ones when they are applied to solve the same problems. As a trade-off, the randomized algorithms are successful in a probability.

We remark that JL-embedding algorithms introduced in Chapter 4.4 cannot be directly applied to low-rank approximation. Firstly, by Theorems 80 and 88, the dimension  $d$  of the reduced data has to obey the inequality (4.33), it cannot be as small as desired. For instance, if  $n = 10000$  and  $\varepsilon = 1/2$ , then, by the estimate in Theorem 80 (also in Theorem 88), the dimension  $d$  has to be restricted to  $d > 331$ , which is still quite large in many tasks. Secondly, JL-embedding algorithms do not have the functionality of locating feature vectors, and therefore are not reliable in data-geometry preservation. Martinsson, Rokhlin, and Tygert proposed a randomized algorithm for approximating SVD decomposition. In their algorithm, a random projection maps the original high-dimensional matrix onto a low-dimensional one, so that the SVD of the transformed matrix approximates the SVD of the original one. More precisely, let  $\mathbf{X}$  be a given  $m \times n$  matrix, where both  $m$  and  $n$  are very large. Let  $\mathcal{B} = \{\mathbf{b}_1, \dots, \mathbf{b}_k\}$  be an o.n. basis of a random  $k$ -dimensional subspace  $S_k \subset \mathbb{R}^m$ , where  $k \ll \min(m, n)$ . Let  $\mathbf{B} = [\mathbf{b}_1, \dots, \mathbf{b}_k]$ . The projections of the column vectors of  $\mathbf{X}$  onto  $S_k$  have the  $\mathcal{B}$ -coordinates  $\mathbf{X}_B = \mathbf{B}^T \mathbf{X}$ . It can be proved that  $\mathbf{B} \mathbf{X}_B$  is a  $k$ -rank revealing approximation of  $\mathbf{X}$  with a certain probability. Then the SVD of  $\mathbf{B} \mathbf{X}_B$  approximates the SVD of  $\mathbf{X}$ . Note that matrix  $\mathbf{X}_B$  has the dimension  $k \times n$ , where  $k$  is much smaller than both  $m$  and  $n$ . Hence, the cost for computing the SVD of  $\mathbf{X}_B$  is much lower than the cost for the SVD of  $\mathbf{X}$ . The project technique used above can also be replaced by interpolation.

There are two randomized algorithms for low-rank approximation of matrices. One adopts interpolative decompositions, the other uses projections. We shall introduce them in the next two subsections. In the following, we assume that the matrix  $\mathbf{A}$  has dimension  $m \times n$  and rank  $r$  with  $r \ll \min(m, n)$ , and both  $m$  and  $n$  are large. We denote by  $\sigma_j$  the  $j$ th singular value of  $\mathbf{A}$ .

### 10.2.2 Randomized Interpolative Algorithm

The randomized interpolative low-rank approximation produces an interpolative  $m \times k$  matrix  $\mathbf{P}$  and a  $k \times n$  matrix  $\mathbf{B}$ , which are similar to  $\mathbf{P}$  and  $\mathbf{B}$  in Theorem 213.

The existing algorithms for computing  $\mathbf{B}$  and  $\mathbf{P}$  in Theorem 213 are computationally expensive. Some authors suggested to use the rank-revealing algorithms to produce  $\mathbf{B}$  and  $\mathbf{P}$ , which satisfy somewhat weaker conditions than those in Theorem 213. The algorithms compute  $\mathbf{B}$  and  $\mathbf{P}$  such that

- some subset of the columns of  $\mathbf{P}$  makes up the  $k \times k$  identity matrix,
- no entry of  $\mathbf{P}$  has an absolute value greater than 2,
- $\|\mathbf{P}\|_2 \leq c_{n,k}$ ,
- the least singular value of  $\mathbf{P}$  is at least 1,
- $\mathbf{BP} = \mathbf{A}$  when  $k = m$  or  $k = n$ , and
- $\|\mathbf{BP} - \mathbf{A}\| \leq c_{n,k}\sigma_{k+1}$  when  $k < \min(m, n)$ ,

where

$$c_{n,k} = \sqrt{4k(n-k) + 1}. \quad (10.29)$$

The algorithms of this type are based upon the Cramer's rule to obtain the minimal norm (or at least nearly minimal-norm) solutions of linear systems. Comparing the conditions above with those in Theorem 213, we can find two differences that (1) the upper bound  $\|\mathbf{P}\|_2 \leq k(n-k) + 1$  now is relaxed to  $\|\mathbf{P}\|_2 \leq c_{n,k}$  and (2)  $\max(|\mathbf{P}_{i,j}|) \leq 1$  is relaxed to  $\max(|\mathbf{P}_{i,j}|) \leq 2$ . Besides, for any positive real number  $\varepsilon$ , the algorithms can identify the least  $k$  such that  $\|\mathbf{BP} - \mathbf{A}\| \approx \varepsilon$ . An algorithm of this type is computationally economic: there exists a positive real number  $C$  so that the algorithm computes both  $\mathbf{B}$  and  $\mathbf{P}$  using at most  $Ckmn \log(n)$  floating-point operations.

Let  $\mathbf{A}$  be a given high-dimensional matrix. According to the discussion above, a randomized algorithm of interpolative low-rank approximation uses a random projection to produce a low-dimensional projection of  $\mathbf{A}$ , then applies a rank-revealing algorithm to find the submatrix  $\mathbf{B}$  of  $\mathbf{A}$  and the corresponding interpolative matrix  $\mathbf{P}$ , such that  $\mathbf{BP}$  is a low-rank approximation of  $\mathbf{A}$ . The randomized interpolative algorithm consists of the following steps.

Assume that an  $m \times n$  matrix  $\mathbf{A}$  is given. Both  $m$  and  $n$  are large. We set  $l < \min(m, n)$  and  $k < l$ .

Step 1. Create a random projection of  $\mathbf{A}$ . Create an  $l \times m$  random matrix  $\mathbf{G}$  whose entries are i.i.d. Gaussian random variables of zero mean and unit variance, and compute the  $l \times n$  product matrix  $\mathbf{R} = \mathbf{GA}$ .

Step 2. Find the interpolative matrix  $\mathbf{P}$ . Using a rank-revealing algorithm to create a real  $l \times k$  matrix  $\mathbf{S}$  and a real  $k \times n$  matrix  $\mathbf{P}$ , where the columns of  $\mathbf{S}$  constitute a subset of the columns of  $\mathbf{R}$  and  $\mathbf{P}$  satisfies  $\|\mathbf{P}\|_2 \leq c_{n,k}$ , such that

$$\|\mathbf{SP} - \mathbf{R}\|_2 \leq c_{n,k} \rho_{k+1}. \quad (10.30)$$

where  $\rho_{k+1}$  is the  $(k+1)$ st greatest singular value of  $\mathbf{R}$ .

Step 3. Find the submatrix  $\mathbf{B}$  of  $\mathbf{A}$ . Due to Step 2, the columns of  $\mathbf{S}$  constitute a subset of the columns of  $\mathbf{R}$ . Let  $J = \{i_1, \dots, i_k\}$  be the index set such that the  $j$ th column of  $\mathbf{S}$  is the  $i_j$ th column of  $\mathbf{R}$ . Create an  $m \times k$  submatrix  $\mathbf{B}$  of  $\mathbf{A}$ , so that the  $j$ th column of  $\mathbf{B}$  is the  $i_j$ th column of  $\mathbf{A}$ . The matrix  $\mathbf{B}$  can be computed by  $\mathbf{B} = \mathbf{AP}^T$ .

Then  $\tilde{\mathbf{A}} \triangleq \mathbf{BP} = \mathbf{AP}^T \mathbf{P}$  is a  $k$ -low-rank approximation of  $\mathbf{A}$ .

We give the error estimate of the approximation realized by the algorithm above. Let  $\beta$  and  $\gamma$  be two positive real numbers such that  $\gamma > 1$  and define

$$\begin{aligned} \tilde{p}(l, k, \beta) &= \frac{1}{\sqrt{2\pi(l-k+1)}} \left( \frac{e}{(l-k+1)\beta} \right)^{l-k+1}, \\ p(x, \gamma) &= \frac{1}{4(\gamma-1)\sqrt{\pi x \gamma}} \left( \frac{2\gamma}{e^{\gamma-1}} \right)^x. \end{aligned} \quad (10.31)$$

We have the following error estimate of the approximation given by the algorithm above.

