基于正交迭代的增量 LLE 算法

朱明旱[1,2], 罗大庸[1], 易励群[3], 王一军[1]
(1. 中南大学信息科学与工程学院, 湖南长沙 410083; 2. 湖南文理学院电气与信息工程学院, 湖南常德 415000;
3. 湖南文理学院现代教育技术中心, 湖南常德 415000)

摘 要: LLE(Locally Linear Embedding)算法是一种较好的流形学习算法,但它只能以批处理的方式进行.只要有新的样本加入,就必须重作该算法的全部内容,而原处理结果被全部丢弃.本文提出了一种基于正交迭代的增量 LLE 算法,能有效地利用前面的处理结果,实现增量处理.实验表明该算法是有效的.

关键词: 局部线性嵌入; 流形学习; 正交迭代; 增量
中图分类号: TP391.4 文献标识码: A 文章编号: 0372-2112 (2009) 01-0132-05

Incremental Locally Linear Embedding Algorithm Based on Orthogonal Iteration Method

ZHU Ming-han[1,2], LUO Da-yong[1], YI Li-qun[3], WANG Yi-jun[1]
(1. *College of Information Science and Engineering, Central South University, Changsha, Hunan 410083, China*)
2. *College of Communication and Electric Engineering, Hunan University of Arts and Science, Changde, Hunan 415000, China*)
3. *Modern Education Technology Center, Hunan University of Arts and Science, Changde, Hunan 415000, China*)

Abstract: Locally Linear Embedding(LLE) is a sort of powerful manifold learning algorithm. However, LLE is a batch method. If only one new sample arrives, the whole algorithm must run repeatedly and all the former computational results are discarded. In this paper, an incremental locally linear embedding algorithm based on orthogonal iteration method is proposed, which can take advantage of former computational results effectively to process the increasing data sets. Experimental results show the effectiveness of the proposed algorithm.

Key words: locally linear embedding(LLE); manifold learning; orthogonal iteration; increment

1 引言

流形学习是一种非监督学习方法,它能够提取出高维数据的内在流形结构,其中最具代表性的有等距映射(Isomap)[1]、局部线性嵌入(LLE)[2]、拉普拉斯特征映射(Laplacian Eigenmap)[3]这三种.近年来,这些方法在医学数据处理、人脸图像处理、数据挖掘等方面都有较好的应用.但是,它们不像 PCA(Principal Component Analysis)和 LDA(Linear Discriminant Analysis)那样,能从训练集中得到适用于待测样本的投影向量,只能以批处理的方式进行,不具有增量处理能力.因此,为了求出待测样本的投影值,必须将待测样本加入到原样本集中,重做该算法的全部内容,这给实际处理带来了极大的不便.对此,学者们提出了一系列的解决方法.

针对等距映射算法,M. H. Law 等人[4]提出了一种增量的等距映射算法.针对拉普拉斯特征映射算法,Xiao He 等人[5]提出了保局投影算法(Locality Preserving Projections),一种线性化的拉普拉斯映射算法.针对 LLE, Saul 等人[6]首先提出了一种线性化的 LLE 算法.为了得到待测样本 $\boldsymbol{x}_{N+1}$ 的投影值,该算法先在训练集中找出它的 k 域邻点,然后求出用这 k 个样本重建它的权值 $w_{(N+1)j}(1\leqslant j\leqslant k)$,最后待测样本的投影 $\boldsymbol{y}_{N+1}$,就用这 k 个样本在流形空间里的投影值 $\{\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_k\}$,加权而得到 $\boldsymbol{y}_{N+1}=\sum_{j=1}^{k} w_{(N+1)j}\boldsymbol{y}_j$.这种方法是一种近似的处理方法,它假设训练样本的投影值不会因新样本的加入而发生改变.实际上,在 LLE 的处理中,当有新样本加入时,由于训练集中某些样本的 k 域邻点会发生改变,因此它的投影值也会发生改变.此外,还有一些其它的线性化 LLE 算法,基本思想都与此差不多. Olga Kouropteva 等人[7]提出了一种增量 LLE 算法,他们假设新的样本加入后,新代价矩阵 $\boldsymbol{M}_{\text{new}}$ 与原代价矩阵 $\boldsymbol{M}$ 的前 d 个最小特征值近似相等,通过最小化 $(\boldsymbol{Y}_{\text{new}}\boldsymbol{M}_{\text{new}}\boldsymbol{Y}_{\text{new}}^{\text{T}} - diag(\lambda_1, \lambda_2, \cdots, \lambda_d))$,求出所有样本的投影值,该算

