计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2019,28(8):162−169 [doi: 10.15888/j.cnki.csa.007009]
©中国科学院软件研究所版权所有.
E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 CURE 聚类算法改进的原型选择算法①

孙元元, 张德生, 张　晓

(西安理工大学 理学院, 西安 710054)
通讯作者: 孙元元, E-mail: 506027677@qq.com

摘　要: 针对传统 K 近邻分类器在大规模数据集中存在时间和空间复杂度过高的问题, 可采取原型选择的方法进行处理, 即从原始数据集中挑选出代表原型 (样例) 进行 K 近邻分类而不降低其分类准确率. 本文在 CURE 聚类算法的基础上, 针对 CURE 的噪声点不易确定及代表点分散性差的特点, 利用共享邻居密度度量给出了一种去噪方法和使用最大最小距离选取代表点进行改进, 从而提出了一种新的原型选择算法 PSCURE (improved prototype selection algorithm based on CURE algorithm). 基于 UCI 数据集进行实验, 结果表明: 提出的 PSCURE 原型选择算法与相关原型算法相比, 不仅能筛选出较少的原型, 而且可获得较高的分类准确率.

关键词: K 近邻分类器; 原型选择; 共享邻居密度; CURE 层次聚类; 代表点

引用格式: 孙元元,张德生,张晓.基于 CURE 聚类算法改进的原型选择算法.计算机系统应用,2019,28(8):162–169. http://www.c-s-a.org.cn/1003-3254/7009.html

Improved Prototype Selection Algorithm Based on CURE Algorithm

SUN Yuan-Yuan, ZHANG De-Sheng, ZHANG Xiao

(Faculty of Science, Xi'an University of Technology, Xi'an 710054, China)

Abstract: Since the traditional K-nearest neighbor classifier possesses large time and space complexity for larger-scale data sets, prototype selection is an effective processed method which selects representative prototypes (instances) from the original data set for K-nearest neighbor classifier without reducing the classification accuracy. At present, there exist many prototype selection methods. In this paper, based on the existing CURE algorithm, which is difficult to determine the noise points and has bad dispersed of representative points, the shared neighbor density metric is presented to delete noise points and the maximum and minimum distances are employed to obtain scattered representative points, which generates a novel prototype selection methods PSCURE (improved Prototype Selection algorithm based on CURE algorithm). Some numerical experiments are further conducted to show the performance of the proposed prototype selection algorithm compared with other related prototype selection algorithms. The experimental results show that the proposed algorithm not only can select fewer prototypes but also can achieve higher classifier accuracy for almost all the data sets.

Key words: K nearest neighbor classifier; prototype selection; shared neighbor density; CURE; representative point

1　引言

K 近邻分类器 (KNN)[1]是用于分类任务的最常用和最著名的技术之一, 其简单易操作性被人们广泛应用于文本分类、图像处理、手写数字识别等方面. 然而 K 近邻分类器存在计算量大、内存开销大的特点, 目前已经有许多改进方法. 其中最主要的技术有两

① 收稿时间: 2019-01-23; 修改时间: 2019-02-26; 采用时间: 2019-03-04; csa 在线出版时间: 2019-08-08

© 中国科学院软件研究所 http://www.c-s-a.org.cn

种: 原型生成和原型选择. 原型生成 PG (Prototype Generation)[2]是通过生成人造数据来代替原始数据, 使其能够填充原始数据集中没有代表性样例的区域. 原型选择 PS (Prototype Selection)[3]是对整个数据集进行约简, 在保证不降低甚至提高分类精度等性能的基础上, 获取具有较高分类贡献的同时能够反映原始数据集的分布状况具有一定代表性的原型子集.

