LSNCCP——一种基于最大不相含核心点集的聚类算法

薛永生[1] 翁 伟[1] 文 娟[1] 王劲波[2] 张 宇[1]
[1](厦门大学计算机科学系 厦门 361005)
[2](厦门大学经济学院 厦门 361005)
(weng-wei5205@sina.com)

摘 要 聚类在数据挖掘、模式识别等许多领域有着重要的应用. 提出了一种新颖的聚类算法: 一种基于最大不相含核心点集的聚类算法 LSNCCP(a clustering algorithm based on the largest set of not-covered core points). 在密度定义的基础上, 考察核心点之间的距离关系, 定义相含、相交、相离这 3 种核心点之间的关系, 最后找出一个最大不相含核心点集, 在此基础上进行聚类, 并且找到解决丢失点问题的快速方法. 该最大不相含核心点集只是全部核心点集合的一个很小的子集, 因此有效地缩减了同类算法中搜寻核心点的时间. 理论和实验上证明了这种算法的可行性和优越性.

关键词 数据挖掘; 聚类; 密度; 核心点; 最大不相含核心点集

中图法分类号 TP311.13

LSNCCP: A Clustering Algorithm Based on the Largest Set of Not-Covered Core Points

XUE Yong-Sheng[1], WENG Wei[1], WEN Juan[1], WANG Jing-Bo[2], and ZHANG Yu[1]
[1](*Department of Computer Science, Xiamen University, Xiamen* 361005)
[2](*Economic School, Xiamen University, Xiamen* 361005)

Abstract Clustering is an important application area for many fields including data mining, pattern recognition, etc. In this paper, a novel clustering algorithm LSNCCP(a clustering algorithm based on the largest set of not-covered core points) is proposed. On the basis of the definition of density, the distance between the core points is discussed. And then, the three essential distance relation: covered core points, intersectant core points, and separate core points. Finally, the largest set of not-covered core points is found and based on the set the data can cluster very well. Because the largest set of not-covered core points is a lesser subset of the all core points, the new algorithm cuts short the time of searching all core points in the similar algorithms. The feasibility and the advantage or the new algorithm are proved in theory and experiment.

Key words data mining; clustering; density; core points; largest set of not-covered core points

1 引 言

数据挖掘(data mining)就是从大量的数据中发现隐含的、先前未知的、对决策有潜在价值的规则的过程[1]. 经过近 20 年来的发展, 该领域的研究已经取得了丰硕的成果, 并产生了许多应用于实际商业活动的系统, 为企业带来了巨大的效益.

聚类分析是数据挖掘中的一个重要的任务. 聚类分析就是根据样本之间的相似度将其自动地分为几个群组, 且使得同一群组内的样本基本相似, 而不属于同一群组的样本不相似的方法. 我们称这些群

收稿日期: 2004-07-15
基金项目: 福建省自然科学基金项目(A0310008); 福建省高新技术研究开放计划重点项目(2003H043)

组为分区(类). 聚类直观上把大量的数据进行了自动的综合描述, 这种描述对于我们进一步分析数据提供了方便. 由于对于处理来说, 样本常称为对象或记录, 坐标上用点表示, 所以下文若在不引起歧义的地方, 它们之间具有相同的含义.

(1) 相关工作

依据对聚类的不同理解, 目前人们已经提出了许多的聚类算法, 如 DBSCAN[2], CURE[3], CLIQUE[4]等.

DBSCAN 是一种基于密度的聚类算法. 其基本思想是: 对于一个聚类中的每一个对象, 在其给定的半径 Eps 的邻域中包含的对象不能少于某一给定的最少数目 $MinPts$, 然后对具有密度连接特性的对象进行聚类. 在该算法中, 发现一个聚类的过程是基于这样的事实: 一个聚类能够被其中的任意一个核心对象(参考点)所确定. 该算法可以挖掘任意形状的聚类, 对数据输入顺序不敏感, 并且具有处理异常数据(噪音)的能力. 该算法的时间复杂度为 $O(n^2)$.

