文章编号:1671-9352(2025)01-0014-15 DOI:10.6040/j.issn.1671-9352.4.2023.0212

融合矩阵分解和空间划分的微生物数据扩增方法

温柳英,吴俊,闵帆
(西南石油大学计算机与软件学院, 四川 成都 610500)

摘要:针对微生物数据类内和类间不平衡、高稀疏性的问题,提出一种融合矩阵分解和空间划分的数据扩增算法。采用矩阵分解技术将原始数据空间分解为对象子空间和特征子空间,提取潜在空间表示,对象子空间划分为多个数据子空间,缓解了类内不平衡问题。为了解决类间不平衡问题,在每个数据子空间中生成合成样本,使用欧氏距离对合成样本进行过滤,获得高质量的样本。在 9 个微生物数据集上实验,再与 9 个采样算法进行性能对比。结果表明,本文算法生成的样本在多样性上具有较大优势,采用多个分类器时,能识别出更多的阳性样本。

关键词:矩阵分解;空间划分;类内不平衡;类间不平衡;对象子空间;特征子空间
中图分类号:TP391 **文献标志码**:A
引用格式:温柳英,吴俊,闵帆. 融合矩阵分解和空间划分的微生物数据扩增方法[J]. 山东大学学报(理学版),2025,60(1):14-28,44.

Fusing matrix factorization and space partition microbial data augmentation algorithm

WEN Liuying, WU Jun, MIN Fan
(School of Computer and Software, Southwest Petroleum University, Chengdu 610500, Sichuan, China)

Abstract: Aiming at the problems of intra-class imbalance and inter-class imbalance and high sparsity of microbial data, a data augmentation method that fuses matrix factorization and space partition is proposed. Matrix factorization technology is used to decompose the original data space into object subspace and feature subspace to extract the latent space representation. The object subspace is divided into multiple data subspaces to alleviate the intra-class imbalance problem. Synthetic samples are then generated in each data subspace to address the inter-class imbalance. Synthetic samples are filtered using Euclidean distance to obtain high-quality samples. The experiment is conducted on 9 microbial data sets, and the performance is compared with 9 sampling algorithms. The results show that the samples generated by the proposed method have great advantages in diversity, and more positive samples can be identified under multiple classifiers.

Key words: matrix factorization; space partition; intra-class imbalance; inter-class imbalance; object subspace; feature subspace

0 引言

微生物组技术已经广泛应用于疾病的检测和预测中,如苯丙酮尿症、肠道疾病和癫痫等疾病等[1-3]。操作分类单元(operational taxonomic units, OTU)可以通过高通量测序获得。研究人员对微生物组 OTU 的分析实验揭示了微生物组与人类健康或疾病之间的关系[4]。这类微生物数据呈现显著的类别不平衡特征,即阳性、阴性样本比例严重失衡,例如:阿尔茨海默病阳性样本与阴性样本的比例高达 1∶195[5],传统分类器[6-7]在分类准确性方面都能够取得较好的效果,即使将所有阳性样本分为阴性样本,准确率也能达到 95%以上,然而,对于疾病检测来说,误诊对患者的影响很可能是致命的,须要尽可能地识别更多的患病样本,即提升阳性的召回率[8-9]。这项任务存在着一定的挑战性,原因有以下几点:(1)微生物数据高稀疏性,即数据

收稿日期:2023-05-18;**网络出版时间**:2024-02-28 21:09:37
基金项目:中央引导地方科技发展专项资助项目(2021ZYD0003)
第一作者:温柳英(1983—),女,副教授,博士,研究方向为机器学习、不平衡学习和微生物信息学. E-mail: wenliuying1983@ 163.com

矩阵中存在着大量的零值;(2)类别不平衡,且以类间不平衡和类内不平衡混合的形式存在。类间不平衡指少数类与多数类样本数量不平衡,类内不平衡指类内出现子聚集现象,主要是由同类样本在特征空间分布不均所致;(3)高维特性,即微生物数据具有庞大的 OTU 特征属性。