目前, 研究学者已经提出了许多原型选择算法, 最著名的两个方法是压缩和剪辑策略. 在 1968 年, Hart[4]最早提出的原型选择算法是压缩最近邻 CNN (Condensed Nearest Neighbor), 该算法主要针对整个数据集, 尽可能多的约简数据集的样例, 只保留贡献率最高的边界样例, 但该算法分类精度却有所降低, 而且计算过程非常依赖原样例扫描的顺序. 在 1972 年, Tomek[5]提出了基于 K 近邻规则的剪辑算法 ENN(Edited Nearest Neighbor), 该算法旨在有效排除原始训练集中的噪声数据和重叠数据, 但该算法是基于剪辑策略的算法, 并没有删除对分类没有重要贡献的内部样例点, 因此其数据约简率比较低. 之后, 基于 ENN 的一系列改进算法被提了出来. 在 2009 年, Fayed 和 Atiya[6]提出了一种新的压缩算法—KNN 模板约简算法 TRKNN (Template Reduction for KNN), 该算法的基本思想是定义一个最近邻居“链”, 然后根据链上的每一段距离, 设置一个阈值来将其划分成保留的压缩数据集和要移除的数据集. 在 2010 年, Ougiaroglou[7]等提出了一个基于聚类的原型选择算法 PSC (Prototype Selection by Clustering), 该算法首先使用 K-means 聚类算法将整个数据集划分成不同的簇, 然后在每一个簇中选择代表样例. 对于同类簇集, 最靠近簇中心的样例被保留下来; 对于不同类簇集, 不同类边界的样例被保留下来; 保留下来的样例作为最终的原型进行分类. 在 2014 年, Li 和 Wang[8]提出了一个基于二叉最近邻树的原型选择算法 BNNT (the Binary Nearest Neighbor Tree), 该算法首先建立一个二叉最近邻树, 当二叉树位于一个类的内部时, 生成一个中心点代替树的所有节点被保留; 当二叉树位于多个类的边界时, 拥有不同类标签同时在树中直接相连的原型被保留下来, 保留下来的所有样例点作为原型集进行分类. 在 2016 年, Li[9]等人提出了基于自然邻居和最近敌人的原型选择算法, 该算法利用自然邻居过滤噪声模式并平滑类边界, 再使用基于最近敌人新的冷凝方法来减少原型的数量, 该编辑算法和冷凝算法的结合去除内部冗余样例, 保留了边界样例, 有效地减少了数据集的数量. 在 2017 年, 朱庆生[10]等人提出基于自然邻居和最小生成树的原型选择算法 2NMST (Prototype Selection Algorithm Based on Natural Neighbor and MST), 该算法使用自然邻居做数据预处理, 然后基于设定的终止条件构建 Prim 最小生成树, 生成一些具有代表性的内部原型, 并保留边界原型. 在 2018 年, 黄宇扬[11]等人提出了一种面向 K 最近邻 (KNN) 的遗传实例选择算法. 该算法采用基于决策树和遗传算法的二阶段筛选机制, 先使用决策树确定噪声样本存在的范围; 再使用遗传算法在该范围内精确删除噪声样本, 可有效地降低误删率并提高效率, 采用基于最近邻规则的验证集选择策略, 进一步提高了遗传算法实例选择的准确度; 最后引进基于均方误差 (MSE) 的分类精度惩罚函数来计算遗传算法中个体的适应度, 提高有效性和稳定性. 同年, 王熙照[12]等人提出基于非平稳割点的样例选择方法.依据在区间端点得到凸函数的极值这一基本性质, 通过标记非平衡割点度量一个样例为端点的程度, 然后选取端点程度较高的样例, 从而避免样例之间距离的计算. 同时, 进化算法已经应用于原型选择算法的, Acampora G[13]等人首次提出将“后验”算法, 即 SPEA2 应用于原型选择问题, 以明确处理 KNN 时间和空间复杂度两个目标, 并在分类和降低性能之间提供更好的权衡.

上述提到的所有算法虽然在存储空间和时间复杂度上都有了较好的改进, 但仍具有较低的分类准确率和较高的原型保留率. 针对目前原型选择存在的问题, 本文在 CURE 算法的基础上对其进行改进, 从而提出一种新的原型选择方法 PSCURE. 同时针对 CURE 聚类算法改进有两方面, 一方面是异常点剔除方法的改进. 原 CURE 算法很难确定异常点, 本文使用共享邻居密度计算每个原样例周围的密度, 再根据密度特征值进行去噪; 另一方面是代表点选择的改进. 原 CURE 算法挑选的代表点容易集中在图形最长的两端[14], 而本文使用最大最小距离来选取代表点, 使代表点具有分散性, 由此提出了一种新的原型选择算法. 本文的结构如下: 第 2 部分主要介绍共享邻居密度和最大最小距离, 以及 CURE 的相关知识; 第 3 部分提出基于 CURE 聚类改进的原型选择方法及计算; 第 4 部分针对所提出算法 PSCURE 进行数值实验.

© 中国科学院软件研究所 http://www.c-s-a.org.cn

2 相关工作

为了更好的理解本文算法, 本章介绍了涉及到的 3 种重要算法: 共享最近邻密度, 最大最小距离和 CURE 聚类算法.

2.1 共享邻居密度

共享邻居是两个样例数据共同拥有的近邻个数, 基本思想是如果两个样例点拥有的近邻个数越多证明两点越相似, 根据共享邻居的相似性得知每个点的局部密度, 即共享邻居密度.