**定理217.** *Let  $\mathbf{A}$  be an  $m \times n$  matrix,  $k \leq l < \min(n, m)$ . Let  $\mathbf{P}$  and  $\mathbf{B}$  be the matrices that are produced by the randomized algorithm of interpolative low-rank approximation above. Then*

$$\|\mathbf{A} - \mathbf{BP}\|_2 \leq \left( \sqrt{2lm\beta\gamma + 1}(c_{n,k} + 1) + \sqrt{2lm\beta\gamma}c_{n,k} \right) \sigma_{k+1} \quad (10.32)$$

with probability at least  $\chi$ , where  $c_{n,k}$  is in (10.29),  $\sigma_{k+1}$  is the  $(k+1)$ th singular value of  $\mathbf{A}$ , and

$$\chi = 1 - \tilde{p}(l, k, \beta) - 2p(m, \gamma). \quad (10.33)$$

When the singular values decay fast. We can get the following more accurate estimate.

**定理218.** *Let  $\mathbf{A}$ ,  $\mathbf{P}$ , and  $\mathbf{B}$  be the matrices as in Theorem 217. Assume that  $j > 0$  satisfies  $k+j < \min(m, n)$ . Write  $m_1 = \max(m-k-j, l)$ ,  $m_2 = \max(j, l)$ , and  $m_3 = \max(j+k, l)$ . Then*

$$\|\mathbf{A} - \mathbf{BP}\|_2 \leq \xi \sigma_{k+1} + \eta \sigma_{k+j+1} \quad (10.34)$$

with probability at least  $\kappa$ , where

$$\kappa = 1 - \tilde{p}(l, k, \beta) - 2p(m_1, \gamma) - p(m_2, \gamma) - p(m_3, \gamma) \quad (10.35)$$

and

$$\begin{aligned} \xi &= \sqrt{2l\beta\gamma m_2 + 1}(c_{n,k} + 1) + \sqrt{2l\beta\gamma m_3}c_{n,k}, \\ \eta &= \sqrt{2l\beta\gamma m_1 + 1}(c_{n,k} + 1) + \sqrt{2l\beta\gamma m_1}c_{n,k}. \end{aligned} \quad (10.36)$$

Moreover, if  $\eta\sigma_{k+j+1}$  is less than  $\xi\sigma_{k+1}$ , then

$$\|\mathbf{A} - \mathbf{B}\mathbf{P}\|_2 \leq 2\xi\sigma_{k+1} \quad (10.37)$$

with probability at least  $\kappa$ .

In practice, we usually choose  $l = k + 20$ . In this case we have the following.

**推论219.** *In Theorem 217, if we choose  $l = k + 20$ ,  $\beta = 9/16$ , and  $\gamma = 5$ , then*

$$\|\mathbf{A} - \mathbf{B}\mathbf{P}\|_2 < 10\sqrt{k(k+20)m}\sigma_{k+1}.$$

with probability not less than  $1 - 10^{-17}$ .

### 10.2.2.1 Randomized SVD Algorithm

The randomized algorithm of SVD low-rank approximation uses a random projection to project  $\mathbf{A}$  on a random subspace of  $\text{Row}(\mathbf{A})$ , then applies an SVD algorithm to yield the SVD of the projection of  $\mathbf{A}$ . Since the projected matrix has a much lower dimension than  $\mathbf{A}$ , the algorithm performs fast. The randomized algorithm consists of the following steps.

Assume that an  $m \times n$  matrix  $\mathbf{A}$  is given. Both  $m$  and  $n$  are large. We set  $l < \min(m, n)$  and  $k < l$ .

Step 1. Create a random projection of  $\mathbf{A}$ . Create an  $l \times m$  random matrix  $\mathbf{G}$  whose entries are i.i.d. Gaussian random variables of zero mean and unit variance, and compute the  $l \times n$  product matrix  $\mathbf{R} = \mathbf{G}\mathbf{A}$ .

Step 2. Find approximative principal subspace  $\mathbf{L}$ . Using an SVD algorithm to find a real  $n \times k$  matrix  $\mathbf{Q}$  and a  $k \times l$  matrix  $\mathbf{S}$ , where the columns of  $\mathbf{Q}$  are orthonormal and

$$\|\mathbf{QS} - \mathbf{R}^T\|_2 \leq \rho_{k+1},$$

where  $\rho_{k+1}$  is the  $(k+1)$ st greatest singular value of  $\mathbf{R}$ . Let  $\mathbf{L} = \text{Col}(\mathbf{Q})$ , which approximates the the principal  $k$ -subspace of  $\mathbf{A}$ . The columns of  $\mathbf{Q}$  form an o.n. basis of  $\mathbf{L}$ .

Step 3. Find o.g. projection  $\mathbf{T}$  from  $\mathbf{A}$  to  $\mathbf{L}$ .  $\mathbf{T}$  is given by  $\mathbf{T} = \mathbf{AQ}$ .

Step 4. Find SVD of  $\mathbf{T}$ . Let  $\mathbf{T} = \mathbf{U}\Sigma\mathbf{W}^T$  be the SVD of  $\mathbf{T}$ .

Step 5. Find low-rank approximation of SVD of  $\mathbf{A}$ . Let  $\mathbf{V} = \mathbf{Q}\mathbf{W}$ . Then the triple  $[\mathbf{U}, \Sigma, \mathbf{V}]$  is a low-rank SVD approximation of  $\mathbf{A}$ :  $\mathbf{U}\Sigma\mathbf{V}^T \approx \mathbf{A}$ .

We have the following error estimate of the approximation given by the algorithm above.

**定理220.** *Let  $\mathbf{A}$  be an  $m \times n$  matrix,  $k \leq l < \min(n, m)$ . Let the triple  $[\mathbf{U}, \Sigma, \mathbf{V}]$  is produced by the randomized algorithm of low-rank SVD approximation above. Then*

$$\|\mathbf{A} - \mathbf{U}\Sigma\mathbf{V}\|_2 \leq 2 \left( \sqrt{2lm\beta\gamma + 1} + \sqrt{2lm\beta\gamma} \right) \sigma_{k+1} \quad (10.38)$$

with probability at least  $\chi$ .

When the singular values of  $\mathbf{A}$  decay fast, a better error estimate is given in the following theorem.

**定理221.** *Let  $\mathbf{A}$ ,  $\mathbf{U}$ ,  $\Sigma$ , and  $\mathbf{V}$  be the matrices in Theorem 220, and let  $j > 0$  satisfy  $k+j < \min(m, n)$ . Write  $m_1 = \max(m-k-j, l)$ ,  $m_2 = \max(j, l)$ , and  $m_3 = \max(j+k, l)$ . Then*

$$\|\mathbf{A} - \mathbf{U}\Sigma\mathbf{V}\|_2 \leq \xi\sigma_{k+1} + \eta\sigma_{k+j+1} \quad (10.39)$$

with probability at least  $\kappa$  given in (10.35), and

$$\begin{aligned} \xi &= 2 \left( \sqrt{2l\beta\gamma m_2 + 1} + \sqrt{2l\beta\gamma m_3} \right), \\ \eta &= 2 \left( \sqrt{2l\beta\gamma m_1 + 1} + \sqrt{2l\beta\gamma m_1} \right). \end{aligned} \quad (10.40)$$

Moreover, if  $\eta\sigma_{k+j+1}$  is less than  $\xi\sigma_{k+1}$ , then

$$\|\mathbf{A} - \mathbf{U}\Sigma\mathbf{V}\|_2 \leq 2\xi\sigma_{k+1} \quad (10.41)$$

with probability at least  $\kappa$ .

Similarly, we have the following.

**推论222.** *In Theorem 220, if we choose  $l = k + 20$ ,  $\beta = 9/16$ , and  $\gamma = 5$ , then*

$$\|\mathbf{A} - \mathbf{U}\Sigma\mathbf{V}\|_2 \leq 10\sqrt{k(k+20)m}\sigma_{k+1} \quad (10.42)$$

with probability not less than  $1 - 10^{-17}$ .

