高维数据特征提取算法的研究及比较*)

林晓立　陈恩红　任皖英
(中国科学技术大学计算机系　合肥230027)

Algorithms of Feature Extraction from High-Dimensional Objects and their Comparisons

LIN Xiao-Li　CHEN En-Hong　REN Wan-Ying
(Department of Computer Science, University of science and Technology of China, Hefei 230027)

Abstract This paper introduces and analyzes several feature extraction algorithms. These algorithms use linear or non-linear feature extraction methods to project high-dimensional objects into lower dimensional space, thus the complexity of the operations upon them, such as clustering, the nearest-neighbor search, visualization and etc can be reduced. The paper also presents some comparative experimental results of these algorithms and analyzes briefly their advantages or shortcomings.

Keywords High-dimensional data, Feature extraction, Dimension reduction

1 引言

近年来随着科技的发展，出现了越来越多的复杂数据，如多媒体数据、蛋白质数据等。直接对这些高维数据进行诸如聚类、最近邻搜索及可视化等操作[4~6]，其代价十分昂贵，需要的运算量往往超出机器的容许范围。特征提取的目的旨在对这些数据进行降维，从而大大降低对它们进行各种操作的计算复杂度。

特征提取一般涉及"距离"问题和"降维"问题。"距离"问题就是已知复杂对象之间的距离信息，要求提取它们的特征信息来代表这些数据；降维问题就是已知高维数据点的k个特征，要求从中提取k'(k'≪k)个特征来代表这些数据。提取出的特征要尽可能保持这些数据点之间的距离，从而保持它们的特性。

一种理想的特征提取算法具有以下的特点[2]：

1. 具有较低的计算复杂度，只需计算高维数据集中部分数据间的实际距离，计算复杂度只能为O(N)或O(NlogN)，不能达到或超过$O(N^2)$。

2. 具有低"stress"值，stress值是衡量距离保持程度的标尺(其定义如下)。某种算法得出的stress值越低，表明对象间的距离保持得越好。

$$\text{stress}=\sqrt{\frac{\sum_{i,j}(d'_{ij}-d_{ij})^2}{\sum_{i,j}d_{ij}^2}}$$

其中d_{ij}是对象间实际距离，d'_{ij}是对象间特征提取后的新距离。

3. 能够提供对新的高维对象进行降维的快速算法，复杂度为O(1)或O(N)。

2 特征提取算法

本节将从算法的特征提取过程、算法计算量及算法特点等方面对当前比较有代表性的几种特征提取算法进行介绍与分析，并基于我们对其中的Cofe算法的理解和实验分析，提出了一种改进方法。

2.1 FastMap算法[2]

1. 特征提取过程

假设已知k维(k较大)空间中的N个点以及它们之间的距离(符合三角不等式)，设法把它们投影到相互正交的k'维空间中(k'≪k)，从而得出数据对象的k'维特征。

该算法的核心在于如何选定投影参考线或参考面，以及如何把数据对象投影到参考线或参考面上。

第一维特征的提取：

a)选择导航点O_a、O_b(这两点确定了参考线)。很明显，我们希望选中距离相距最远的点。为了降低计算复杂度，使用了启发式算法：

step1：从N个对象中任选一个对象，假定是O_a；

step2：计算O_a与其它N－1个对象之间的距离，选择与其相距最远的对象，设其为O_b。再将O_a与O_b互换，此步骤重复若干次；

step3：最后得到的O_a、O_b就是选中的相距距离最远点。

b)投影计算：对于其余的每个对象O_i，它与O_a、O_b构成一个三角形$O_aO_bO_i$，由如下的余弦法则：

$$x_i=\frac{d_{a,i}^2+d_{a,b}^2-d_{b,i}^2}{2d_{a,b}}$$

可以得到点O_i在直线O_aO_b上的投影与点O_a的距离x_i。

经过这两步，我们就能将对象映射成参考线上的点，同时保持它们之间的一些距离信息。例如：如果O_i与O_a较近，则得出的x_i相对较小。这样，投影到一维空间的问题就解决了，x_i就是对象i被提取的第一维特征。