解决类间不平衡的方法主要有欠采样、过采样和合成采样技术。欠采样技术[10]是为了减少大多数类样本以平衡数据集,但它会导致一些信息的丢失。过采样技术[11]是增加少数类样本以平衡数据集,但它可能导致信息冗余和模型过拟合。合成少数过采样技术(synthetic minority oversampling technique, SMOTE)[12]是平衡数据集最有效的技术之一,将随机采样和过采样方法相结合,从已有的数据样本中生成新的样本,但该算法仍有以下缺点:(1)它可能会放大噪声,没有关注样本的分布,选择样本时存在一些盲目性;(2)SMOTE 生成的新样本可能是重复的,导致过度拟合;(3)由于 SMOTE 算法以线性方式创建一个新样本,因此无法克服少数群体内部的不平衡问题,即类内不平衡。

聚类合成少数类过采样技术(*k*-means synthetic minority oversampling technique, KMeans_SMOTE)[13]和代表合成少数过采样技术(clustering using representatives synthetic minority oversampling technique, CURE_SMOTE)[14]是目前较为有效的采样方法。前者避免了生成有噪声的样本,消除了类内和类间的不平衡,然而,这些技术只考虑了数据不平衡的问题,忽略了原始微生物数据集的高稀疏性[5,15]特点。矩阵分解技术[1]能够将原本复杂的数据模型分解成 2 个简单的低秩矩阵。矩阵分解技术可以有效地降低原始数据的稀疏性,但不能解决类内不平衡。对数据模型进行操作前,先降低数据集的稀疏性对实验结果有着进一步的提升。空间划分是缓解类内不平衡的一种技术,利用递归思想对数据空间进行划分,在空间划分后的数据空间中,能够有效地识别少数类样本并采用过采样技术使得达到类间和类内平衡。

本文针对微生物数据的高稀疏、混合不平衡问题,提出了一种融合矩阵分解和空间划分的数据扩增(fusion matrix factorization and space partitioning for data augmentation, MFSP)算法。MFSP 算法由矩阵分解、空间划分、数据扩增和数据过滤 4 个阶段组成,算法结构如图 1 所示(图中 $\boldsymbol{D}$ 为 $m\times n$ 的数据矩阵,$\boldsymbol{S}$ 为对象子空间,$\boldsymbol{F}$ 为特征子空间,$\boldsymbol{S}_{\text{new}}$为合成向量矩阵,$\tau$ 为负样本所占总样本比例,b 为正样本数量,$\boldsymbol{S}'$为新对象子空间,$\boldsymbol{D}'$为 $m'\times n$ 的新数据矩阵)。