### 10.2.3 Randomized Greedy Algorithm

The deterministic greedy low-rank approximation algorithm introduced in the last section can be modified to a randomized one. The idea is very simple: Change the greedy selection of columns in  $\mathbf{G}$  to a random selection. Assume that  $\mathbf{G}$  is an  $n \times n$  positive semi-definite matrix. The algorithm randomly selects an index set  $J = \{i_1, \dots, i_k\}$  for the approximation. Since  $\mathbf{G}$  is positive semi-definite, it induces that the probability distribution on the set of all  $J$  with  $|J| = k$  is

$$p(J) = \frac{\det(\mathbf{G}_J)}{\sum_{|L|=k} \det(\mathbf{G}_L)}, \quad (10.43)$$

where  $\det(\mathbf{G}_L)$  is the determinant of the submatrix  $\mathbf{G}_L$  whose rows and columns are chosen from the index set  $\mathbf{L}$ .

The randomized greedy algorithm for low-rankmatrix approximation consists of the same steps as the deterministic one except that the index now is randomly selected. The algorithm is well behaved in the sense that if  $\mathbf{G}$  is of rank  $k$ , then  $\tilde{\mathbf{G}} = \mathbf{G}$ . (see page 316 of [133] for proof)

For the general case whereupon  $\text{rank}(\mathbf{G}) > k$ , we have the following error bound in expectation.

**定理223.** Let  $\mathbf{G}$  be a real,  $n \times n$ , positive semidefinite matrix with eigenvalues  $\lambda_1 \geq \dots \geq \lambda_n$ . Let  $\tilde{\mathbf{G}}$  be the Nyström approximation to  $\mathbf{G}$  corresponding to  $J$ , with  $p(J)$  as given in (10.43). Then

$$E(\|\mathbf{G} - \tilde{\mathbf{G}}\|_F) \leq (k+1) \sum_{l=k+1}^n \lambda_l,$$

with the probability  $p(J)$ .

## 10.3 Fast Anisotropic Transformation DR Algorithms

In this section, we introduce fast anisotropic transformation (FAT) DR algorithms, which is based on low-rank approximation technique.

### 10.3.1 Fast Anisotropic Transformation

FAT provides a framework for developing fast DR algorithms. Recall that in each of nonlinear DR methods introduced previously, the kernel is an  $n \times n$  matrix, where  $n$  is the total number of points in the observed data, which can be very large in many applications. For example, even for a small HSI cube with  $560 \times 480 = 268800$  raster

cells, when the spectral curves are selected as the objective vectors, the number of entries of a nonlinear DR kernel is almost  $7 \times 10^{10}$ . Hence, when nonlinear DR methods are applied to a larger data set, they usually encounter difficulties in the three aspects: memory usage, computational complexity, and computational instability. Although some nonlinear DR kernels are sparse matrices, which enable us partially to overcome the difficulties in memory usage and computational complexity, yet it is not clear if the instability issue can be settled. The aim of fast anisotropic transforms is to avoid spectral decomposition of large-dimensional DR kernels to dramatically reduce memory usage and computational complexity, as well as to increase numerical stability.

FAT is based on the observation of the particular structures of DR kernels. Over all of the discussed nonlinear DR kernels, we can computationally classify them into two types. In the first type are Isomaps, MVU, and Dmaps, which find the DR data from the leading eigenvectors of their kernels. In the second type are LLE, LTSA, Leigs, and HLLE, which find the DR data from their bottom eigenvectors. To find the DR data, the methods of the first type establish maximization models while the methods of the second type adopt minimization models. A modeling duality view of nonlinear DR methods is discussed in [141]. Using the duality of maximization and minimization, we can convert the minimization problems in LLE, LTSA, Leigs, and HLLE to maximization ones. Therefore, in the following discussion, we assume that the DR kernels of the second type have been converted so that the DR data in these methods are also obtained from the several leading eigenvectors of the converted kernels. A slight difference still exists for such kernels. For example, in Isomaps the DR data are obtained from the  $d$ -leading eigenvectors, but in Dmaps from the 2nd to the  $(d + 1)$ th ones. Surely, this difference does not impact our FAT framework very much.

The main idea of FAT is similar to diffusion maps. Assume that a data set  $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subset \mathbb{R}^D$  (nearly) resides on a  $d$ -dimensional manifold  $M \subset \mathbb{R}^D$ , whose geometry is described by a DR kernel  $\mathbf{K}$ . Since  $\mathbf{K}$  is positive semi-definite, we can define the kernel distance between  $\mathbf{x}_i$  and  $\mathbf{x}_j$  by

$$d_{\mathbf{K}}^2(\mathbf{x}_i, \mathbf{x}_j) = k_{ii} + k_{jj} - 2k_{ij}. \quad (10.44)$$

Assume the rank of  $\mathbf{K}$  is  $m$ . In the ideal case  $m = d$ . Otherwise  $d < m$ . Let a Cholesky decomposition of  $\mathbf{K}$  be

$$\mathbf{K} = \mathbf{P}^T \mathbf{P},$$

where  $\mathbf{P} = [\mathbf{p}_1, \dots, \mathbf{p}_n] \in \mathcal{M}_{m,n}$  corresponds to the data  $\mathcal{P} = \{\mathbf{p}_1, \dots, \mathbf{p}_n\} \subset \mathbb{R}^m$ . Then

$$d_2(\mathbf{p}_i, \mathbf{p}_j) = d_{\mathbf{K}}(\mathbf{x}_i, \mathbf{x}_j).$$

Let  $\mathbf{h} = [\mathbf{h}_1, \dots, \mathbf{h}_d]$  be the manifold coordinate mapping. Then the DR data are represented by  $\mathcal{Y} = h(\mathcal{X})$ . If  $h$  is an isometric,  $\mathcal{Y}$  exactly preserves the kernel distance on  $\mathcal{X}$ ,

$$d_2(\mathbf{y}_i, \mathbf{y}_j) = d_{\mathbf{K}}(\mathbf{x}_i, \mathbf{x}_j),$$

which yields

$$d_2(\mathbf{y}_i, \mathbf{y}_j) = d_2(\mathbf{p}_i, \mathbf{p}_j).$$

In general, the equation above only holds in the approximative sense. Let the SVD of  $\mathbf{P}$  be

$$\mathbf{P} = \sum_{i=1}^m \sigma_i \mathbf{u}_i \mathbf{v}_i^T.$$

Write  $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_m]$ . Then  $\mathbf{Y}$  can be chosen as  $\mathbf{Y} = [\mathbf{v}_1, \dots, \mathbf{v}_d]^T$ . Note that the column space  $S = \text{Col}(\mathbf{V})$  (which is equal to  $\text{Row}(\mathbf{P})$ ) is an  $m$  dimensional subspace of  $\mathbb{R}^n$ , which “wraps” coordinate functions  $h^1, \dots, h^d$ .

**定义224.** Let  $\mathcal{X} \subset \mathbb{R}^D$  be a given data set, which resides on a  $d$ -dimensional manifold  $M \subset \mathbb{R}^D$ . Let  $h = [h^1, \dots, h^d]^T$  be a coordinate mapping on  $M$ . An  $m$ -dimensional subspace  $S \subset \mathbb{R}^n$  is called a wrapped coordinate subspace if  $h^i = h^i(\mathcal{X}) \in S$ ,  $1 \leq i \leq d$ . A function  $f = [f^1, \dots, f^m]^T$  is called an anisotropic transformation if the vector set  $\{f^i = f^i(\mathcal{X})\}_{1 \leq i \leq m}$  spans  $S$ . Let  $\mathbf{p}_j = f(\mathbf{x}_j)$ ,  $1 \leq j \leq n$ . Then  $\mathbf{p}_j$  is called a wrapped-coordinate representation of  $\mathbf{x}_j$  and  $\mathbf{P} = [\mathbf{p}_1, \dots, \mathbf{p}_n]$  is called a wrapped DR data matrix.

An FAT algorithm first constructs the wrapped DR data matrix  $\mathbf{P}$ , and then finds the DR data  $\mathbf{Y}$  from  $\mathbf{P}$  using a linear method, say, PCA. Since the SVD decomposition in an FAT algorithm is only applied to the low-dimensional matrix  $\mathbf{P}$ , the algorithm is fast. Note that the wrapped DR data matrix  $\mathbf{P}$  is not unique. There are many different “wraps” for the same DR data. The variety of  $\mathbf{P}$  offers the freedom for developing various FAT DR algorithms.

### 10.3.2 Greedy Anisotropic Transformation

The first type of FAT algorithms is based on the greedy low-rank approximation. Hence, we call them greedy anisotropic transformation (GAT) algorithms, which are described as follows.