二维以上特征的提取：首先选取一平面，垂直于上次相距最远点O_a、O_b所确定的直线，然后把所有点投影到这个平面上。这样，问题就与最初一样了。唯一的问题就是计算投影之间的新距离：

$$(D'(O'_i,O'_j))^2=(D(O_i,O_j))^2-(x_i-x_j)^2 \quad i,j=1,\cdots,N$$

这样进行k'次之后，将每次各点的投影值(x_i)组合起来，

*)本文研究得到国家自然科学基金(60005004)和安徽省自然科学基金(批准号：01042302)资助。林晓立　实验师，硕士生，主要研究领域为数据挖掘与智能信息处理等。陈恩红　博士，副教授，主要研究领域为机器学习、数据挖掘与智能信息处理等。任皖英　讲师，硕士生，主要研究领域为信息检索与智能信息处理等。

就形成了它们的新特征值。

2. 新数据对象降维过程

如需对新数据对象进行 k' 特征提取，只要对其进行 k' 次投影并取各次投影值即可，投影参考线分别为上述过程各次采用的距离最远点所确定的直线。

FastMap 算法的计算复杂度为 $O(tkn)$，其中 t 为每次选择参考点对时循环的次数，n 为数据个数，k 为提取出的特征数。

2.2 Bourgain 算法[1]

1. 特征提取过程

Bourgain 算法不是一种具有"疏松"性的算法，它需要计算所有对象之间的距离。它是 Cofe 算法的基础，其特征提取原理如下：

假设集合 X 包含 n 个对象，对象间的距离函数为 d，集合 X 中的元素 x 与 X 的子集 X′的距离函数为 D：

$D(x,X')=\min_{y\in x'}\{d(x,y)\}$。

又假设 $R=\{X_1,X_2,\cdots,X_k\}$是 X 的一系列子集，我们可以定义一个关于 R 的 embedding 函数：

$E_R(x)=\{D(x,X_1),D(x,X_2)\cdots,D(x,X_k)\}$。

可以用 $E_R(x)$表示对象 x 的特征的原因：假设空间中的对象被很好地分配在各个簇中，那么，如果一个子集 X_1含有一个点 x∈簇 C1，并且 x∉簇 C2，则对于簇 C1中的对象 x 来说，其 $D(x,X_1)$比簇 C2中的对象 y 的 $D(y,X_1)$小。这样，就能够体现出同类内对象之间特征相似，异类对象间特征区别较大。

Bourgain 算法的主要步骤：

Step1. 选取参考子集 R：

集合 R 包含 $O(\log^2 n)$个子集合 $X_{i,j}$。设想它们被排成 κ 列 β 行。$R=\{X_{1,1},X_{1,2},\cdots,X_{1,k},X_{2,1}\cdots,X_{2,k},X_{\beta,1}\cdots X_{\beta,k}\}$。其中 $\kappa=O(\log n)$，$\beta=O(\log n)$。

Step2. 对每个对象 x 计算其 $E_R(x)$；

Step3. 将所有对象的 E_R 组合起来，就形成了新的向量空间。

2. Bourgain 算法的缺点

(1) 产生的向量维数过高，为 $O(\log^2 n)$。如果我们有1024个对象，就要产生100维新向量。

(2) 由于参考集过多，每个对象都有可能被选在某个参考集中，这样，就需要计算所有对象之间的距离，算法复杂度太高，为 $O(n^2)$。

2.3 Cofe 算法[3]

1. 特征提取过程

Cofe (Complex Object Feature Extraction)，复杂对象特征提取算法是对 Bourgain 算法的一种启发式修改，用来减少距离计算量。这种修改体现在两个方面：

1)Bourgain 算法的外循环针对所有对象，内循环计算所有的特征。Cofe 算法则相反，先计算所有对象的第一个特征，再计算第二个，循环下去，直到所计算出的特征满足要求为止。这样，就不一定要计算出全部的 $O(\log^2 n)$维特征。从而大大减少了计算量。