收稿日期: 2007-12-25; 修回日期: 2008-05-08
基金项目: 湖南省教育厅科研项目(No. 08C606)

法在求新样本投影值的同时, 也重新计算了原样本的投影值. 但是, 该方法有两个缺点: ①牵涉到最优化问题, 处理不便. LLE 本身是将最优化问题转化为求解矩阵特征向量的问题, 而这种方法实际上又重新回到了最优化问题; ②由于 $\boldsymbol{M}_{\text{new}}$的特征值没有更新, 随着新增样本数目的增加, $\boldsymbol{M}_{\text{new}}$与 $\boldsymbol{M}$ 的前 d 个最小的特征值间的差值会越来越大, 从而误差会越来越大. 最近, 曾宪华[8]等人提出了一种动态增殖流形学习算法, 先用 LLE 计算各子集的流形, 再将各子集的流形整合, 得到整体流形结构, 这是一种分批处理的方法.

本文对 LLE 算法进行了改进, 提出一种基于正交迭代的增量 LLE 算法(简记为 OILLE). 该算法有效地利用了原处理结果, 避免了新样本加入时的重复运算, 并将求投影坐标时, 所需的对高阶矩阵的分解转化为对低阶矩阵的分解, 大大降低了数据的运算量, 提高了处理速度, 实现了 LLE 的增量处理. 在 Rendered face、Swiss roll、S-curve 数据库上验证了该算法的有效性.

2　LLE 算法概述

LLE 算法在将高维数据集 $\boldsymbol{X}=\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\}(\boldsymbol{x}_i \in R^D)$ 投影为低维数据集 $\boldsymbol{Y}=\{\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_N\}(\boldsymbol{y}_i \in R^d)$ 时, 通常分三步来完成:

(1) 计算数据集中每个样本 $\boldsymbol{x}_i$ 的 k 域邻点$\{\boldsymbol{x}_{i(1)}, \boldsymbol{x}_{i(2)}, \cdots, \boldsymbol{x}_{i(k)}\}$.

(2) 通过最小化式(1)的目标函数, 计算重建每个样本点 $\boldsymbol{x}_i$ 的权值w_{ij}

$$\varepsilon(\boldsymbol{W})=\sum_{i=1}^{N}\left|\boldsymbol{x}_i-\sum_{j=1}^{k}w_{ij}\boldsymbol{x}_j\right|^2 \tag{1}$$

$(\sum_{j=1}^{k}w_{ij}=1$, 如果 $\boldsymbol{x}_j \notin\{\boldsymbol{x}_{i(1)}, \boldsymbol{x}_{i(2)}, \cdots, \boldsymbol{x}_{i(k)}\}$, 则 $w_{ij}=0)$.

(3) 根据权值 w_{ij}, 最小化式(2)的目标函数, 求得 $\boldsymbol{x}_i$ 的 d 维投影向量$\boldsymbol{y}_i$.

$$\varepsilon(\boldsymbol{Y})=\sum_{i=1}^{N}\left|\boldsymbol{y}_i-\sum_{j=1}^{k}w_{ij}\boldsymbol{y}_j\right|^2 \tag{2}$$

$$\left(\frac{1}{N}\boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}}=\boldsymbol{I}, \sum_{i=1}^{N}\boldsymbol{y}_i=\boldsymbol{0}\right)$$

根据 Rayleighr-Ritz 理论, 对称稀疏代价矩阵 $\boldsymbol{M}=(\boldsymbol{I}-\boldsymbol{W})^{\mathrm{T}}(\boldsymbol{I}-\boldsymbol{W})$ 的 d 个最小非零特征值所对应的特征向量就是最小化式(2)的解.

由于 LLE 是一种批处理算法, 若有新的样本加入, 必须重做该算法的全部内容. 实际上, 新样本加入时, 由于许多样本的 k 域邻点根本没有发生改变, 它们的权值也就没变, 因此无需重新计算这些点的权值.