Let  $\mathbf{P} = [\mathbf{p}_1, \dots, \mathbf{p}_n] \in \mathcal{M}_{m,n}$  be a wrapped DR data matrix. The relation  $\mathbf{K} = \mathbf{P}^T \mathbf{P}$  yields  $\text{Row}(\mathbf{P}) = \text{Col}(\mathbf{K})$ . Therefore, there are  $m$  columns in  $\mathbf{K}$  spanning  $\text{Row}(\mathbf{P})$ . GAT algorithm selects the  $m$  greedy columns of  $\mathbf{K}$  to (approximately) span  $\text{Row}(\mathbf{P})$ . Write  $\mathbf{K} = [\mathbf{k}_1, \dots, \mathbf{k}_n]$ . Without losing generality, we assume that  $\mathbf{k}_1, \dots, \mathbf{k}_m$  are  $m$  greedy

columns of  $\mathbf{K}$ . Denote by  $S$  the subspace spanned by these vectors. Let  $\{\mathbf{b}_1, \dots, \mathbf{b}_m\}$  be an o.n. basis of  $S$ . Then  $\mathbf{p}_j$  can be chosen as the o.g. projection of  $\mathbf{x}_j$  on  $S$ , represented in  $\mathcal{B}$ -coordinates. The SVD of  $\mathbf{P}$  yields the DR data matrix  $\mathbf{Y}$ .

**备注225.** *If the  $m$  columns of  $\mathbf{K}$  are chosen randomly, then we obtain a randomized greedy anisotropic transformation algorithm, which will be discussed in the next subsection.*

The pseudo-code of the GAT algorithm is the following.

Input: the positive semi-definite  $n \times n$  kernel  $\mathbf{K}$ , the dimension of wrapped coordinate space  $m$ , and the dimension of the DR data  $d$ .

Output: the DR data matrix  $\mathbf{Y}$  and the corresponding leading eigenvalue set  $\mathbf{ev}$ .

Step 1. Find greedy columns. Compute the norm of each column vector of  $\mathbf{K}$  and construct the  $n \times m$  submatrix  $\mathbf{K}_m$  that consists of the  $m$  columns that have  $m$  largest norms.

Step 2. Make wrapped DR data matrix. Make the QR factorization of  $\mathbf{K}_m$  to obtain a matrix  $\mathbf{Q} \in \mathcal{O}_{n,m}$  whose columns form an o.n. basis of  $\text{Col}(\mathbf{K}_m)$ . Then the wrapped DR data matrix is  $\mathbf{A} = (\mathbf{K}\mathbf{Q})^T$ .

Step 3. Find DR data. Let the SVD decomposition of  $\mathbf{A}$  be

$$\mathbf{A} = \sum_{i=0}^m \sigma_i \mathbf{u}_i \mathbf{v}_i^T,$$

where  $\sigma_0 > \sigma_1 \geq \dots \geq \sigma_m > 0$ . If  $\mathbf{K}$  is an Isomaps-type kernel, then  $\mathbf{Y}^T = [\mathbf{u}_0, \dots, \mathbf{u}_{d-1}]$ . Otherwise, if  $\mathbf{K}$  is a Dmaps-type kernel, we have  $\sigma_0 = 1$ . Setting  $\mathbf{Y}^T = [\mathbf{u}_1, \dots, \mathbf{u}_d]$ , and then we entry-wise divide each  $\mathbf{u}_j$  by  $\mathbf{u}_0$ .

### 10.3.3 Randomized Anisotropic Transformation

Randomized anisotropic transformation (RAT) algorithms construct the approximative wrapped-coordinate subspace  $S$  by random projection or random interpolation. The randomized interpolative anisotropic transformation (I-RAT) algorithm randomly selects  $m$  columns of  $\mathbf{K}$  to span the subspace  $S$ , while the randomized project anisotropic transformation (P-RAT) algorithm randomly selects an  $m$ -subspace of  $\text{Col}(\mathbf{K})$  to be the subspace  $S$ . Once the subspace  $S$  is constructed, the remaining steps in the algorithms are the same as in GAT algorithm. The pseudo-codes of the algorithms are in the following.

Input: the positive semi-definite kernel  $\mathbf{K}$ , the dimension of the wrapped coordinate space

$m$ , and the dimension of the DR data  $d$ .

Output: the DR data matrix  $\mathbf{Y}$  and the corresponding eigenvalues  $\mathbf{ev}$ .

Step 1. Find data-wrapping subspace. If the interpolative method is applied, randomly select  $m$  columns of  $\mathbf{K}$  to make the  $n \times m$  submatrix  $\mathbf{K}_m$ . If the project method is selected, then create an  $n \times m$  random matrix  $\mathbf{R}$  and set  $\mathbf{K}_m = \mathbf{KR}$ .

Step 2. Make DR data-wrap matrix. Make the QR factorization of  $\mathbf{K}_m$  to obtain a matrix  $\mathbf{Q} \in \mathcal{O}_{n,m}$  whose columns form an o.n. basis of  $\text{Col}(\mathbf{K}_m)$ . Then make the wrapped DR data matrix  $\mathbf{Ab} = (\mathbf{K}\mathbf{Q})^T$ .

Step 3. Find DR data. Let the SVD decomposition of  $\mathbf{A}$  be

$$\mathbf{A} = \sum_{i=0}^m \sigma_i \mathbf{u}_i \mathbf{v}_i^T,$$

where  $\sigma_0 > \sigma_1 \geq \dots \geq \sigma_m > 0$ . If  $\mathbf{K}$  is an Isomaps-type kernel, then  $\mathbf{Y}^T = [\mathbf{u}_0, \dots, \mathbf{u}_{d-1}]$ . Otherwise, if  $\mathbf{K}$  is a Dmaps-type kernel, set  $\mathbf{Y}^T = [\mathbf{u}_1, \dots, \mathbf{u}_d]$ , and then divide each  $\mathbf{u}_j$  entry-wise by  $\mathbf{u}_0$ .

**备注226.** The steps 2 and 3 in RAT algorithm are the same as in GAT algorithm. The algorithms can be also used for linear DR of a data set  $\mathcal{X}$ . In the case that the input matrix is the  $D \times n$  data matrix  $\mathbf{X}$ , we use  $\mathbf{K} = \mathbf{X}^T$  in the first step of RAT.

The computational cost of an FAT algorithm (GAT or RAT) is economic. Let the cost be measured by the operational times. Assume that  $\mathbf{G}$  is a general  $n \times c$  matrix, which is not necessary to be positive semi-definite. Let  $C_G$  denote the cost of the multiplication of  $G$  to a  $c$ -vector. The number  $C_G$  is dependent on the sparsity of  $\mathbf{G}$ . If  $\mathbf{G}$  is dense,  $C_G \sim nm$ , and if  $\mathbf{G}$  is sparse,  $C_G \sim \max(c, n)$ . In GAT and I-RAT, Step 1 costs  $O(c)$ , and in P-RAT, Step 1 costs  $C_G$ . In Step 2, the QR decomposition costs  $O(m^2 \times c)$ . The SVD decomposition in Step 3 costs  $m^2 \times n$ . The total cost is  $2 \times m \times C_G + O(m^2(c+n))$ . Hence, the algorithms cost  $O(n \times c)$  for dense matrices and  $O(n)$  for sparse ones, while a standard SVD algorithm requires  $O(c^2 \times n)$ . Hence an FAT algorithm is faster than the corresponding standard algorithm at least by one order. By the way, in FAT algorithms,  $O$  is for a small constant.

## 10.4 Implementation of FAT Algorithms

In this section, we illustrate with various examples the validity and efficiency of the FAT algorithms.

#### 10.4.1 FAT DR of Artificial Surfaces

All computations were carried out on a Mac Pro Desktop computer with 3.06 GHz Intel Core 2 Duo with 4 GB 1067 MHz DDR3 and Mac OS X version 10.5.8 operating system. The algorithms are run by Matlab.

The first four experiments display the results of various FAT algorithms performed on four surfaces: 3D-cluster, Punched sphere, Swiss roll, and S-curve. To carry out our experimentation, 2,000 points are randomly sampled on each surface. The surfaces are reduced to 2-dimensional data on the plane. We select Isomap Kernel in the experiments. The 10-neighborhood is selected for computing the graph metric on the data set.

In the following, we compare the efficiency as well as quality of the algorithms by compiling the CPU time and illustrating the three largest eigen-values and deviations, as well as displaying the resulting DR pictures. Our experimental results on CPU time, three largest eigenvalues, and deviation from the standard Isomap method for the four methods, namely, standard Isomap method, GAT, I-RAT, and P-RAT (applied to the Isomap DR kernel) are compared in Tables 10.1-10.6.