(2)由于原始对象是复杂对象，因而计算它们的距离计算量非常大，Cofe 算法先对所有对象计算出前 k' 个特征。这样，我们对这些对象之间的距离就有了初略的估计：

$$d'_{k'}(p,q)=\sqrt{\sum_{l=1}^{k'}(p_l-q_l)^2}$$

其中 p_l 是对象 p 的第 l 个特征，q_l 也类似。

具体地，当计算第 k 维特征时，对于数据集 X 中的每个对象 p 和参考集 X_k，近似距离是由下列计算确定的：

(a)对每个对象 $q\in X_k$，计算近似距离 $d'_{k-1}(p,q)$；

(b)从参考集 X_k 中选出 σ 个对象，它们具有距 p 最近的近似距离；

(c)对这 σ 个对象中的每个对象 q，计算与对象 p 的实际距离 $d(p,q)$；

(d)$D(p,X_k)=d(p,q')$，q' 是 σ 个对象中具有实际最小距离的对象。

2. 算法精确计算量分析

Cofe 算法的计算复杂度为 $O(\sigma kn)$，其中 σ 为近似距离较近点的个数。假设用实际距离处理参考集 α 行中的 β 行，剩下的 α-β 行参考集中，处理第 i 个特征时是先用已算出的 $i-1$ 维特征产生的近似距离来代替实际距离，然后从中找出与当前处理数据近似距离最近的 σ 个对象，再求这 σ 个对象与当前处理对象的实际距离中的最近距离。假设 n 为数据个数，k 为提取出的特征数，d 为计算一次平方距离的代价，i 为空间的维数，D 为计算一对对象实际距离的代价，则

前 β 行中 $T(\beta)=2nk(2^\beta-1)D$，后 α-β 行中 $T(\alpha-\beta)=nk\sigma(\alpha-\beta)D+n\sum_{i=\beta+1}^{\alpha}(2^i\sum_{j=k(i-1)}^{ki-1}(jd))$

$T_{Cofe}=T(\beta)+T(\alpha-\beta)=nk(2^{\beta+1}-2+\sigma(\alpha-\beta))D+nk((\alpha-1)2^{\alpha+1}-\beta 2^{\beta+2}+4)d$

由于计算 D 比计算 d 代价大得多，因此计算量的主要部分为：

$T_{Cofe}=nk(2^{\beta+1}+\sigma(\alpha-\beta))D$

2.4 Cofe 算法的改进-ImprovedCofe 算法

由于 Cofe 算法中参考集中各元素的选取是随机的，那么如果选取的元素属于"粗糙"数据，即该数据并不明显属于某簇。这样，由这些元素作为参考点产生的特征，对不同簇中的元素不能体现出同类相似、异类相异。我们对参考集元素选取方法改进如下：

假设已知待求数据类别数 k，求出每类的"中心"点，假设为 $A_1,A_2,\cdots,A_k$。选取参考集时，参考点在 $A_1,A_2,\cdots,A_k$ 中选取，其它步骤与 Cofe 算法类似。

由于参考元素为各类的"中心"点，因此能够在总体上最大程度地体现出同类相似、异类相异。因而得出的特征应最为真实、有效。实验表明，这种改进效果较好。

3 实验结果

实验所用环境为 WIN98平台下的 Visual C++6.0，计算机配置为赛扬500，128M 内存。FastMap 算法所用的参数 t=3，Cofe 算法所用的参数 σ=3。所用数据为"模拟"的文本数据，即假设每个数据对应一个文本。数据是用 k 维空间中的向量来表示的(k 是文本集合中单词的个数)。对于一个文本来说，其数据向量的每一位的值代表参考词典中某个单词是否在该文本中出现，如这个文本中出现这个词，则该位置的值为1，否则为0。

对象实际距离计算方法如下：

$$similarity(d_1,d_2)=\frac{\vec{u}_1\cdot\vec{u}_2}{\|\vec{u}_1\|^2\cdot\|\vec{u}_2\|^2}$$

$$D(d1,d2)=2\times sin(\theta/2)=\sqrt{2\times(1-cos(\theta))}=\sqrt{2\times(1-similarity(d_1,d_2))}$$