3　正交迭代 LLE

本文提出的基于正交迭代的增量 LLE 算法, 分两步来完成新样本加入后投影值的计算: ①更新代价矩阵, 在这个步骤中, 我们避免了一些重复的权值计算; ②利用原样本的投影结果, 计算所有样本的投影值. 在这个步骤中, 我们将高阶矩阵的分解转化为对低阶矩阵的分解.

3.1　更新代价矩阵

设原样本集为$\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\}$, 权值矩阵为 $\boldsymbol{W}$, 样本 $\boldsymbol{x}_i$ 的 k 域邻点为$\{\boldsymbol{x}_{i(1)}, \boldsymbol{x}_{i(2)}, \cdots, \boldsymbol{x}_{i(k)}\}$. $\boldsymbol{x}_{i(k)}$ 为 $\boldsymbol{x}_i$ 的第k 个最近邻点, 若用 Δ_i^j 表示$\boldsymbol{x}_i$ 和$\boldsymbol{x}_j$ 间的欧氏距离, 则有 $\Delta_i^{i(1)} \leqslant \Delta_i^{i(2)} \leqslant \cdots \leqslant \Delta_i^{i(k)}$.

现有 r 个新样本$\{\boldsymbol{x}_{N+1}, \cdots, \boldsymbol{x}_{N+r}\}$ 加入, 首先, 用 LLE 算法原理, 计算这 r 个样本的 k 域邻点, 并计算出它们的权值$\{w_{(N+1)j}, \cdots, w_{(N+r)j}\}(1 \leqslant j \leqslant k)$. 然后, 找出原样本集中, k 域邻点有改变的样本, 重新计算这些样本的 k 域邻点和权值, 对于 k 域邻点无变化的那些样本, 保持原权值不变. 具体处理如下:

if　$\Delta_i^{N+l} < \Delta_i^{i(k)}$　$(1 \leqslant l \leqslant r)$

　　则 $\boldsymbol{x}_i$ 的k 域邻点有变化

　　在$\{\boldsymbol{x}_{i(1)}, \boldsymbol{x}_{i(2)}, \cdots, \boldsymbol{x}_{i(k-1)}, \boldsymbol{x}_{N+l}, \cdots\}$中, 重新计算 $\boldsymbol{x}_i$ 的k 域邻点和权值

else

　　$\boldsymbol{x}_i$ 的k 域邻点无变化, $\boldsymbol{x}_i$ 的权值不变

最后, 根据所有的权值, 得到新的$(N+r)\times(N+r)$阶的权值矩阵 $\boldsymbol{W}_{\text{new}}$, 从而得到新的代价矩阵:

$$\boldsymbol{M}_{\text{new}}=\left(\boldsymbol{I}-\boldsymbol{W}_{\text{new}}\right)^{\mathrm{T}}\left(\boldsymbol{I}-\boldsymbol{W}_{\text{new}}\right)$$

更新代价矩阵时, 我们只对 k 域邻点有改变的那些样本进行了权值的重新计算, k 域邻点的重新计算也是在一个很小的子集$\{\boldsymbol{x}_{i(1)}, \boldsymbol{x}_{i(2)}, \cdots, \boldsymbol{x}_{i(k-1)}, \boldsymbol{x}_{N+l}, \cdots\}$上进行的. 与重做 LLE, 在整个数据集上重新计算所有样本的域邻点和权值相比, 大大降低了运算量, 提高了处理速度. 特别是在原样本较多, 而新加入样本相对较少时, 更显本算法的优越性.

3.2　求投影值

设新代价矩阵 $\boldsymbol{M}_{\text{new}}$的特征值为:

$\lambda^{\text{new}}=\{\lambda_1^{\text{new}}, \lambda_2^{\text{new}}, \cdots, \lambda_{N+r}^{\text{new}}\}$, $(0 \equiv \lambda_1^{\text{new}} \leqslant \{\lambda_2^{\text{new}}, \cdots, \leqslant \lambda_{N+r}^{\text{new}})$ 对应的特征向量矩阵为 $\boldsymbol{Z}=\{z_1, z_2, \cdots, z_{N+r}\}$, 根据矩阵理论有

$$\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{M}_{\text{new}}\boldsymbol{Z}=diag(\lambda_1^{\text{new}}, \lambda_2^{\text{new}}, \cdots, \lambda_{N+r}^{\text{new}}) \tag{3}$$