表 10.1: Comparison of standard Isomap algorithm with FAT applied to Isomap DR kernel for Swiss roll: All FAT algorithms employ the 10-neighborhood and Gaussian random projection is used in PRAT. For normalization, all eigenvalues are divided by the first eigenvalue = 936340 of the DR kernel.

| Algorithm | CPU time | eigen 1 | eigen 2 | eigen 3 | deviation |
|-----------|----------|---------|---------|---------|-----------|
| Isomaps   | 2.5285   | 1.0000  | 0.0529  | 0.0093  |           |
| GAT       | 0.1523   | 1.000   | 0.0514  | 0.0072  | 0.0283    |
| IRAT      | 0.1018   | 1.0000  | 0.0529  | 0.0091  | 0.0017    |
| PRAT      | 0.1025   | 1.0000  | 0.0529  | 0.0089  | 0.0014    |

表 10.2: Comparison of standard Isomap algorithm with FAT applied to Isomap DR kernel for S-curve: All FAT algorithms employ the 10-neighborhood and Gaussian random projection is used in PRAT. For normalization, all eigenvalues are divided by the first eigenvalue = 9985 of the DR kernel.

| Algorithm | CPU time | eigen 1 | eigen 2 | eigen 3 | deviation |
|-----------|----------|---------|---------|---------|-----------|
| Isomaps   | 2.7253   | 1.0000  | 0.3097  | 0.0280  |           |
| GAT       | 0.1519   | 1.0000  | 0.3094  | 0.0273  | 0.0014    |
| IRAT      | 0.1036   | 1.0000  | 0.3096  | 0.0274  | 0.0001    |
| PRAT      | 0.1144   | 1.0000  | 0.3096  | 0.0276  | 0.0002    |

表 10.3: Comparison of standard Isomap algorithm with FAT applied to Isomap DR kernel for 3D-cluster: All FAT algorithms employ the 10-neighborhood and Gaussian random projection is used in PRAT. For normalization, all eigenvalues are divided by the first eigenvalue = 12900 of the DR kernel.

| Algorithm | CPU time | eigen 1 | eigen 2 | eigen 3 | deviation |
|-----------|----------|---------|---------|---------|-----------|
| Isomaps   | 2.9455   | 1.0000  | 0.0974  | 0.0053  |           |
| GAT       | 0.1542   | 0.9997  | 0.0792  | 0.0046  | 0.0088    |
| IRAT      | 0.1203   | 1.0000  | 0.0974  | 0.0053  | 0.0000    |
| PRAT      | 0.1287   | 1.0000  | 0.0974  | 0.0053  | 0.0001    |

表 10.4: Comparison of standard Isomap algorithm with FAT applied to Isomap DR kernel for Punched sphere: All FAT algorithms employ the 10- neighborhood and Gaussian random projection is used in PRAT. For normalization, all eigenvalues are divided by the first eigenvalue = 716.63 of the DR kernel.

| Algorithm | CPU time | eigen 1 | eigen 2 | eigen 3 | deviation |
|-----------|----------|---------|---------|---------|-----------|
| Isomaps   | 1.4423   | 1.0000  | 0.9459  | 0.3852  |           |
| GAT       | 0.0954   | 1.0000  | 0.9105  | 0.3497  | 0.0543    |
| IRAT      | 0.0636   | 1.0000  | 0.9455  | 0.3849  | 0.0020    |
| PRAT      | 0.0739   | 1.0000  | 0.9457  | 0.3845  | 0.0010    |

表 10.5: Comparison of standard Isomap algorithm with PRAT applied to Isomap DR kernel for S-curve: All PRAT algorithms employ the 10-neighborhood and random projections of three different types are applied. For normalization, all eigenvalues are divided by the first eigenvalue = 9985 of the DR kernel.

| Algorithm | CPU time | eigen 1 | eigen 2 | eigen 3 | deviation |
|-----------|----------|---------|---------|---------|-----------|
| Isomaps   | 2.7681   | 1.0000  | 0.3097  | 0.0280  |           |
| PRAT-1    | 0.0826   | 1.0000  | 0.3096  | 0.0277  | 0.0004    |
| PRAT-2    | 0.0758   | 1.0000  | 0.3096  | 0.0277  | 0.0001    |
| PRAT-3    | 0.0714   | 1.0000  | 0.3096  | 0.0274  | 0.0002    |

表 10.6: Comparison of standard Isomap algorithm with PRAT applied to Isomap DR kernel for Punched sphere: All PRAT algorithms employ the 10- neighborhood and random projections of three different types are applied. For normalization, all eigenvalues are divided by the first eigenvalue = 725.76 of the DR kernel.

| Algorithm | CPU time | eigen 1 | eigen 2 | eigen 3 | deviation |
|-----------|----------|---------|---------|---------|-----------|
| Isomaps   | 1.3520   | 1.0000  | 0.9819  | 0.3870  |           |
| PRAT-1    | 0.0572   | 0.9998  | 0.9817  | 0.3867  | 0.0005    |
| PRAT-2    | 0.0452   | 0.9997  | 0.9814  | 0.3859  | 0.0016    |
| PRAT-3    | 0.0484   | 0.9998  | 0.9817  | 0.3865  | 0.0005    |



## 第十一章 期末考题

Given the image in Figure 11.1 with missing values (downloadable from the course web), use any manifold learning or sparse representation method, or their combination, to inpaint it (i.e., fill in the values at pixels whose RGB values are *all zeros*). It is OK to refer to existing literature. However, *you have to code all by yourself*. The score is proportional to the PSNR. **If you do not use any manifold learning or sparse representation method, i.e., doing simple interpolation, your score will be zero.**



图 11.1: The test image to be inpainted.

Submit the report by June 20th. Also submit your code and your inpainted image. The teaching assistant will compute the PSNR by comparing your inpainted image with the ground truth.

No plagiarism! Those detected will all be marked zeros!



## 参 考 文 献

- [1] 林宙辰. 机器学习及其应用 2013, available at [http://www.cis.pku.edu.cn/faculty/vision/zlin/Publications/Rank\\_Minimization.pdf](http://www.cis.pku.edu.cn/faculty/vision/zlin/Publications/Rank_Minimization.pdf).
- [2] 白正国、沈一兵、水乃翔、郭孝英. 黎曼几何导论（修订版）. 高等教育出版社, 2004.
- [3] 范金城、梅长林. 数据分析（第二版）. 科学出版社, 2010.
- [4] 孙尚拱. 应用多变量统计分析. 科学出版社, 2011.
- [5] 林宙辰、马毅. 信号与数据处理中的低秩模型——理论、算法与应用. 中国计算机学会通讯, 2015.
- [6] A. Adler, M. Elad, and Y. Hel-Or. Probabilistic subspace clustering via sparse representations. *IEEE Signal Processing Letters*, 2013.
- [7] Heinz H. Bauschke and Patrick L. Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer, 1 edition, 2011.
- [8] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, pp. 183-202, 2009.
- [9] Amir Beck and Marc Teboulle. A linearly convergent algorithm for solving a class of nonconvex/affine feasibility problems. *Fixed-Point Algorithms for Inverse Problems in Science and Engineering, Springer Optimization and Its Applications*, pages 33–48, 2011.
- [10] M. Becka and G. Oksa. On variable blocking factor in a parallel dynamic block-Jacobi SVD algorithm. *Parallel Computing*, pp. 1153-1174, 2003.
- [11] M. Becka, G. Oksa, and M. Vajtersic. Dynamic ordering for a parallel block-Jacobi SVD algorithm. *Parallel Computing*, pp. 243-262, 2002.
- [12] J. Bennett and S. Lanning. The netflix prize. In Proceedings of KDD Cup and Workshop, 2007.
- [13] D.P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2nd edition, 1999.