定义 1[15](共享邻居). 对于数据集 X 中的任意点 i和j, 点i和点j的 K-最近邻居集合分别是$\Gamma(i)$和$\Gamma(j)$, 点i和点j的共享邻居是它们的公共邻居集, 表示为

$$SNN(i,j)=\Gamma(i)\cap\Gamma(j) \tag{1}$$

定义 2[15](SNN 相似性). 对于数据集 X 中的任何点 i和j, 它们的 SNN 相似性定义为

$$Sim(i,j)=\begin{cases}\dfrac{|SNN(i,j)|^2}{\sum\limits_{p\in SNN(i,j)}\left(d_{ip}+d_{jp}\right)}, & \text{if } i,j\in SNN(i,j)\\ 0, & otherwise\end{cases} \tag{2}$$

其中, d_{ij}是点i和j之间的距离. 仅当点i和点j出现在彼此的 K 邻居集中时才计算 SNN 相似性, 否则, 点i和点j之间的 SNN 相似性为 0. 式 (2) 的非零部分通过以下形式表示, 从中可以清楚地观察到 SNN 相似性的含义, 即:

$$Sim(i,j)=|SNN(i,j)|\times\frac{1}{\frac{1}{|SNN(i,j)|}\sum\limits_{p\in SNN(i,j)}\left(d_{ip}+d_{jp}\right)} \tag{3}$$

等式右端的左边部分表示点i和点j的共享邻居的数量, 右边部分是从点i和点j到所有共享邻居的距离的平均值的倒数, 它表示在一定程度上围绕两点的密度. 通过同时检查共享邻居和此两点的密度, SNN 相似性可以更好地适应各种环境.

在定义任意两点的 SNN 相似性之后, 我们使用该相似性来计算点i的局部密度ρ_i.

定义 3[15](SNN 局部密度). 设点i是数据集 X 中的任意点, $L(i)=\{x_1,x_2,\cdots x_k\}$是与点$i$具有最高相似度的 K 个点的集合. 点i的局部密度定义为与点i具有最高相似度的 K 个点的相似度之和, 即

$$\rho_i=\sum_{j\in L(i)}Sim(i,j) \tag{4}$$

基于共享邻居密度包含了数据点的局部邻域信息的特点, 当一个点邻域中的数据点分布较密时, 该点的密度就相对较大, 反之较小.

根据上述定义计算数据集中的每个数据对象的共享最近邻密度值, 然后按照共享邻居密度值对数据对象进行升序排列, 画出密度递增曲线. 根据假设, 数据集中的噪声点与正常点相比分布相对稀疏, 因而其密度较小, 那么在密度递增曲线上, 噪声点和正常点之间存在一个拐点, 在该拐点之前的数据点为噪声点, 而在此之后的点为正常点. 利用共享邻居密度去噪算法的伪代码如下:

算法 1: 利用共享邻居密度去噪的算法

输入: 数据集 X, 近邻数 K
输出: 去噪后的数据集

找出所有点的 K-最近邻
For ∀*i*, *j*∈*X*
If 两个点 *i* 和 *j* 不是互相在对方的 K 最近邻中 Then
number(*SNN*(*i*,*j*))←0
Else
number(*SNN*(*i*,*j*))←共享的近邻个数
End If
similarity(*i*, *j*)=*number*(*SNN*(*i*, *j*)) *(1/(1/*number*(*SNN*(*i*, *j*)) **sum*(*dist*(*i*, *p*))+*dist*(*j*, *p*)))
density=*sum*(*similarity*(*i*,*j*)) //局部密度ρ_i
End for
den_threshold=*computer_denthr*(*density*)
//remove noise points
For ∀*i*∈*X*
If *density*(*i*)<=*den_threshold*
X.*delete*(*i*)
End If
End for

在算法 1 中, *getKnn*(*i*,*X*) 表示在数据集 X 中寻找 i的 K 个近邻, *number*() 表示两个样例中的共同的邻居, 也就是共享邻居个数, 如果两个样例不是互相在对方的 K 最近邻中, 定义这两个样例的相似度为 0, 否则根据式 (1) 计算共享近邻个数. 通过式 (3) 计算共享近邻的相似性 *similarity*(), 再根据相似性通过式 (4) 求出数据集中每个样例的密度, 这里的密度主要是根据一个样例与周围 K 近邻的共享近邻的相似性之和定义, *computer_denthr*() 确定密度阈值, 从而去除数据集中的噪声点.

2.2 最大最小距离

最大最小距离算法也称小中取大距离算法[16]. 本

© 中国科学院软件研究所 http://www.c-s-a.org.cn

文使用欧式距离, 已知 N 个样本$X_1,X_2,\cdots,X_N$, 算法描述:

(1) 给定θ, $0<\theta<1$(一般$\theta=1/2$), 计算每个特征的平均值生成一个中心点, 计算离该中心点最近的数据集中的点作为第一代表点Z_1;

(2) 计算所有样例到Z_1的距离D_{i1}, 选择距离第一个代表点Z_1最远的样例点作为第二个代表点Z_2, 距离表示为D_{12};

(3) 计算其余样例点到Z_1, Z_2之间的距离, 并求出他们中的最小值, 即$D_i=\min(D_{i1},D_{i2})$, $i=1,2,\cdots,N$;