由 DBSCAN 的基本思想可以看出, 如何求出核心对象是关键的. 文献[5]提出了一种基于密度的快速聚类算法. 其基本思想是通过选用核心对象附近区域包含的所有对象的代表对象作为种子对象来扩展, 该算法减少了区域查询的次数, 从而减少了聚类的时间和 I/O 开销. 该算法必须对可能因为简化而丢失的对象进行处理. 文献[6]提出了一种基于参考点和密度的快速聚类算法 CURD(clustering using references and density). 该算法采用一定数目的参考点(references)来有效地表示一个聚类区和形状, 不过该参考点是虚拟的, 并且一个聚类中的参考点数目不是固定的. CURD 算法也是采用密度的方法屏蔽异常数据(噪音)对算法的影响, 与 DBSCAN 算法在处理任意形状聚类方面的能力接近. CURD 算法具有和 K-means 算法相同的时间复杂性, 此外, 该算法采用距离的方法可以将高维数据的处理转换到一维空间[7,8], 在这个意义上可以处理高维数据.

(2) 本文的工作

本文提出了一种基于最大不相含核心点集的聚类算法 LSNCCP(a clustering algorithm based on the largest set of not-covered core points). 在定义密度的基础上, 通过考察核心点之间的 3 种关系, 进而选出所有核心点的一个子集, 该子集满足特定的关系, 称为最大不相含核心点集(largest set of not-covered core points), 在此基础上进行聚类. 该算法能高效地产生聚类并且不丢失任何可以聚类的点. 实验结果证明, 该方法在速度和精度方面都有非常高的效率, 特别适用于高密度大数据量的聚类. 本文的内容结构安排如下: 首先介绍相关的概念, 然后详细给出算法描述, 然后给出实验结果与分析, 说明该算法的可行性和优越性.

2 相关概念

定义 **1**. 点的密度. 给定空间中任意的一个点 p 和半径 Eps, 以点 p 为中心, 以 Eps 为半径的区域内的点的个数称为点 p 基于距离 Eps 的密度(density), 记做 $Density(p, Eps)$.

定义 **2**. 代表区域. 以点 p 为圆心、半径为 Eps 的圆形区域, 我们称该区域为核心点 p 的代表区域, 该区域内的点集合记做 $N_{Eps}(p)$.

定义 **3**. 核心点. 给定空间任意一个点 p, 距离 Eps 和阈值 $MinPts$, 如果满足 $N_{Eps}(p) \geqslant MinPts$, 则称 p 为核心点(core points), 同时我们称为 $MinPts$ 为密度阈值(density threshold).

下面给出核心点之间的 3 种关系的定义.

定义 **4**. 相含核心点. 如果两个核心点 p, q, 有 $Dist(p, q) \leqslant Eps$, 则称 p, q 互为相含核心点(cover core points).

定义 **5**. 相交核心点. 如果两个核心点 p, q, 有 $Eps < Dist(p, q) \leqslant 2Eps$, 则称 p, q 互为相交核心点(intersectant core points).

定义 **6**. 相离核心点. 如果两个核心点 p, q, 有 $Dist(p, q) > 2Eps$, 则称 p, q 互为相离核心点(separate core points).

定义 **7**. 不相含核心点. 如果两个点 p, q 不是相含核心点, 则就称 p, q 互为不相含核心点. 也就是 p, q 要么是相交核心点, 要么是相离核心点.

定义 **8**. 噪音. 不在任何核心点的代表区域内的点称为噪音. 显然噪音不能聚集到任何一个类中.

为了下文算法描述的方便性, 用 Eps 表示设定的半径. 另外, 设定一个密度阈值 $MinPts$. 同时我们用 $Add(S, p)$ 表示把点 p 加到集合 S 中去, 用 $Delect(S, p)$ 表示把点 p 从集合 S 中删除.

3 算法描述

算法流程如图 1 所示:

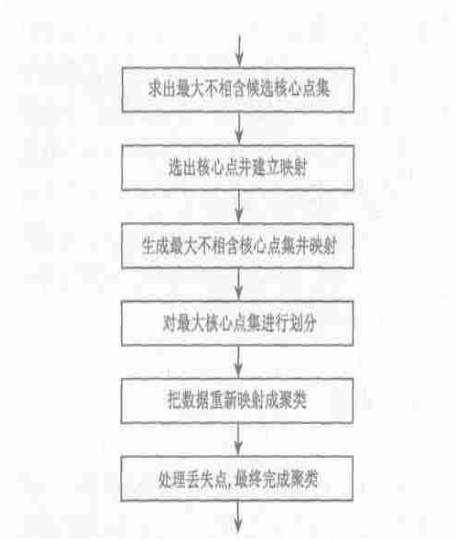

图 1 算法流程图

从图 1 可以看出, 算法首先要求出一个最大不相含核心点集, 把每个核心点都看成是一个小类的中心建立一个个单独的类, 然后, 如果有两个核心点是相交核心点, 则合并这两个小类为一个大类; 同理, 如果一个核心点与一个大类中某个核心点是相交核心点, 则也相合并. 这就是对最大不相含核心点集的分类和重新建立数据到类的映射的过程. 然后找回丢失点, 正确聚类到某一个类中, 最终完成聚类. 下面用两个算法来详细描述上面的过程.