- [14] D.P. Bertsekas, A. Nedic, and A.E. Ozdaglar. *Convex Analysis and Optimization*. Athena Scientific, 2003.
- [15] C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007.
- [16] William M. Boothby. *An Introduction to Differentiable Manifolds and Riemannian Geometry*. Elsevier, 2003.
- [17] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Jordan M (ed) Foundations and Trends in Machine Learning*, 2011.
- [18] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
- [19] P. A. Businger and G. H. Golub. *Algorithm 358: singular value decomposition of a complex matrix*. Communications of the ACM, pp. 564–565, 1969.
- [20] J. F. Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.
- [21] E. J. Candès. *Compressive sampling*. In Proceedings of the International Congress of Mathematicians, 2006.
- [22] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM*, 58(1):1–37, 2011.
- [23] E. J. Candès and Y. Plan. Matrix completion with noise. *Proceedings of the IEEE*, 98(6):925–936, 2010.
- [24] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.
- [25] E. J. Candès and J. Romberg. *Sparsity and incoherence in compressive sampling*. Inverse Problems, pp. 969–985, 2006.
- [26] E. J. Candès, M. Rudelson, T. Tao, and R. Vershynin. *Error correction via linear programming*. In Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science, 2005.
- [27] E. J. Candès and T. Tao. *Reflections on compressed sensing*. IEEE Information Theory Society Newsletter, pp. 14–17, 2008.

## 参 考 文 献

---

- [28] E. J. Candès and M. Wakin. *An introduction to compressive sampling*. IEEE Signal Processing Magazine, pp. 21-30, 2008.
- [29] Hong Chang, Dit-Yan Yeung, and Yimin Xiong. Super-resolution through neighbor embedding. pages 275–282. In Proceedings of Computer Vision and Pattern Recognition, 2004.
- [30] Y. Chen, H. Xu, C. Caramanis, and S. Sanghavi. Robust matrix completion and corrupted columns. In Proceedings of the 21st International Conference on Machine Learning, 2011.
- [31] B. Cheng, G. Liu, J. Wang, Z. Huang, and S. Yan. Multi-task low-rank affinity pursuit for image segmentation. In Proceedings of International Conference on Computer Vision, 2011.
- [32] B. Cheng, J. Yang, S. Yan, Y. Fu, and T.S. Huang. Learning with  $\ell_1$ -graph for image analysis. *IEEE Transactions on Image Processing*, 19(4), 2010.
- [33] Hong Cheng, Zicheng Liu, Lu Yang, and Xuewen Chen. Sparse representation and learning in visual recognition: Theory and applications. *Signal Processing*, 93(6):1408–1425, 2013.
- [34] Fan R.K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.
- [35] J. Costeira and T. Kanade. A multibody factorization method for independently moving objects. *International Journal of Computer Vision*, 29(3):159–179, 1998.
- [36] J. W. Demmel and W. Kahan. *Accurate singular values of bidiagonal matrices*. SIAM Journal on Scientific Computing, pp. 873-912, 1990.
- [37] W. Deng, W. Yin, and Y. Zhang. Group sparse optimization by alternating direction method. *TR11-06, Department of Computational and Applied Mathematics, Rice University*, 2011.
- [38] I. Dhillon, J. Demmel, and M. Gu. *Efficient computation of the singular value decomposition with applications to least squares problems*. LBL Report #36201, 1994.
- [39] D. Donoho. De-noising by soft-thresholding. *IEEE Transaction on Information Theory*, 41(3):613–627, 1995.

- [40] Z. Drmac. *A posteriori computation of the singular vectors in a preconditioned Jacobi SVD algorithm.* IMA Journal of Numerical Analysis, pp. 191-213, 1999.
- [41] Z. Drmac and K. Veselic. *New fast and accurate Jacobi SVD algorithm. I.* SIAM Journal on Matrix Analysis and Applications, 2007.
- [42] Z. Drmac and K. Veselic. *New fast and accurate Jacobi SVD algorithm: II.* SIAM J. Matrix Anal. Appl., pp. 1343-1362, 2008.
- [43] A. Edelman, T. Arias, and S.T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.
- [44] Michael Elad. *Sparse and Redundant Representations - From Theory to Applications in Signal and Image Processing.* Springer, 1 edition, 2010.
- [45] E. Elhamifar and R. Vidal. Sparse subspace clustering. In *IEEE International Conference on Computer Vision and Pattern Recognition*. IEEE, 2009.
- [46] P. Favaro, R. Vidal, and A. Ravichandran. A closed form solution to robust subspace estimation and clustering. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2011.
- [47] M. Fazel. *Matrix Rank Minimization with Applications.* Ph.D. Thesis, 2002.
- [48] Jiashi Feng, Zhouchen Lin, Huan Xu, and Shuicheng Yan. Robust subspace segmentation with block-diagonal prior. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2014.
- [49] Simon Foucart and Holger Rauhut. *A Mathematical Introduction to Compressive Sensing.* Birkhäuser, 1 edition, 2013.
- [50] Yifan Fu, Junbin Gao, David Tien, and Zhouchen Lin. Tensor lrr based subspace clustering. In Proceedings of International Joint Conference on Neural Networks, 2014.
- [51] A. Ganesh, Z. Lin, J. Wright, L. Wu, M. Chen, and Y. Ma. Fast algorithms for recovering a corrupted low-rank matrix. pages 213–216. International Workshop on Computational Advances in Multi-Sensor Adaptive Processing, 2009.

- [52] H. Gao, J.-F. Cai, Z. Shen, and H. Zhao. Robust principal component analysis-based four-dimensional computed tomography. *Physics in Medicine and Biology*, 2011.
- [53] Navin Goel, George Bebis, and Ara Nefian. Face recognition experiments with random projection. *Proc. SPIE 5779*, 2004.
- [54] G. H. Golub and C. F. Van Loan. *Matrix computations (3rd ed.)*. Johns Hopkins University Press, 1996.
- [55] D. Gross. *Recovering low-rank matrices from few coefficients in any basis*. Preprint, 2009.
- [56] M. Gu and S. C. Eisenstat. *A divide-and-conquer algorithm for the bidiagonal SVD*. SIAM Journal on Matrix Analysis and Applications Volume 16, Number 1, pp. 79-92, 1995.
- [57] Shuhang Gu, Lei Zhang, Wangmeng Zuo, and Xiangchu Feng. Weighted nuclear norm minimization with application to image denoising. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2014.
- [58] Jihun Ham, Daniel D. Lee, Sebastian Mika, and Bernhard Schölkopf. A kernel view of the dimensionality reduction of manifolds. In Proceedings of the 21st International Conference on Machine Learning, 2004.
- [59] B. He, M. Tao, and X. Yuan. Alternating direction method with Gaussian back substitution for separable convex programming. *SIAM J. Optimization*, 22(2):313–340, 2012.
- [60] B. He and X. Yuan. Linearized alternating direction method with Gaussian back substitution for separable convex programming. *Numerical Algebra, Control and Optimization*, 3(2):247–260, 2013.
- [61] A. S. Householder. *Unitary triangularization of a nonsymmetric matrix*. Journal of the ACM, pp. 339-342, 1958.
- [62] Han Hu, Zhouchen Lin, Jianjiang Feng, and Jie Zhou. Smooth representation clustering. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2014.

- [63] Yao Hu, Debing Zhang, Jieping Ye, Xuelong Li, and Xiaofei He. Fast and accurate matrix completion via truncated nuclear norm regularization. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 35(9):2117–2130, 2013.
- [64] L. Jacob, G. Obozinski, and J.P. Vert. Group Lasso with overlap and graph Lasso. *ICML*, pages 433–440, 2009.
- [65] D. James and V. Kresimir. *Jacobi's method is more accurate than QR*. SIAM Journal on Matrix Analysis and Applications, pp. 1204-1245, 1992.
- [66] I.-H. Jhuo, D. Liu, D. T. Lee, and S.-F. Chang. Robust visual domain adaptation with low-rank reconstruction. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2012.
- [67] H. Ji, C. Liu, Z. Shen, and Y. Xu. Robust video denoising using low rank matrix completion. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010.
- [68] I.T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer-Verlag, Berlin, 1986.
- [69] A. Kerr, D. Campbell, and M. Richards. *QR decomposition on GPUs*. In Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units, 2009.
- [70] R. H. Keshavan, A. Montanari, and S. Oh. *Matrix completion from noisy entries*. Priprint, 2009.
- [71] S. Lafon. *Diffusion maps and geometric harmonics*. Ph.D. thesis, Yale University, 2004.
- [72] Stéphane Lafon and Ann B. Lee. Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 2006.
- [73] S. Lahabar and P. J. Narayanan. *Singular value decomposition on GPU using CUDA*. IEEE International Symposium on Parallel&Distributed Processing, 2009.
- [74] C. Lang, G. Liu, J. Yu, and S. Yan. Saliency detection by multitask sparsity pursuit. *IEEE Transactions on Image Processing*, 21(3):1327–1338, 2012.
- [75] R. M. Larsen. <http://soi.stanford.edu/rmunk/PROPACK/>. 2004.