(4) 若$D_{i0}=\max_i(\min(D_{i1},D_{i2}))>\theta\cdot D_{12}$, 则相应的样例点$x_{i0}$作为第三个代表点$Z_3$;

(5) 以此类推计算每个样例点i到已确定的所有代表点j之间的距离D_{ij}, 并求出$D_{i0}=\max_i(\min(D_{i1},D_{i2},\cdots,D_{ij}))$, 若$D_{i0}>\theta\cdot D_{12}$, 则继续建立代表点, 否则结束寻找. 利用最大最小距离选取代表点算法的伪代码如下:

算法 2. 最大最小距离选取代表点

输入: 数据集 X, 阈值θ
输出: 代表点集合 *center*

```
point=[每个特征的平均值]
center[0]=距离 point 最近的点          //第一代表点
center[1]=距离 center[0]最远的点       //第二个代表点
D12=get_distance(center[0], center[1])
//计算所有代表点
max_min_distance=0
While max_min_distance>θ*D12
index=0
For i in range(len(X))
   min_distance=[]
   For j in range(len(center))
      distance=get_distance(X[i],center[j])
      min_distance.append(distance)
   End For
   min_dis=min(dis for dis in min_distance)
   If min_dis>max_min_distance
   max_min_distance=min_dis
   index=i
End if
   center.append(X[index])
End While
Return center
```

2.3 CURE 层次聚类算法

CURE (Clustering Using REpresentative) 算法[17]采用了一种自底向上的层次聚类算法, 最突出的特点是利用代表点而不是中心点代表一个类, 利用代表点计算一个类与另一个类之间的距离. 代表点是指能够代表这个类的形状、密度和分布的一些数据点, 基本的思想是: 所有数据点在最初都是一个簇, 然后不断合并, 直到最后的簇个数为指定的数目. 详细步骤如下:

(1) 抽样过程: 从原始数据对象中选取一个随机样例 D;

(2) 划分过程: 对样例 D 进行划分, 一般按数量均匀划分;

(3) 初始化过程: 设置参数, 参数一是要形成的簇数 K, 参数二是代表点 (代表该簇的点) 的数目 m, 参数三是收缩因子 *alpha*;

(4) 聚类过程: 对每个划分挑选代表点进行聚类; 第一个代表点选择离簇中心最远的点, 而其余点选择离所有已经选取的点最远的点;

(5) 剔除离群点: 第一阶段删除在聚类过程中增长非常缓慢的簇; 第二阶段在聚类结束时含数据对象明显少的簇.

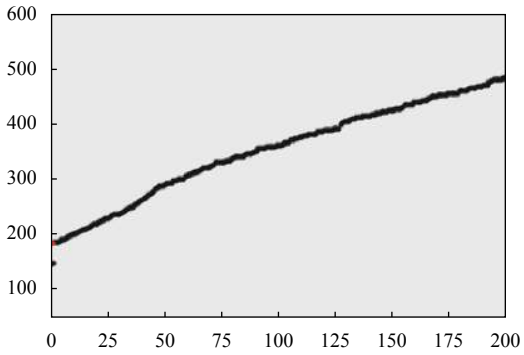
3 基于 CURE 聚类算法改进的原型选择

基于聚类的原型选择的动机是快速执行通过聚类挑选出有代表性的内部点和边界点, 为了实现这一点, 本文针对 CURE 层次聚类算法中对噪声点不易判断性, 使用了共享邻居密度计算每个样例的密度, 根据密度增值曲线对整个样例集确定密度阈值, 进而确定噪声点.

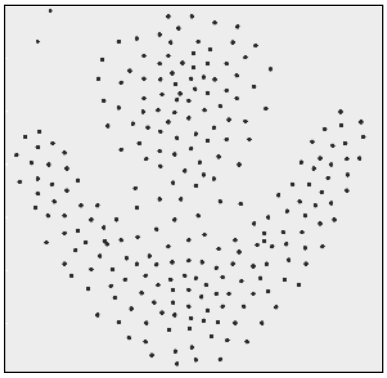
在此使用人造数据集 Flame people[18]对此算法进行说明. 图 1(a) 中表示该人造数据集的密度增值曲线, 其中红色断点处是我们确定的噪声点和正常点之间的拐点. 图 1(b) 中展示了利用算法 1 识别噪声点之后的结果, 其中“+”为识别出的噪声点.

另一方面, 针对 CURE 选择代表点的局限性, 利用最大最小距离选择代表点的方法代替 CURE 算法挑选代表点的算法, 使得代表点能够尽量分散, 从而更好地代表该簇. 在此利用随机产生的数据说明该情况. 图 2(a) 是利用 CURE 算法挑选出代表点的图示, 其中圆圈点为代表点, 可见代表点集中在两端, 不能很好地反应数据的形状和分布信息. 图 2(b) 中是利用算法 2 挑选的代表点, 可以观察到, 代表点分散在每个边缘地方, 并选择了适当的中心点来做代表点, 满足分散点的特性, 较好的描述了数据集的分布情况.