3.1 可视化及聚类效果

本实验对四类、每类各50个的高维数据进行了三维特征提取，原始维数为10000，效果如图1～3：

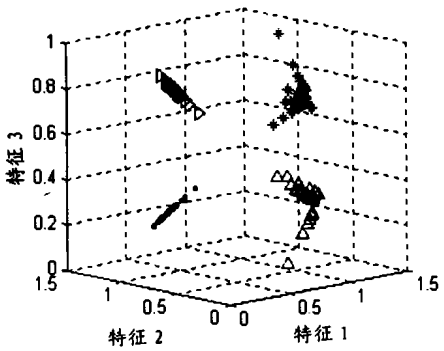

图1 FastMap 算法聚类效果图

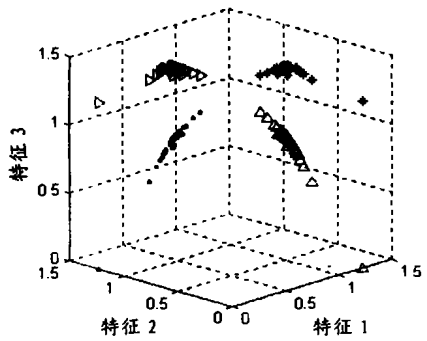

图2 Cofe 算法聚类效果图

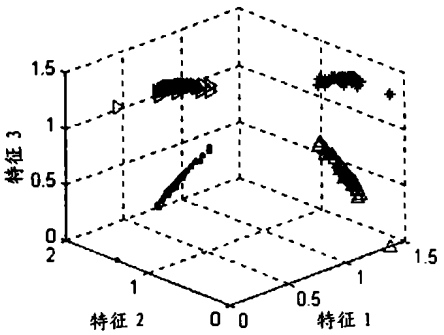

图3 ImprovedCofe 算法聚类效果图

实验表明这三种算法对高维数据均具有较好的可视化及聚类效果。

3.2 特征质量

实验对20类、每类各100个、共2000个数据进行了特征提取，原维数为10000。

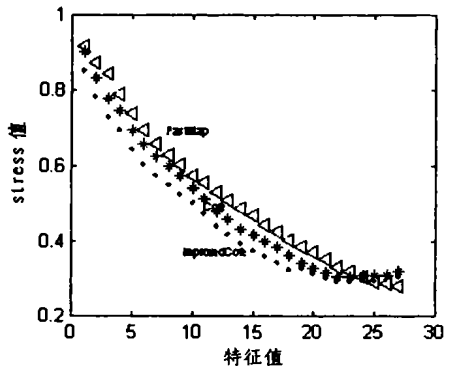

图4 各算法特征质量比较图

本实验表明：

(1)FastMap 算法提取特征时，stress 值随着提取特征维数的增加而持续降低；

(2)Cofe 及 Cofe_New 算法提取特征时，提取的特征数达到其数据类别数时，stress 值就不再下降，反而上升；

(3)ImprovedCofe 算法的特征提取效果比 Cofe 算法的更好。

3.3 运行时间

本实验对25类、共2000个数据进行了特征提取，原维数为10000。

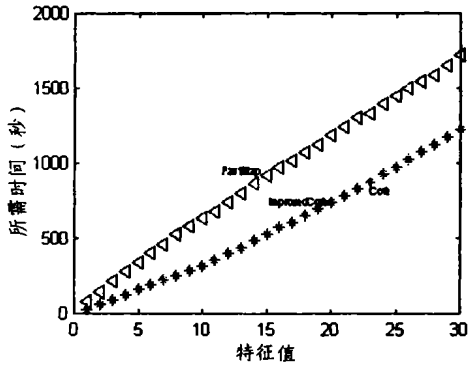

图5 各算法特征运行时间比较图

实验表明 Cofe 算法的运行时间大约是 FastMap 算法运行时间的3/4，与 ImprovedCofe 算法的运行时间相似。

总结 Bargain 算法仅具有理论价值，而 FastMap 算法及 Cofe 算法在对高维数据特征提取方面均具有较好的效果。比较而言，FastMap 算法得出的结果较好，适用于要求处理精度高的场合；Cofe 算法对大量数据提取特征的速度较快，适用于数据量大的场合。此外，Cofe 算法提取的特征数只要达到处理数据的类别数即可，可通过其它改进方法使其产生的 stress 值继续下降。在已知数据类别信息时，可对 Cofe 算法的参考点用“中心”点代替，从而得出更好的效果。

参 考 文 献

1 Bourgain J. On Lipschitz embedding of finite metric spaces in Hilbert space. Isreal J. of Math, 1985, 52: 46～52

2 Faloutsos C, King-Ip lin. FastMap: a Fast Algorithm for Indexing, Datamining and Visualization of Traditional and Mltimedia Dtasets. ACM SIGMOD, 1995, 24(2): 163～174

3 Hristescu G, Farach-colton M. Cofe: A Scalable Method for Feature Extraxtion from Complex Objects. In: Proc. 2nd Intl. Conf. on Data Warehousing and Knowledge discovery, 2000. 358～371

4 Jagadish H V. Linear clustering of objects with multiple attributes. In: Proc. of ACM SIGMOD Conf. 1990. 332～342

5 Ramarkrishnan G V, et al. Clustering large datasets in arbitrary metric spaces. In: Proc. 15th ICDE Conf., 1999. 142～176

6 YaKatama N, Satoh S. The Sr-tree: An index structure for high-dimensional nearest neighbor queries. In: Proc. ACM SIGMOD Conf., 1997. 369～380