- [76] Erik G. Larsson and Yngve Selen. Linear regression with a sparse parameter vector. *IEEE Transactions on Signal Processing*, 55(2):451–460, 2007.
- [77] Chun-Guang Li, Zhouchen Lin, and Jun Guo. Bases sorting: Generalizing the concept of frequency for over-complete dictionaries. *Neurocomputing*, 115:192–200, 2013.
- [78] Z. Lin, M. Chen, L. Wu, and Y. Ma. *The augmented Lagrange multiplier method for exact recovery of a corrupted low-rank matrix*. Preprint, 2009.
- [79] Z. Lin, A. Ganesh, J. Wright, L. Wu, M. Chen, and Y. Ma. Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix. *UIUC Technical Report UILU-ENG-09-2214*, 2009.
- [80] Z. Lin, R. Liu, and Z. Su. Linearized alternating direction method with adaptive penalty for low-rank representation. pages 612–620. In Proceedings of Advances in Neural Information Processing Systems (NIPS), 2011.
- [81] Zhouchen Lin, Risheng Liu, and Huan Li. Linearized alternating direction method with parallel splitting and adaptive penalty for separable convex programs in machine learning. *Machine Learning*, page to appear.
- [82] G. Liu, Z. Lin, and Y. Yu. Robust subspace segmentation by low-rank representation. In *Proceedings of the 26th International Conference on Machine Learning, Haifa, Israel*, 2010.
- [83] G. Liu, H. Xu, and S. Yan. Exact subspace segmentation and outlier detection by low-rank representation. pages 703–711. Proceedings of International Conference on Artificial Intelligence and Statistics (AISTAT), 2012.
- [84] G. Liu, H. Xu, and S. Yan. Exact subspace segmentation and outlier detection by low-rank representation. In Proceedings of International Conference on Artificial Intelligence and Statistics, 2012.
- [85] G. Liu and S. Yan. Latent low-rank representation for subspace segmentation and feature extraction. In *ICCV*, 2011.
- [86] Guangcan Liu, Zhouchen Lin, Shuicheng Yan, Ju Sun, and Yi Ma. Robust recovery of subspace structures by low-rank representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):171–184, 2013.

- [87] Guangcan Liu, Zhouchen Lin, Shuicheng Yan, Ju Sun, and Yi Ma. Robust recovery of subspace structures by low-rank representation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 35(1):171–184, 2013.
- [88] Ji Liu, Przem Musalski, Peter Wonka, and Jieping Ye. Tensor completion for estimating missing values in visual data. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 2013.
- [89] R. Liu, Z. Lin, and Z. Su. Linearized alternating direction method with parallel splitting and adaptive penalty for separable convex programs in machine learning. In Proceedings of Asian Conference on Machine Learning (ACML), 2013.
- [90] Risheng Liu, Zhouchen Lin, Zhixun Su, and Junbin Gao. Linear time principal component pursuit and its extensions using  $\ell_1$  filtering. *Neurocomputing*, 2014.
- [91] Risheng Liu, Zhouchen Lin, Fernando De La Torre, and Zhixun Su. Fixed-rank representation for unsupervised visual learning. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2012.
- [92] Z. Liu and L. Vandenberghe. *Semidefinite programming methods for system realization and identification*. In Proceedings of IEEE Conference on Decision and Control, 2009.
- [93] Can-Yi Lu, Jiashi Feng, Zhouchen Lin, and Shuicheng Yan. Correlation adaptive subspace segmentation by trace lasso. In Proceedings of International Conference on Computer Vision, 2013.
- [94] Can-Yi Lu, Hai Min, Zhong-Qiu Zhao, Lin Zhu, De-Shuang Huang, and Shuicheng Yan. Robust and efficient subspace segmentation via least squares regression. In Proceedings of European Conference on Computer Vision, 2012.
- [95] Canyi Lu, Zhouchen Lin, and Shuicheng Yan. Smoothed low rank and sparse matrix recovery by iteratively reweighted least squared minimization. *IEEE Trans. Image Processing*, 2015.
- [96] Canyi Lu, Changbo Zhu, Chunyan Xu, Shuicheng Yan, and Zhouchen Lin. Generalized singular value thresholding. In Proceedings of AAAI Conference on Artificial Intelligence, 2015.
- [97] Qin Lyu, Zhouchen Lin, Yiyuan She, and Chao Zhang. A comparison of typical  $\ell_p$  minimization algorithms. *Neurocomputing*, 119:413–424, 2013.

- [98] S. Ma, D. Goldfarb, and L. Chen. Fixed point and bregman iterative methods for matrix rank minimization. *Mathematical Programming*, 128(1-2):321–359, 2011.
- [99] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer, 1 edition, 2004.
- [100] L. Meier, S. Van De Geer, and P. Bühlmann. The group Lasso for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(1):53–71, 2008.
- [101] R. Meka, P. Jain, and I. S. Dhillon. *Matrix completion from power-law distributed samples*. In Proceedings of the Conference on Neural Information Processing Systems (NIPS), 2009.
- [102] J. Meng, W. Yin, E. Houssain, and Z. Han. *Collaborative spectrum sensing from sparse observations using matrix completion for cognitive radio networks*. In Proceedings of International Conference on Acoustics, Speech, and Signal Processing, 2010.
- [103] Y. Ming and Q. Ruan. Robust sparse bounding sphere for 3d face recognition. *Image and Vision Computing*, 2012.
- [104] L. Mukherjee, V. Singh, J. Xu, and M. D. Collins. Analyzing the subspace structure of related images: Concurrent segmentation of image sets. In Proceedings of European Conference on Computer Vision, 2012.
- [105] S. Negahban, P. Ravikumar, M. Wainwright, and B. Yu. *A unified framework for high-dimensional analysis of  $m$ -estimators with decomposable regularizers*. In Proceedings of the Conference on Neural Information Processing Systems (NIPS), 2009.
- [106] A. Nemirovski. *Efficient Methods in Convex Programming*. Preprint, 1996.
- [107] Y. Nesterov. A method of solving a convex programming problem with convergence rate  $O(1/k^2)$ . *Soviet Mathematics Doklady*, 27(2):372–376, 1983.
- [108] NVIDIA. *CUDA CUBLAS library*. NVIDIA Corporation, 2009.
- [109] NVIDIA. *NVIDIA CUDA programming guide*. 2009.
- [110] C. C. Paige and M. A. Saunders. *LSQR: an algorithm for sparse linear equations and sparse least squares*. ACM Transactions on Mathematical Software, pp. 43-71, 1982.

- [111] Y. Panagakis and C. Kotropoulos. Automatic music tagging by low-rank representation. In Proceedings of International Conference on Acoustics, Speech, and Signal Processing, 2012.
- [112] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6):559–572.
- [113] Y. Peng, A. Ganesh, J. Wright, W. Xu, and Y. Ma. RASL: Robust alignment by sparse and low-rank decomposition for linearly correlated images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012.
- [114] Jianjun Qian, Jian Yang, Fanlong Zhang, and Zhouchen Lin. Robust low-rank regularized regression for face recognition with occlusion. In Proceedings of CVPR 2014 Biometrics Workshop, 2014.
- [115] Xiang Ren and Zhouchen Lin. Linearized alternating direction method with adaptive penalty and warm starts for fast solving transform invariant low-rank textures. *International Journal of Computer Vision*, 104:1–14.
- [116] J. Rennie and N. Srebro. *Fast maximum margin matrix factorization for collaborative prediction*. International Conference on Machine Learning (ICML), 2005.
- [117] R. Rockafellar. *Convex Analysis*. Princeton University Press, 1 edition, 1970.
- [118] S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, December 2000.
- [119] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 22(8):888–905, August 2000.
- [120] A. Singer and M. Cucuringu. *Uniqueness of low-rank matrix completion by rigidity theory*. SIAM Journal on Matrix Analysis and Applications, pp. 1621–1641, 2010.
- [121] A. Subramanian, P. Tamayo, V.K. Mootha, S. Mukherjee, and et al. Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences*, 102(43):267–288, 2005.
- [122] Huachun Tan, Jianshuai Feng, Guangdong Feng, Wuhong Wang, and Yu-Jin Zhang. Traffic volume data outlier recovery via tensor model. *Mathematical Problems in Engineering*, 2013.