© 中国科学院软件研究所 http://www.c-s-a.org.cn

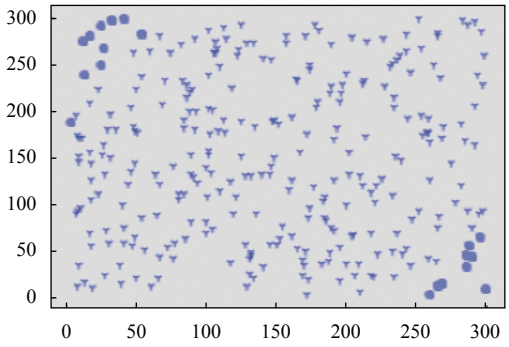


(a) 密度增值曲线

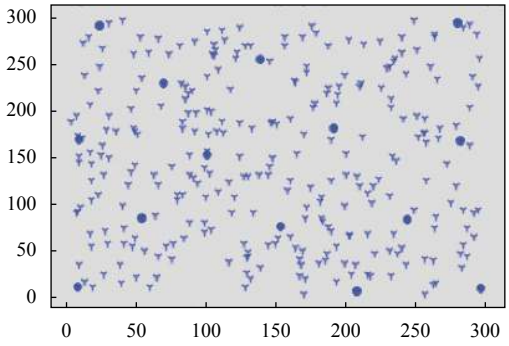


(b) 识别噪声点后的结果

图 1 利用共享邻居密度确定噪声点



(a) 原算法挑选代表点



(b) 最大最小距离挑选代表点

图 2 CURE 与最大最小距离代表点选取结果

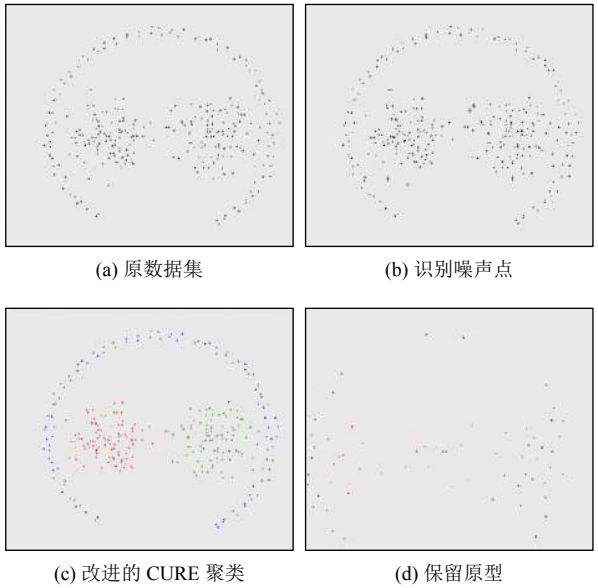
PSCURE 算法的主要步骤如下:

(1) 针对原始样例集$S=\{X_1,X_2,\cdots,X_N\}$使用算法 1 对原始样例集S进行去噪处理, 得到去噪后的样例集S';

(2) 针对样例集S'使用算法 2 替代 CURE 聚类算法步骤 (4) 挑选代表点的方法, 所得到的代表点生成最终原型集 PS;

(3) 使用最终的原型集 PS 进行 KNN 分类.

为了测试本文基于 CURE 算法改进的原型选择算法效果, 使用了合成数据集 pathbased[19]和 Flame people[18]进行评估. 图 3(a) 展示了 pathbased 原始数据集的分布情况, 图 3(b) 通过共享近邻的密度对整个样例集进行去噪处理, 其中图中的“+”表示应用该算法识别的噪声点, 直观地可以看出噪声点周围的密度相对其它数据集点的密度较低, 图 3(c) 通过最大最小距离改进的 CURE 层次聚类算法针对数据集进行聚类, 可以看出该算法将该数据集准确的划分为三个类, 保留其聚类完成后的所有代表点, 图 3(d) 所示, 该算法保留了一些靠近中心的样例点和簇边界点, 得到的原型最大程度的代表了整个数据集.