3.1 最大不相含核心点集选取算法

本步骤实现这样的目标: 找出所有核心点的一个子集 *ReferenceSet*, 在该子集中任何两个点均不相含, 并且不存在其他的核心点 r, $r \notin ReferenceSet$, 且 r 与集合 *ReferenceSet* 中任何点都不相含. 不妨称这种集合为最大不相含核心点集.

这一步骤又细分为两个步骤: 首先找出那些可能成为核心点的点, 不妨称这些点为候选核心点; 然后进一步判断这些点是否是核心点, 并同时建立数据到核心点的映射, 如算法 1 所示:

算法 **1**.

GenerateReference (*PointSet*, *Eps*, *MinPts*, *ReferenceSet*)

①For i= 1 to *PointSet*.*size*{
②If *dist*(*ReferenceSet*, P_i) > *Eps* Then
③*Add*(*ReferenceSet*, P_i);
④End If}
⑤For j = 1 to *ReferenceSet*.*size*{
⑥*Result*= *PointSet*. *RegionQuery*(P_j, *Eps*);
⑦If *Result*.*Size* ≥*MinPts* Then
⑧*PointSet*. *ChangeClIds*(*Result*, j);
⑨Else *Delete*(*ReferenceSet*, P_j);}
⑩For k= 1 to *PointSet*.*size*{
⑪If P_k. *ClIds*= 0 and *dist*(*ReferenceSet*, P_k) > *Eps* Then{
⑫*Result*= *PointSet*. *RegionQuery*(P_k, *Eps*);
⑬If *Result*. *Size* ≥*MinPts* Then{
⑭*PointSet*. *ChangeClIds*(*Result*, k);
⑮*Add*(*ReferenceSet*, P_k);}
⑯}}

算法 1 最大不相含核心点生成并建立数据到核心点的映射算法开始时, 核心点集合 *ReferenceSet* 是空的. 语句①~④用来选取候选核心点. 该步骤对所有待聚类数据进行逐个检查: 对于任何一个数据(点) P_i, 一旦发现与候选核心点集合中的任意点 R_j 的距离都大于预先设定的半径 *Eps*, 则把点 P_i 加入到该集合之中. 这个循环结束后, 对集合 *ReferenceSet* 中的所有点进行区域查询, 如语句⑤~⑨所示. 一旦发现某个点的代表区域内的点的个数不少于 *MinPts*, 则该点是核心点, 把该区域看成是一个小类, 做数据到这个核心点的映射, 如语句⑧所示; 否则, 在集合 *ReferenceSet* 中删除该点. 此时, 集合 *ReferenceSet* 中的点虽然都是核心点并且两两不相含, 但不是一个最大的不相含核心点集. 语句⑩~⑯最终生成一个最大不相含的核心点集. 语句⑪表示如果当前点没有与核心点集中任何一个点做映射(说明没有在任何已经选出的核心点的代表区域中), 则首先判断该点与选出的核心点集中的任意点之间的距离是否大于 *Eps*, 若大于 *Eps* 则在语句⑫中进行区域查询, 在语句⑬中判断该点的代表区域中的点如果不小于 *MinPts*, 表明该点是一个核心点, 且与集合 *ReferenceSet* 中的点都不相含, 则加进核心点集合之中, 并做数据到该点的映射. 整个过程在生成了一个最大的不相含核心点集的同时把数据做了到选出的核心点的映射, 形成了一个个小类.

对算法 1 需要进一步说明:

首先, 这个算法其实是一个贪心算法. 正因为这是一个贪心算法, 所以以不同的顺序输入数据时, 得到的候选核心点的数量有所不同. 同样就会导致语句⑤~⑨得出的核心点的数目有少量偏差. 在此做了实验验证. 我们对 700 个二维数据运行上述两个算法. 该 700 个点中每个点的 x 坐标范围是(0, 4), y 坐标范围也同样是(0, 4). 我们在 *Eps*= 0.1, *MinPts*= 3 的情况下以不同顺序输入数据共做了 4 次实验, 结果如表 1 所示(第 3 行相应聚类点个数是

指由第 2 行的核心点能够进行聚类的点的个数, 后文有分析).