对式(3)进行如下变换

$$\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{M}_{\text{new}}\boldsymbol{Z}+\varepsilon\boldsymbol{I}=diag(\lambda_1^{\text{new}}, \cdots, \lambda_{N+r}^{\text{new}})+\varepsilon\boldsymbol{I}$$

$$\boldsymbol{Z}^{\mathrm{T}}(\boldsymbol{M}_{\text{new}}+\varepsilon\boldsymbol{I})\boldsymbol{Z}=diag(\lambda_1^{\text{new}}+\varepsilon, \cdots, \lambda_{N+r}^{\text{new}}+\varepsilon)$$

$$\boldsymbol{Z}^{\mathrm{T}}(\boldsymbol{M}_{\text{new}}+\varepsilon\boldsymbol{I})^{-1}\boldsymbol{Z}=diag((\lambda_1^{\text{new}}+\varepsilon)^{-1}, \cdots, (\lambda_{N+r}^{\text{new}}+\varepsilon)^{-1})$$

$$\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{B}\boldsymbol{Z}=diag((\lambda_1^{\text{new}}+\varepsilon)^{-1}, \cdots, (\lambda_{N+r}^{\text{new}}+\varepsilon)^{-1}) \tag{4}$$

这里 $\boldsymbol{B}=(\boldsymbol{M}_{\text{new}}+\varepsilon\boldsymbol{I})^{-1}$, ε为一很小的正常数.

由式(4)可知, 矩阵 $\boldsymbol{B}$ 的前 $d+1$ 个最大特征值所对应的特征向量, 就是矩阵 $\boldsymbol{M}_{\text{new}}$前 $d+1$ 个最小特征值对应的特征向量. 设原代价矩阵 $\boldsymbol{M}$ 的前$d+1$ 个最小特征值对应的特征向量组成的矩阵为 $\boldsymbol{V}=\{\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_{d+1}\}$, $(\boldsymbol{V}\in R^{N\times(d+1)})$.

我们采用 Ritz 加速的正交迭代算法[9], 求 $\boldsymbol{B}$ 的前$d+1$ 个最大特征值所对应的特征向量.

初始化 $\boldsymbol{Q}_0=\begin{bmatrix}\boldsymbol{V}\\ \boldsymbol{0}\end{bmatrix}$, $\boldsymbol{Q}_0\in R^{(N+r)\times(d+1)}$, $\boldsymbol{0}$是 $r\times(d+1)$阶的零矩阵, 有 $\boldsymbol{Q}_0^{\mathrm{T}}\boldsymbol{Q}_0=\boldsymbol{I}_{(d+1)}$

for $k=1:n$

(1)计算 $\boldsymbol{T}=\boldsymbol{B}\boldsymbol{Q}_{k-1}$, 并对 $\boldsymbol{T}$ 执行 $\boldsymbol{QR}$ 分解, 记为 $\tilde{\boldsymbol{Q}}_k\boldsymbol{R}_k=\boldsymbol{T}$

(2)计算 $\boldsymbol{T}^*=(\tilde{\boldsymbol{Q}}_k(:,1:d+1))^{\mathrm{T}}\boldsymbol{B}\tilde{\boldsymbol{Q}}_k(:,1:d+1)$

(3)对$(d+1)\times(d+1)$阶的矩阵 $\boldsymbol{T}^*$ 执行 Schur 分解, 记为 $\boldsymbol{U}_k^{\mathrm{T}}\boldsymbol{T}^*\boldsymbol{U}_k=\boldsymbol{D}_k$

(4) $\boldsymbol{Q}_k=\tilde{\boldsymbol{Q}}(:,1:d+1)\boldsymbol{U}_k$

end

$\boldsymbol{V}^{\text{new}}=\boldsymbol{Q}_k$

$[\boldsymbol{V}^{\text{new}}(:,2:d+1)]^{\mathrm{T}}$就是$N+r$ 个样本的d 维投影坐标值.