(a) 原数据集 (b) 识别噪声点

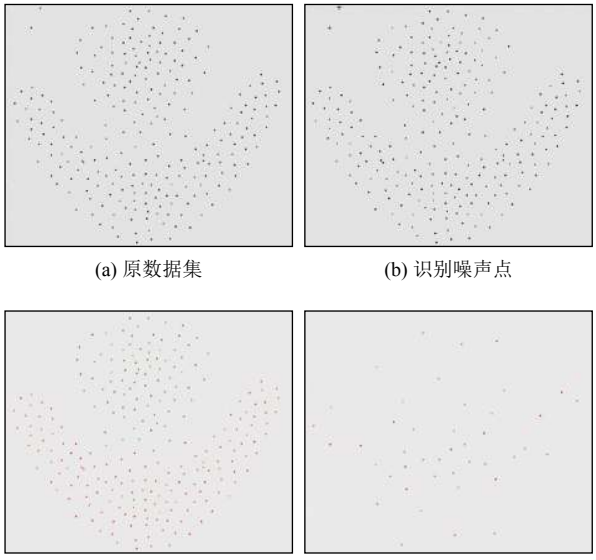
(c) 改进的 CURE 聚类 (d) 保留原型

图 3 PSCURE 原型选择算法在 Pathbased 数据集上的展示

同样, 图 4 展示了所提算法在合成数据集 Flame people 上的结果, 图 4(a) 展示了 Flame people 原始数据集的分布情况, 图 4(b) 图检测出局部密度最低的两个点作为噪声点, 即图中的“+”; 图 4(c) 通过改进的 CURE 聚类算法精确的识别该数据属于两个簇, 针对

© 中国科学院软件研究所 http://www.c-s-a.org.cn

两个簇保留聚类过程所用的代表点作为最后的原型集, 即图 4(d) 展示, 该代表点除了包含内部点之外, 还包含两簇之间的边界点.



(a) 原数据集 (b) 识别噪声点

(c) 改进的 CURE 聚类 (d) 保留原型

图 4 PSCURE 原型选择算法在 Flame people 数据集上的展示

4 实验结果及分析

为了评估 PSCURE 原型选择算法的有效性, 该章节利用该算法与传统的 KNN[1], 经典的 CNN[4]、ENN[5]和最新的 TRKNN[6]、PSC[7]、BNNT[8]和 2NMST[10]算法在原型选择的个数及分类准确率方面进行了实验比较. 为了达到这个目的, 从 UCI 上下载 10 个数据集, 数据集的描述如表 1 所示.

表 1 UCI 数据集

数据集	数据规模	特征数	类别数
Iris	150	4	3
Wine	178	13	3
Sonar	208	60	2
Glass	214	9	6
Liver	345	6	5
Yeast	1484	8	10
Segmentation	2100	19	7
Spambase	4601	57	10
Pendigits	10 992	16	10
Letter	20 000	16	26

4.1 实验设计

本文使用 10 折交叉验证将数据集划分为训练集和测试集, 取平均值作为最终的测试结果. 在 KNN 实验中, 选择 K=3、5、7 中分类准确率最高的值作为最终分类准确率, 距离度量采用的是欧氏距离. 为了精确度量算法的性能, 本文使用分类准确率和样例的保留率两个主要的考察指标. 分类准确率表示针对测试样例集利用分类器正确分类的样例数与测试集总样例数的比值. 保留率表示训练数据集经过该算法处理之后保留下来的数据子集与训练集中样本个数的比值. 为了方便我们把这两个指标分别记为 Acc 和 Str. 公式分别如下:

$$Acc=\frac{|P_r|}{n_1}\times 100\%\ ,\ Str=\frac{|PS|}{n_2}\times 100\%$$

其中, n_1表示测试集个数, n_2表示训练集个数, $|P_r|$表示被正确分类的样例个数, $|PS|$表示算法最后保留下来的样例集个数.

4.2 实验结果

针对表 1 给出的 UCI 数据集中的数据, 本文首先利用 PSCURE 算法与传统的 KNN 算法和基于聚类的原型选择 PSC 做比较, 表 2 列出了这 3 个算法的准确率和保留率, 同样, 图 5 展示准确率和保留率之间的比较, 从实验结果可以看出 KNN 算法对于整个样例集没有进行约简, 因此保留率是 100%, 在没有缩减的情况下准确率为 77.65%. PSC 算法跟传统的 KNN 算法进行比较, 降低了样例数量, 平均保留率为 38.67%, 但在准确率上较传统的 KNN 算法低于 10.91%. PSCURE 算法的准确率为 81.66%, 高于 PSC 算法的同时, 更高于传统的 KNN, 但 PSCURE 算法的平均保留率仅有 16.97%, 大大约简了数据, 提高了效率. 因此 PSCURE 算法有效的提高了分类算法的准确率和原型缩减率.

为了更进一步验证 PSCURE 算法的有效性, PSCURE 算法与 CNN, ENN, TRKNN, BNNT 相比 (见表 3), PSCURE 算法对几乎所有数据集有更高的或差不多的分类准确率, 但却有较低的保留率. 与 2NMST 相比, PSCURE 算法在数据集 Sonar、Wine、Glass、Yeast、Segmentation、Letter 上有较高的分类准确率且有较低的保留率. 从图 6 可以看出, 本文所提算法的平均分类准确率最高, 但平均保留率最低. 从平均结果来看, PSCURE 算法在样本保留率和分类准确率方面都有明显的优势.