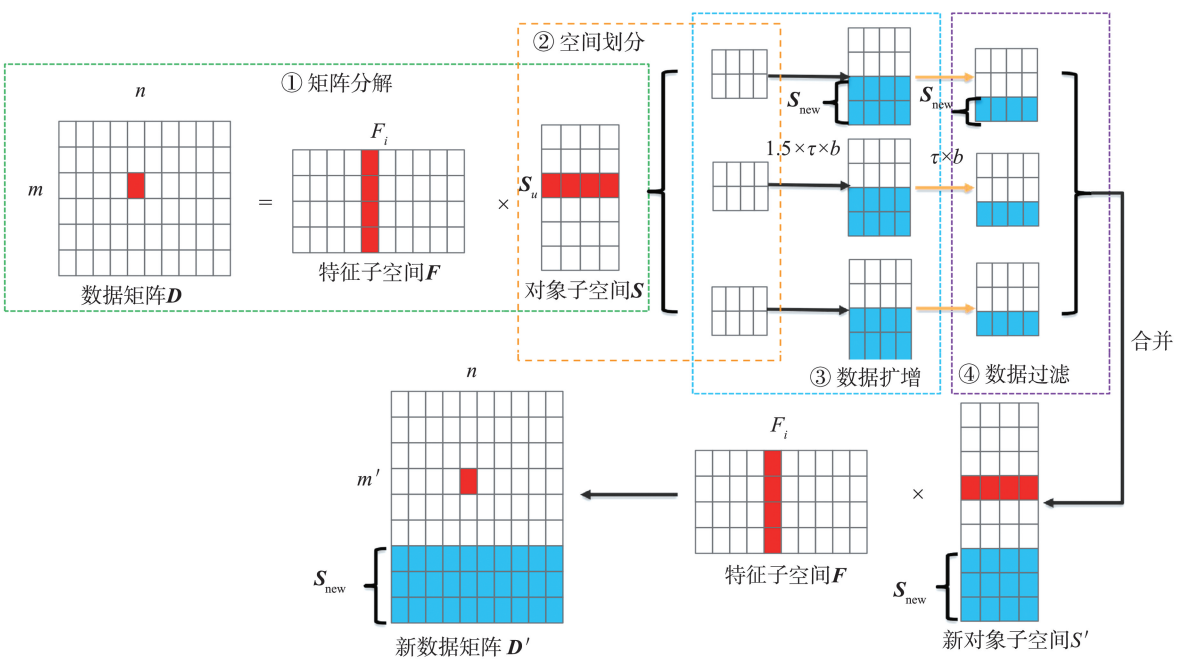


图 1　MFSP 算法框架

Fig.1　MFSP algorithm framework

矩阵分解阶段。使用矩阵分解算法将 $m\times n$ 的数据矩阵划分为对象子空间和特征子空间,该过程将原始矩阵转换为 2 个低秩矩阵进行处理,使得复杂的建模过程能够得到有效的简化,有效提取数据的隐含空间表示。

空间划分阶段。使用空间划分技术将对象子空间划分为若干数据子空间,该过程可以进一步将数据矩

阵简化,便于接下来针对不同的数据子空间给予有效的处理。同时,空间的有效划分,在一定程度上缓和类内出现的子聚集不平衡问题。

数据扩增阶段。对空间划分后的每一个数据子空间进行数据扩增。各个数据子空间扩增的样本数量为$1.5\times\tau\times b$,能够针对各个数据子空间的不同特性给予有效的扩增。

数据过滤阶段。对合成矢量进行过滤,用于选择质量好的$\tau\times b$个合成向量,并且直接丢弃质量差的合成向量,形成新对象子空间$\boldsymbol{S}'$。最终$\boldsymbol{S}'\times\boldsymbol{F}$,得到$m'\times n$的新数据矩阵$\boldsymbol{D}'$。

为了评估MFSP算法的性能,在9个具有不同平衡比的微生物数据集上进行实验,并与其他9种算法进行对比。结果表明,MFSP算法既能合成质量较高的样本,也能识别出更多的疾病样本。

本文利用矩阵分解技术将原始数据空间分解为对象和特征子空间,提取隐含l维空间,该方法缓解数据高稀疏性问题。基于近似最近邻搜索(approximate nearest neighbors oh yeah, Annoy)算法[16],提出了矩阵近似最近邻搜索(matrix approximate nearest neighbors oh yeah, Mannoy)新空间划分方法。Mannoy算法采用了一种新的空间划分点的选择,使数据样本在空间划分后的分布相对均匀,并且使得划分空间更加稳定,从而缓解了类内不平衡问题。在空间划分后的数据空间中生成合成样本,能够有效解决类间的不平衡问题。设计了一种新的数据过滤方案,该方案同时考虑了类间距离和类内距离,可以有效地过滤质量差的样本,保留质量较好的样本。

1 相关工作

1.1 数据模型

微生物群落计数数据(OTU计数、类群丰度)存在以下几个研究难题:(1)微生物组序列数据集是具有数万个不同类别的高维数据集,不能鉴定分类群或OTUs的数量远远大于样本的数量。例如,病人群体很小,描述性特征很多,导致建模的困难。(2)分类群计数数据过度分散,无论是来自微生物组研究中扩增子测序实验的分类或OTU计数,还是来自RNA测序实验[17]的差异表达数据,通常都是过度分散的。(3)在微生物组数据中,稀疏性即在样本中缺少很多分类群,并且在大多数实验[1]中都会产生零点。

1.2 矩阵分解

矩阵分解是将复杂矩阵分解成简单矩阵的乘积形式。奇异值分解(singular value decomposition, SVD)是矩阵分解的一个主要分支[18],通常用于处理密集的数据矩阵。微生物数据极其稀疏,若使用SVD对微生物数据进行分解,须要预先对缺失值进行填充,但缺失值填充后会与已知数据混淆,不能区分开,导致算法复杂度增加且数据可能失真。由于SVD求解过程存在一些问题,因此实际场景中一般不用SVD,而是采用数值迭代和梯度下降的方式,实现矩阵分解。通过运用数值迭代和梯度下降的方法Basic-SVD[19]、Funk-SVD[20]、SVD++[21]矩阵分解方法被相继提出,不再将矩阵分解为3个矩阵,而是分解为2个低阶矩阵,同时降低了计算复杂度。