因矩阵 $\boldsymbol{M}_{\text{new}}$奇异, $rank(\boldsymbol{M}_{\text{new}})=N+r-1$, 我们引入正常数 ε, 使矩阵$(\boldsymbol{M}_{\text{new}}+\varepsilon\boldsymbol{I})$的逆存在. 采用 Ritz 加速正交迭代算法, 矩阵 $\boldsymbol{B}$ 的第 i 个最大特征值$(i=(1:d+1)$所对应特征向量的收敛速度为 $O\left(\left|\frac{\lambda_i^{\text{new}}+\varepsilon}{\lambda_{d+2}^{\text{new}}+\varepsilon}\right|^k\right)$, 所以正常数 ε取值很小, 否则会影响收敛速度, 我们取 $\varepsilon=1\text{e}-010$.

通过 Ritz 加速的正交迭代算法, 我们将原 LLE 对$(N+r)\times(N+r)$阶矩阵 $\boldsymbol{M}_{\text{new}}$的分解, 转化为对$(N+r)\times(d+1)$阶矩阵 $\boldsymbol{T}$ 和$(d+1)\times(d+1)$阶矩阵 $\boldsymbol{T}^*$ 的分解, 提高了求投影坐标值的速度(通常 d 很小, 一般 d 取 2 或 3).

4 实验结果及分析

4.1 实验

为了验证该算法的有效性, 我们在 Rendered face[10]、Swiss roll 和 S-curve 这三个数据库上进行了实验. Rendered face 数据库由 698 幅, 分辨率为 64×64, 不同光照和位置的人脸图像组成, 图 1 是 Rendered face 数据库的一些样本. 我们用式(5)来量化 OILLE 算法的误差.

$$error=\sqrt{\frac{1}{N}\sum_{i=1}^{N}\|\boldsymbol{y}_i-\hat{\boldsymbol{y}}_i\|^2/\|\hat{\boldsymbol{y}}_i\|^2} \qquad (5)$$

式中 $\boldsymbol{y}_i$ 表示用 OILLE 算出的投影值, $\hat{\boldsymbol{y}}_i$ 表示用 LLE 算出的投影值. 实验时, 取邻域 $k=11$, 正交迭代次数 $n=5$, $\varepsilon=1\text{e}-010$.

图1 Randered face 数据库中的5个样本

在 Rendered face 数据库上, 先随机选取 100 幅图像作为原始样本集, 再将图像一幅一幅地加入, 用 OILLE 计算投影, 直到第 698 幅. 图 3(a)为 OILLE 的误差曲线图, 表 1 记录了初始样本数 $N=499$, 第 500 幅图像加入时, 运用 LLE 和 OILLE 分别所消耗的时间.

在 Swiss roll 和 S-curve 这两个数据库上, 我们先随机地取 500 个数据点作为原始样本集, 用 LLE 计算它们的投影值. 再将新数据一个接一个地加入, 用 OILLE 计算投影, 直到 2000 个数据点. 图 2 展示了 Swiss roll 数据库上, 用 LLE 和 OILLE 所得的投影的结果(图中"+"表示样本数据, "."表示用 LLE 得到的投影点, "o"表示用 OILLE 得到的投影点). 图 3(b)为在Swiss roll 数据库上, OILLE 的误差曲线图. 图 3(c)为在 S-curve 数据库上, OILLE 的误差曲线图. 表 1 也记录了原始样本数 $N=999$、1499、1999 时, 第 1000、1500、2000 个样本加入时, 运用 LLE 和 OILLE 各自所消耗的时间.

表 1 执行 LLE 与 OILLE 的时间(单位: 秒)

$N+1$	Swiss roll (3 to 2)		S-curve (3 to 2)		Rendered (4096 to 3)	
	LLE	OILLE	LLE	OILLE	LLE	OILLE
500	4.56	0.63	4.53	0.56	12.47	0.59
1000	28.06	3.82	29.80	3.78	N/A	N/A
1500	89.47	11.44	81.23	11.72	N/A	N/A
2000	234.16	34.41	208.94	36.34	N/A	N/A

4.2 结果分析

误差 在 Swiss roll 数据库, LLE 算法和 OILLE 算法的投影结果几乎没有什么差别, 如图 2 所示. OILLE 的量化平均误差, 在 Rendered face 数据库上为 1.660e-005, 在 Swiss roll 数据库上为 2.94e-008, 在 S-curve 数据库上为 2.25e-009, 这些误差均很小, 小于 0.01%, 可见 OILLE 的处理精度完全满足要求.