5 结束语

KNN 在大规模数据集下具有过高的时间和空间

© 中国科学院软件研究所 http://www.c-s-a.org.cn

复杂度而限制了其应用, 为了解决该问题, 本文提出了基于 CURE 聚类算法改进的原型选择, 该算法可以保证在分类准确率不降低情况下, 缩减原始数据集样例数, 从而提高分类效率. 本文在 CURE 算法基础上进行改进, 挑选代表点来对 KNN 进行原型选择. 具体地, 使用共享邻居密度对整个样例集进行噪声点处理, 使用最大最小距离代替 CURE 聚类算法代表点的选择, 最后利用挑选的代表点集进行 KNN 分类. 实验结果表明 PSCURE 算法比其他原型选择算法能筛选出较少的原型, 但能获取较高的分类准确率.

表 2 PSCURE 算法与 KNN、PSC 的准确率和保留率

Database	KNN		PSC		PSCURE	
	Acc	Str	Acc	Str	Acc	Str
Iris	**95.84**	100	92.72	31.67	95.78	**20.22**
Wine	67.61	100	51.93	36.52	**98.48**	**17.61**
Sonar	78.70	100	63.98	55.59	**80.21**	**28.82**
Glass	62.47	100	57.37	48.78	**63.37**	**23.72**
Liver	54.92	100	46.22	42.39	**65.84**	**7.98**
Yeast	55.88	100	48.42	27.21	**56.92**	**19.23**
Segmentation	**94.03**	100	81.71	50.09	**94.03**	**10.2**
Spambase	**73.51**	100	68.63	42.21	**71.98**	**14.98**
Pendigits	**98.29**	100	82.67	28.75	94.83	**10.98**
Letter	**95.24**	100	73.73	23.51	95.2	**15.96**
average	77.65	100	66.74	38.67	**81.66**	**16.97**

表 3 算法 PSCURE 和 CNN、ENN、TRKNN、BNNT、2NMST 的准确率和保留率的比较结果

Database	CNN		ENN		TRKNN		BNNT		2NMST		PSCURE	
	Acc	Str	Acc	Str	Acc	Str	Acc	Str	Acc	Str	Acc	Str
Iris	88.67	51.67	88.34	70.83	92.08	58.83	94.67	**22.08**	**96.00**	19.17	95.78	20.22
Wine	67.42	67.42	69.40	67.77	63.23	57.94	55.56	23.17	97.76	17.69	**98.48**	**17.61**
Sonar	61.08	46.27	74.76	58.59	56.47	56.19	62.98	40.56	65.35	**28.48**	**80.21**	28.82
Glass	58.87	65.42	53.52	49.36	54.10	63.08	46.49	26.29	63.13	**23.46**	**63.37**	23.72
Liver	45.75	58.33	46.87	91.49	48.02	43.84	46.47	49.82	**66.09**	19.42	65.84	**7.98**
Yeast	46.09	37.82	44.51	26.28	**75.96**	41.23	47.17	19.12	56.48	**18.97**	56.92	19.23
Segmentation	**94.42**	74.70	77.06	16.43	58.49	39.73	86.87	19.67	85.14	20.92	94.03	**10.2**
Spambase	55.31	22.69	46.04	40.25	76.54	35.79	68.76	35.36	**74.59**	**8.88**	71.98	14.98
Pendigits	**98.43**	32.06	84.33	21.24	80.05	39.53	95.39	17.12	96.32	**9.28**	94.83	10.98
Letter	57.63	29.15	54.42	43.73	68.32	24.19	83.91	24.92	84.80	29.10	**95.2**	**15.96**
average	67.37	48.55	63.92	48.60	67.33	46.03	68.83	27.81	78.57	19.54	**81.66**	**16.97**