表 **1**　以不同顺序数据的实验结果

次序	候选核心点个数	核心点的个数	相应聚类点个数
第 1 次	205	97	545
第 2 次	212	101	543
第 3 次	205	101	551
第 4 次	210	103	543

虽然算法 1 对数据输入的顺序敏感, 也就是不同的输入顺序可能导致产生的最大不相含核心点集不一样. 但是因为有后面对丢失点的处理, 并不会影响我们目标. 特别说明, 最大不相含核心点集可能不只一个, 我们只要找出任何一个就可以了. 同时很容易看出, 最大不相含核心点集只是所有核心点集合的一个很小的子集.

3.2　聚集和处理丢失点

聚类有两个问题: 对于输入的数据最后分为几个类和每个类中包含那些数据. 在本文中, 类 C_i 和类 C_j 之所以划分成不同的两个类, 是因为这两个类中的最近的核心点之间的聚类大于 $2Eps$. 事实上只对相应的最大不相含核心点集进行处理就可以知道分为几个类了. 开始我们把最大不相含核心点集中的每一个点 P_i 看成是一个类 C_i, 即 $C_i = \{P_i\}$, 然后, 如果类 C_i 中有一个点 p 与类 C_j 中一个点 q 互为相交核心点, 则合并这两个类成一个类 C, 既 $C = C_i \cup C_j$. 然后, 对输入的整个数据, 逐一查看每一个点与某个类中的核心点的距离, 一旦发现距离小于等于 Eps, 即分配到该类中. 这个过程结束后得到的类的个数就是整个数据可以分出的类数. 换句话说, 任何最大不相含核心点集就代表了聚类的空间几何特性. 这里的空间几何特性是指输入的数据到底分为那几个类以及类中有那些点.

定理 **1**. 任何最大不相含核心点集就代表了聚类的空间几何特性.

证明. 可以从以下几个方面来证明:

一方面, 不相含核心点集可以区分任何一个类. 根据最大不相含核心点集的生成算法, 任何类中都至少有一个核心点被选入最大不相含核心点集, 在聚类中又能准确地把代表该类的核心点选在一个类中.

另一方面, 已聚类的点 p, 只可能分配到一个类中. 反之, 设点 p 既可以分配到类 C_i 中, 又可以分配到类 C_j 中, 则在类 C_i 中有一核心点 q, $p \in N_{Eps}(q)$, 同理, 在类 C_j 中有一个核心点 r, $p \in N_{Eps}(r)$. 则由点 p, q 和 r 形成的三角形中, 边 $pq \leqslant Eps$, 边 $pr \leqslant Eps$, 故边 $qr \leqslant (pq + pr) \leqslant 2Eps$, 这样, 类 C_i 与类 C_j 应该合并为一个类.　　证毕.

我们已经得到了最大不相含核心点集 *ReferenceSet*, 不过仅仅用这个集合中的核心点来聚类, 会有一些本该聚类的点得不到聚类, 也就是说聚类结果会丢失一些点. 显然最大不相含核心点集 *ReferenceSet* 中任何一个点都被聚类. 对于被丢失的点 p, 有下面的定理

定理 **2**. 丢失点 p 一定不是核心点.

证明. 用反证法. 反设该点 p 是核心点. 此时一定有 $p \notin ReferenceSet$, *ReferenceSet* 是选出的最大不相含核心点集. 因此, p 至少有一个相含核心点 q, $q \in ReferenceSet$. 否则, 根据最大不相含核心点集的定义, p 应选进集合 *ReferenceSet* 中去. 既然这样, 由于 $Dist(p, q) \leqslant Eps$, p 应当被聚类. 此与题设矛盾. 故丢失的点 p 一定不是核心点.　　证毕.

定理 2 提供了一种找出丢失点的简单的方法. 可以把一个没有得到聚集的点 p 的代表区域内的所有的点集合 $N_{Eps}(p)$ 与已经得到聚类的点的集合做一个交集, 再判断这个交集中的点是否是核心点. 一旦判断出一个点是核心点, 就说明这个点是丢失点, 我们就把这个丢失点 p 聚集到这个核心点同一个类中去. 如果这个交集为空, 或者交集中没有核心点, 则这个点是噪音.