处理速度 表 1 的数据展示, 在 Rendered face 数据库上, OILLE 的执行时间约为 LLE 的 1/20, 在 Swiss roll 和 S-curve 数据库上, OILLE 的执行时间约为LLE 的1/7. 与批处理的 LLE 算法相比, 改进后的 OILLE 运行时间短, 处理速度快.

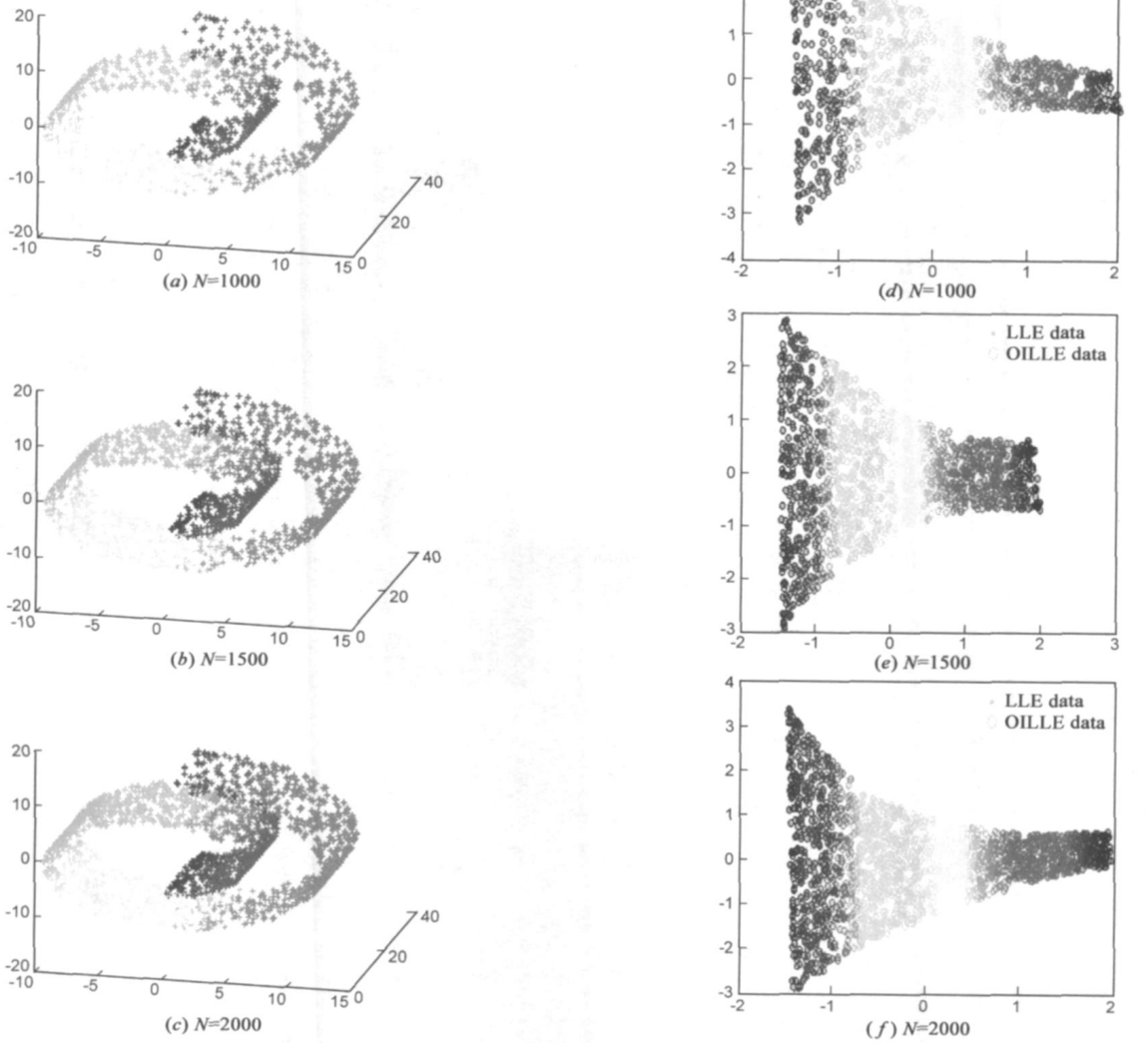

图2　在Swiss roll 数据库上，用OILLE和LLE所得到的投影. 第一列是样本点图,第二列是投影点图. （“+”表示样本点“.”表示用LLE得到的投影点，“o”表示用OILLE得到的投影点）