## 参 考 文 献

---

- [123] J.B. Tenenbaum, V. de Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, December 2000.
- [124] K.C. Toh and S.W. Yun. *An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems*. Accepted by Pacific J. Optimization, 2009.
- [125] Ivana Tošić and Pascal Frossard. Dictionary learning. *IEEE Signal Processing Magazine*, 28(2):27–38, 2011.
- [126] R. Tron and R. Vidal. A benchmark for the comparison of 3d motion segmentation algorithms. In Proceedings of International Conference on Computer Vision and Pattern Recognition, 2007.
- [127] M K-S Tso. Reduced-rank regression and canonical analysis. *Journal of the Royal Statistical Society, Series B (Methodological)*, 43(2):183–189, 1981.
- [128] V. Chandrasekaran V, S. Sanghavi, P. Parrilo, and A. Willsky. Sparse and low-rank matrix decompositions. pages 962–967. Allerton’09 Proceedings of the 47th Annual Allerton Conference on Communication, Control, and Computing, 2009.
- [129] M.J. van de Vijver, Y.D. He, L.J. van’t Veer, H. Dai, and et al. A gene-expression signature as a predictor of survival in breast cancer. *The New England Journal of Medicine*, 347(25):1999–2009, 2002.
- [130] L.J.P. van der Maaten, E. O. Postma, and H. J. van den Herik. Dimensionality reduction: A comparative review, 2008.
- [131] V.N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 2 edition, 1999.
- [132] Rene Vidal, Yi Ma, and S. S. Sastry. *Generalized Principal Component Analysis*. Springer, 2015.
- [133] Jianzhong Wang. *Geometric Structure of High-Dimensional Data and Dimensionality Reduction*. Springer, 2012.
- [134] Joseph Wang, Venkatesh Saligrama, and David Castanon. Structural similarity and distance in learning. Annual Allerton Conf. Communication, Control and Computing, 2011.
- [135] G. A. Watson. Characterization of the subdifferential of some matrix norms. *Linear Algebra and its Applications*, 170:33–45, 1992.

- [136] Siming Wei and Zhouchen Lin. Analysis and improvement of low rank representation for subspace segmentation. Technical report, arXiv: 1107.1561., 2010.
- [137] Zaiwen Wen, Wotao Yin, and Yin Zhang. Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm. *Math. Program. Comput.*, 2012.
- [138] J. Wright, A. Ganesh, K. Min, and Y. Ma. Compressive principal component pursuit. pages 1276–1280. International Symposium on Information Theory (ISIT), 2012.
- [139] J. Wright, A. Ganesh, S. Rao, Y. Peng, and Y. Ma. Robust principal component analysis: exact recovery of corrupted low-rank matrices via convex optimization. pages 2080–2088. In Proceedings of the Conference on Neural Information Processing Systems (NIPS), 2009.
- [140] L. Wu, A. Ganesh, B. Shi, Y. Matsushita, Y. Wang, and Y. Ma. Robust photometric stereo via low-rank matrix completion and recovery. In Proceedings of Asian Conference on Computer Vision, 2010.
- [141] L. Xiao, J. Sun, and S.P. Boyd. A duality view of spectral methods for dimensionality reduction. In Proceedings of International Conference on Machine Learning, 2006.
- [142] Y. Xu, W. Yin, and Z. Wen. An alternating direction algorithm for matrix completion with nonnegative factors. *CAAM Technical Report TR11-03*, 2011.
- [143] Allen Yang, Arvind Ganesh, Shankar Sastry, and Yi Ma. Fast l1-minimization algorithms and an application in robust face recognition: A review. Technical Report UCB/EECS-2010-13, EECS Department, University of California, Berkeley, Feb 2010.
- [144] J. Yang and X. Yuan. Linearized augmented Lagrangian and alternating direction methods for nuclear norm minimization. *Mathematics of Computation*, 82(281):301–329, 2013.
- [145] Jianchao Yang, Zhaowen Wang, Zhe Lin, Scott Cohen, and Thomas S. Huang. Coupled dictionary training for image super-resolution. *IEEE Transactions on Image Processing*, 21(8):3467–3478, August 2012.

- [146] Li Yang, Yang Lin, Zhouchen Lin, and Hongbin Zha. Low rank global geometric consistency for partial-duplicate image search. In Proceedings of International Conference on Pattern Recognition, 2014.
- [147] Stella X. Yu and Jianbo Shi. Multiclass spectral clustering. pages 313–319. In Proceedings of International Conference on Computer Vision, 2003.
- [148] Yao-Liang Yu and Dale Schuurmans. Rank/norm regularization with closed-form solutions: Application to subspace clustering. In Proceedings of Uncertainty in Artificial Intelligence, 2011.
- [149] Hongyang Zhang, Zhouchen Lin, and Chao Zhang. A counterexample for the validity of using nuclear norm as a convex surrogate of rank. In Proceedings of European Conference on Machine Learning, 2013.
- [150] Hongyang Zhang, Zhouchen Lin, Chao Zhang, and Edward Chang. Exact recoverability of robust pca via outlier pursuit with tight recovery bounds. In Proceedings of AAAI Conference on Artificial Intelligence, 2015.
- [151] Hongyang Zhang, Zhouchen Lin, Chao Zhang, and Junbin Gao. Robust latent low rank representation for subspace clustering. *Neurocomputing*, 2014.
- [152] Hongyang Zhang, Zhouchen Lin, Chao Zhang, and Junbin Gao. Relation among some low rank subspace recovery models. *Neural Computation*, under review.
- [153] S. Zhang. *One sided Jacobi method on CUDA for SVD*. Application Research of computers.
- [154] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja. Low-rank sparse learning for robust visual tracking. In Proceedings of European Conference on Computer Vision, 2012.
- [155] Xin Zhang, Zhouchen Lin, Fuchun Sun, and Yi Ma. Rectification of optical characters as transform invariant low-rank textures. In Proceedings of International Conference on Document Analysis and Recognition, 2013.
- [156] Xin Zhang, Zhouchen Lin, Fuchun Sun, and Yi Ma. Transform invariant text extraction. *The Visual Computer*, 2014.
- [157] Z. Zhang, A. Ganesh, X. Liang, and Y. Ma. TILT: Transform invariant low-rank textures. *International Journal of Computer Vision*, 99(1):1–24, 2012.

- [158] Z. Zhang, X. Liang, and Y. Ma. Unwrapping low-rank textures on generalized cylindrical surfaces. In Proceedings of International Conference on Computer Vision (ICCV), 2011.
- [159] Z. Zhang, Y. Matsushita, and Y. Ma. Camera calibration with lens distortion from low-rank textures. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011.
- [160] Z.Y. Zhang and H.Y. Zha. Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. *SIAM J. Sci. Comput.*, 1(26):313–338, 2004.
- [161] Deli Zhao. Formulating LLE using alignment technique. *Pattern Recognition*, 11(39):2233–2235, 2006.
- [162] Deli Zhao, Zhouchen Lin, and Xiaou Tang. Laplacian PCA and its applications. In Proceedings of International Conference on Computer Vision, 2007.
- [163] B. B. Zhou and R. P. Brent. *A parallel ring ordering algorithm for efficient one-sided Jacobi SVD computations*. Journal of Parallel and Distributed Computing, pp. 1-10, 1997.
- [164] D. Zhou, J. Huang, and B. Schölkopf. Learning from labeled and unlabeled data on a directed graph. In Proceedings of International Conference on Machine Learning, 2005.
- [165] Z. Zhou, X. Li, J. Wright, E. J. Candès, and Y. Ma. Stable principal component pursuit. pages 1518–1522. International Symposium on Information Theory (ISIT), 2010.
- [166] G. Zhu, S. Yan, and Y. Ma. Image tag refinement towards low-rank, content-tag prior and error sparsity. In Proceedings of ACM Multimedia, 2010.
- [167] Liansheng Zhuang, Haoyuan Gao, Zhouchen Lin, Yi Ma, Xin Zhang, and Nenghai Yu. Non-negative low rank and sparse graph for semi-supervised learning. In Proceedings of Computer Vision and Pattern Recognition, 2012.
- [168] W. Zuo and Z. Lin. A generalized accelerated proximal gradient approach for total-variation-based image restoration. *IEEE Transactions on Image Processing*, 20(10):2748–2759, 2011.
- [169] A. Zymnis, S. Boyd, and E. J. Candès. *Compressed sensing with quantized measurements*. Signal Processing Letters 17(3), pp. 149-152, 2010.