综上所述, 用算法 2 表示如下:

算法 **2**. 聚类并处理丢失点.

Clustering (PointSet, ReferenceSet, Eps, outliersSet)

① *Cluster*[1] = {*ReferenceSet.get*(1);}
② *K* = 1;
③ For *i* = 2 to *ReferenceSet.size* Do{
④　*CurrentP* = *ReferenceSet.get*(*i*);
⑤　If *CurrentP.IsClassified* = false Then{
⑥　　*bool* = 0;
⑦　　For *j* = 1 to *k* Do{
⑧　　If *Dist*(*Cluster*[*j*], *CurrentP*) ⩽ 2*Eps* Then{
⑨　　If *bool* = 0 Then{
⑩　　*bool* = *j*;
⑪　　*Add*(*Cluster*[*j*], *CurrentP*);}
⑫　　Else{

⑬ *union*(*Cluster*[*bool*], *Cluster*[*j*]);
⑭ *Delete*(*Cluster*[*j*]);
⑮ *k*= *k*- 1;}
⑯ }}
⑰ If *bool*= 0 Then{
⑱ *k*= *k*+ 1;
⑲ *Cluster*[*k*]= {*CurrentP*};}}
⑳ For *i*= 1 to *PointSet*.*size* Do{
㉑ If P_i.*ClId* ∈ *Cluster*[*id*] Then
㉒ P_i.*ClId*= *id*;
㉓ Else
㉔ *Add*(*outliersSet*, P_i);}
㉕ For *j*= 1 to *outliersSet*.*size* Do{
㉖ For each *R* in $N_{Eps}(Q_j)$, If *R*.*ClId*< > 0 and *R* is a core point Then
㉗ Add Q_j in the cluster which *R* is in it.}

在算法 2 中, 我们在语句 ①~ ⑲对最大不相含核心点集进行划分, 最后形成几个划分, 即表示最终可以聚成几个类. 语句 ⑳~ ㉔对每一个点进行检查, 如果该点的映射标志在某个划分中则重新映射到该划分中去; 否则暂时视为噪音. 语句 ㉕~ ㉗识别出丢失点并聚类它们.

4 实验分析

首先, 利用本文提出的算法, 对图 2(a) 中的 1100 个点进行聚类, 聚类结果在图 2(b)(c) 中. 图 2(b) 表示得到的一个最大不相含核心点集, 共有 99 个点; 图 2(c) 为聚类结果, 共有 933 个点. 从结果中可以看出, 该算法的结果跟人的直觉一样, 能够进行正确的聚类

图 2 1000 个点及其聚类结果

从图 2 中也可以看出, 本算法对聚类的形状不敏感, 能够发现任意形状的聚类, 能有效地排除噪音.

我们对上文提到过的 700 个数据以不同的顺序输入初始数据, 得到表 2.

可以看出, 尽管初始数据输入顺序不同, 但是最终的聚类结果是相同的. 用穷尽的方法可知, 该数据集共有 538 个核心点, 但我们选出的最大不相含核心点集已经大大少于这个数目. 同时, 我们用 DBSCAN 对这个数据进行聚类, 得到的结果是一样的, 但 DBSCAN 耗费时间多得多.

表 2 **700** 个点聚类的正确结果

次序	最大不相含核心点的个数	聚类点的个数
第 1 次	110	593
第 2 次	108	593
第 3 次	106	593
第 4 次	106	593

为了进一步说明问题, 我们做了一个实验比较了 DBSCAN 算法和本文提出的算法的时间效率.

实验环境为 P Ⅳ 1.6 GHz CPU, 物理内存 256MB, 可用内存 100 MB, 硬盘空间 40 GB, 操作系统 Windows 2000 Server, 程序用 VB6.0 语言编写. 所有数据读入内存, 每条记录仅仅只有每一维度上的值, 不使用任何索引机制, 区域查询时的距离信息是即时求出. 图 3 是对 DBSCAN, LSNCCP 性能测试的结果:

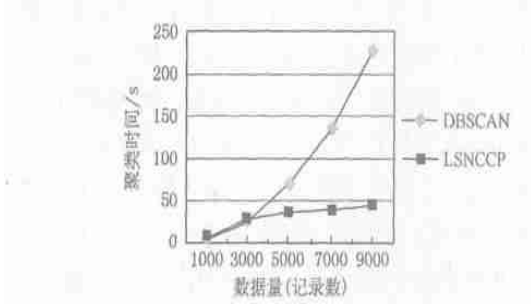

图 3 LSNCCP 算法与 DBSCAN 算法的比较

从图 3 可以看出, 在数据量少的时候, LSNCCP 的效率可能比 DBSCAN 稍差, 其实这跟参数设置有关. 在数据越稀疏, 密度阈值越少, 半径越少的情况下, LSNCCP 的最大不相含核心点集合太少, 在极端的情况下为空, 导致不能体现 LSNCCP 的优势. 但是随着数据量的增多, LSNCCP 的优势体现得非常明显.