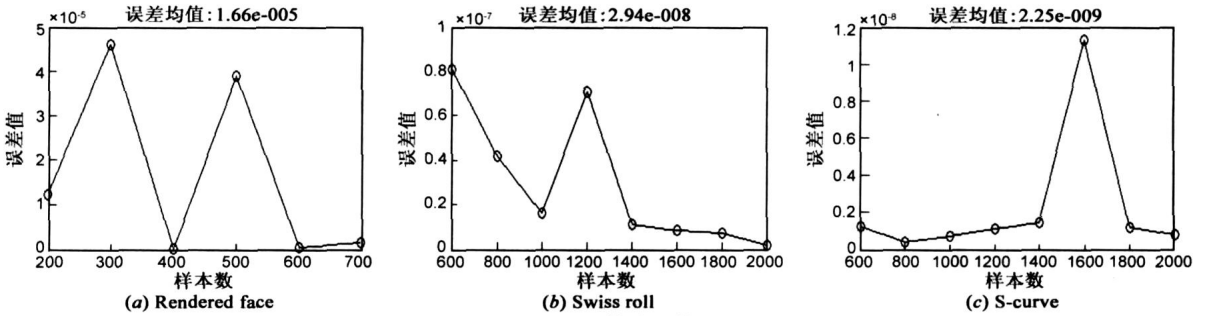

图3　OILLE算法误差

5　结束语

本文的基于正交迭代的增量 LLE 算法, 能够不断地利用前一次的处理结果, 来计算各样本的投影值. 不但避免了重做 LLE 中的重复计算, 而且降低分解矩阵的阶数, 从而提高了计算效率. 该算法是一种具有增量处理能力 LLE 算法, 较好地解决了在新样本不断加入的情况下, 总体流形不断更新的问题.

参考文献:

[1] J B Tenenbaum, V de Silva, et al. A global geometric framework for nonlinear dimensionality reduction[J]. Science, 2000,

290(5500):2319-2323.
[2] S T Roweis, L K Saul. Nonlinear dimensionality reduction by locally linear embedding[J]. Science, 2000, 290(5500): 2323-2326.
[3] M Belkin, P Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering[A]. Neural Information Processing Systems 14[C]. Cambridge, MA: MIT Press, 2002, 585-591.
[4] M H Law, N Zhang, et al. Nonlinear manifold learning for data stream[A]. Proceedings of SIAM Data Mining[C]. Florida: Orlando, 2004. 33-44.
[5] X He, S Yan, et al. Face recognition using Laplacianfaces[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2005, 27(3): 328-340.
[6] L K Saul, S T Roweis. Think globally, fit locally: unsupervised learning of low dimensional manifolds[J]. Journal of Machine Learning Research, 2003, 4(Jun): 119-155.
[7] Olga Kouropteva, Oleg Okun et al. Incremental locally linear embedding algorithm[J]. Pattern Recognition, 2005, 38(10): 1764-1767.
[8] 曾宪华, 罗四维. 动态增殖流形学习算法[J]. 计算机研究与发展, 2007, 44(9): 1462-1468.
Zeng Xianhua, Luo Siwei. A dynamically incremental manifold learning algorithm[J]. Journal of Computer Research and Development, 2007, 44(9): 1462-1468. (in Chinese)
[9] G H Golub, C F Van Loan. Matrix Computations[M]. Baltimore: Johns Hopkins University Press, 1996. 422-424.
[10] Data sets for nonlinear dimensionality reduction[OL] http://isomap.stanford.edu/datasets.html

作者简介:

朱明早 男, 讲师. 1974 年 8 月生于湖南慈利县, 2005 年毕业于中南大学信息科学与工程学院, 获硕士学位, 其后在湖南文理学院电气与信息工程学院任教. 现为中南大学信息科学与工程学院博士生, 主要从事模式识别、计算机视觉方面的研究.
E-mail: zhumh_123@163.com

罗大庸 男, 教授、博士生导师. 1944 年 10 月生于湖南长沙, 主要从事模式识别、计算机视觉、智能控制、信息融合技术等方面的研究工作.
E-mail: dyluo@mail.csu.edu.cn