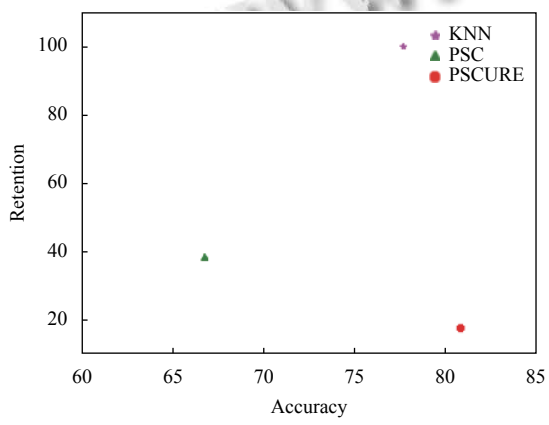


图 5 表 2 中平均准确率和保留率的散点图

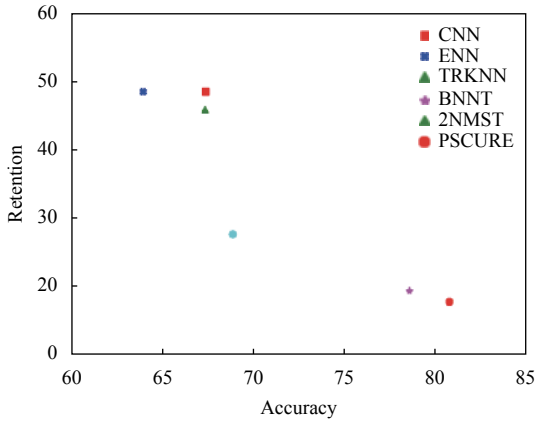


图 6 表 3 中平均准确率和保留率的散点图

© 中国科学院软件研究所 http://www.c-s-a.org.cn

参考文献

1 Cover T, Hart P. Nearest neighbor pattern classification. IEEE Transactions on Information Theory, 1967, 13(1): 21–27. [doi: 10.1109/TIT.1967.1053964]

2 Triguero I, Derrac J, Garcia S, *et al*. A taxonomy and experimental study on prototype generation for nearest neighbor classification. IEEE Transactions on Systems, Man, and Cybernetics, 2012, 42(1): 86–100. [doi: 10.1109/TSMCC.2010.2103939]

3 García S, Derrac J, Cano J, *et al*. Prototype selection for nearest neighbor classification: Taxonomy and empirical study. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2012, 34(3): 417–435. [doi: 10.1109/TPAMI.2011.142]

4 Hart P. The condensed nearest neighbor rule (Corresp.). IEEE Transactions on Information Theory, 1968, 14(3): 515–516. [doi: 10.1109/TIT.1968.1054155]

5 Tome K I. An experiment with the edited-nearest neighbor rule. IEEE Transactions on Systems, Man, and Cybernetics, 1976, SMC-6(6): 448–452. [doi: 10.1109/TSMC.1976.4309523]

6 Fayed HA, Atiya AF. A novel template reduction approach for the *K*-nearest neighbor method. IEEE Transactions on Neural Networks, 2009, 20(5): 890–896. [doi: 10.1109/TNN.2009.2018547]

7 Ougiaroglou S, Evangelidis G. Fast and accurate k-nearest neighbor classification using prototype selection by clustering. Proceedings of 2012 Panhellenic Conference on Informatics. Piraeus, Greece. 2012. 168–173.

8 Li J, Wang YP. A new fast reduction technique based on binary nearest neighbor tree. Neurocomputing, 2015, 149: 1647–1657. [doi: 10.1016/j.neucom.2014.08.028]

9 Yang LJ, Zhu QS, Huang JL. An efficient reduction algorithm based on natural neighbor and nearest enemy. Proceedings of 2016 IEEE International Conference on Cognitive Informatics & Cognitive Computing. Palo Alto, CA, USA. 2016. 212–218.

10 朱庆生, 段浪军, 杨力军. 基于自然邻居和最小生成树的原型选择算法. 计算机科学, 2017, 44(4): 241–245, 268. [doi: 10.11896/j.issn.1002-137X.2017.04.051]

11 黄宇扬, 董明刚, 敬超. 面向 K 最近邻分类的遗传实例选择算法. 计算机应用, 2018, 38(11): 3112–3118. [doi: 10.11772/j.issn.1001-9081.2018041337]

12 王熙照, 邢胜, 赵士欣. 基于非平稳割点的大数据分类样例选择. 模式识别与人工智能, 2016, 29(9): 780–789.

13 Acampora G, Tortora G, Vitiello A. Applying SPEA2 to prototype selection for nearest neighbor classification. Proceedings of 2016 IEEE International Conference on Systems, Man, and Cybernetics. Budapest, Hungary, 2016. 3924–3929.

14 邱荣财. 基于 spark 平台的 CURE 算法并行化设计与应用 [硕士学位论文]. 广州: 华南理工大学, 2014. 6.

15 Liu R, Wang H, Yu XM. Shared-nearest-neighbor-based clustering by fast search and find of density peaks. Information Sciences, 2018, 450: 200–226. [doi: 10.1016/j.ins.2018.03.031]

16 李金宗. 模式识别导论. 北京: 高等教育出版社, 1994.

17 Guha S, Rastogi R, Shim K. Cure: An efficient clustering algorithm for large databases. Information Systems, 2001, 26(1): 35–58. [doi: 10.1016/S0306-4379(01)00008-4]

18 Chang H, Yeung DY. Robust path-based spectral clustering. Pattern Recognition, 2008, 41(1): 191–203. [doi: 10.1016/j.patcog.2007.04.010]

19 Fu LM, Medico E. FLAME, a novel fuzzy clustering method for the analysis of DNA microarray data. BMC Bioinformatics, 2007, 8: 3. [doi: 10.1186/1471-2105-8-3]

© 中国科学院软件研究所 http://www.c-s-a.org.cn