5 结束语

聚类算法一直是数据挖掘中令人关注的重点. 本文在密度的概念上提出了一种新颖高效的聚类算法. 理论和实验都证明这个算法是可行的, 并在理论上说明了其高效性. 我们下一步的工作是在大型数据库环境下测试算法的性能, 与各种基于密度的算法比较其性能和参数对性能的影响. 对于高维空

间中, 由于超出了人的视觉空间, 一般的距离公式不再具有实在的意义, 我们将同时研究结合密度的方法发现高维空间中噪音的可行性和效率问题.

参 考 文 献

1 M Fayyad, G Piatetsky-Shapiro, P Smyth. From data mining to knowledge discovery: Anoverview. In: M Fayyad, G Piatetsky-Shapiro, P Smyth, eds. Advances in Knowledge Discovery and Data Mining. Menlo Park, CA: AAAI Press, 1996. 1~ 36

2 M Ester, HP Kriegel, J Sander, *et al*. A density based algorithm for discovering clusters in large spatial databases with noise. In: E Simoudis, JW Han, UM Fayyad eds. Proc of the 2nd Int'l Conf on Knowledge Discovery and Data Mining. Portland: AAAI Press, 1996. 226~ 231

3 S Guha, R Rastogi, K Shim. CURE: An efficient clustering algorithm for large databases. In: LM Haas, A Tiwary eds. Proc of the ACM SIGMOD Int'l Conf on Management of Data. New York: ACM Press, 1998. 73~ 84

4 R Agrawal, J Gehrke, D Gunopolos, *et al*. Automatic subspace clustering of high dimensional data for data mining application. In: LM Haas, A Tiwary eds. Proc of the ACM SIGMOD Int'l Conf on Management of Data. New York: ACM Press, 1998. 94~ 105

5 周水庚, 周傲英, 曹晶, 等. 一种基于密度的快速聚类算法. 计算机研究与发展, 2003, 37(11): 1287~ 1292
(Zhou Shuigeng, Zhou Aoying, Cao Jing, *et al*. A fast density-based clustering algorithm. Journal of Computer Research and Development(in Chinese), 2003, 37(11): 1287~ 1292)

6 马帅, 王腾蛟, 唐世渭, 等. 一种基于参考点和密度的快速聚类算法. 软件学报, 2003, 14(6): 1089~ 1095
(Ma Shuai, Wang Tengjiao, Tang Shiwei, *et al*. A fast clustering algorithm based on reference and density. Journal of Software (in Chinese), 2003, 14(6): 1089~ 1095)

7 S Berchtold, C Bohm, H-P Kriegel. The pyramid-technique: Towards breaking the curse of dimensionality. In: LM Haas, A Tiwary eds. Proc of the ACM SIGMOD Int'l Conf Management of Data. New York: ACM Press, 1988. 142~ 153

8 C Yu, BC Ooi, K-L Tan, *et al*. Indexing the distance: An efficient method to KNN processing. In: PMG Apers, P Atzeni, S Ceri, *et al* eds. Proc of the 27th Int'l Conf on Very Large Data Bases. San Francisco, CA: Morgan Kaufmann, 2001. 421~ 430

薛永生　男, 1946 年生, 教授, 主要研究方向为数据库理论与应用、分布式数据库、数据仓库等.

翁　伟　男, 1979 年生, 硕士研究生, 主要研究方向为分布式数据库、数据仓库、数据挖掘等.

文　娟　女, 1982 年生, 硕士研究生, 主要研究方向为数据库理论与应用、分布式数据库、数据仓库、数据挖掘等.

王劲波　男, 1978 年生, 硕士, 助教, 主要研究方向为分布式数据库、实时数据库等.

张　宇　女, 1977 年生, 硕士研究生, 主要研究方向为数据库、数据仓库和电子商务等.