1.3 空间划分

空间划分相关研究都是基于最近邻居搜索进行的。二进制树和位置敏感哈希算法[22]已经被广泛用于解决最近邻搜索问题。这2项技术都是基于空间划分的技术。二进制空间划分树,包括k-维树(k-dimension tree, KDT)[23]、共享树(rendezvous point tree, RPT)等[24]。研究表明,具有矢量量化性能的树有更好的搜索性能。关联搜索检索信息通常采用KDT树这类数据结构,然而KDT树在处理高维数据时,性能随维度增加而下降。为了弥补这一差距,现有的RPT树克服了KDT树的不足。RPT树能自动适应内在的低维结构,不必显式地学习数据集结构。Annoy算法是一个高维空间求近似最近邻的算法,使用类似RPT树的思想来划分原始空间,最终建立一棵最近邻查找的二叉树。通过随机选择2个点,Annoy算法利用与这2个点垂直的等距超平面将数据空间划分为2个子空间,再递归的划分空间。

1.4 采样技术

采样技术是通过对训练集进行处理使其从不平衡的数据集变成平衡的数据集。采样分为过采样和欠采样。过采样是把少数类复制多份。过采样后的数据集中会反复出现一些样本,训练出来的模型会有一定的过拟合[11]。欠采样是从大众类中剔除一些样本,或者说只从大众类中选取部分样本。欠采样的缺点是最终

的训练集丢失了数据,模型只学到了总体模式的一部分,与过采样一样,欠采样也存在过拟合的问题。得益于 SMOTE 的成功,研究人员在 SMOTE 上做出了许多改进,如密度、边界线、聚类等[11]。基于遗传的分类算法改进的合成少数过采样技术(genetic algorithm improve synthetic minority oversampling technique, GASMOTE)[25]针对不同的少数类样本实例采用不同的采样率,最终找到一个最佳的采样率组合。KMeans-SMOTE 是一种基于聚类的过采样方法,能够有效的避免噪声的产生,并且能够克服类内和类间不平衡。融合矩阵分解和代价敏感的数据扩增(matrix factorization combined with cost-sensitive sampling, MFCS)算法[1],能够有效解决数据高稀疏性的问题,但它没有考虑类内不平衡的问题。基于核主成分分析(kernel principal component analysis, KPCA)的 integrating KPCA and generative adversarial network augmentation 算法[26],即 KGA 算法,通过构建数据的有效特征空间来解决类别不平衡问题。

2　MFSP 算法

本章将详细介绍融合矩阵分解和空间划分的 MFSP 算法。算法由矩阵分解技术、空间划分技术、数据扩增技术和数据过滤技术组成。

2.1　矩阵分解

为了解决微生物数据的高稀疏性,本文使用矩阵分解的方法将复杂的原始数据矩阵分解为对象子空间和特征子空间,有效减少空间的复杂性,并能够提取隐藏因子 l。矩阵分解公式[27]为

$$\boldsymbol{D}_{m\times n}=\boldsymbol{S}_{m\times l}\boldsymbol{F}_{n\times l}^{\mathrm{T}}, \tag{1}$$

式中,$\boldsymbol{D}$ 表示原始数据矩阵,m 是数据集中的样本数,n 是特征数量,$\boldsymbol{S}$ 是对象子空间,$\boldsymbol{F}$ 是特征子空间,l 是隐藏因子的数量。

矩阵分解的损失函数[27]为

$$\min_{\boldsymbol{S}^*,\boldsymbol{F}^*}\left\{\sum_{u=1}^{m}\sum_{i=1}^{n}\frac{|\boldsymbol{D}_{ui}-\boldsymbol{S}_u\boldsymbol{F}_i^T|}{m\times n}\right\}, \tag{2}$$

式中,$\boldsymbol{D}_{ui}$表示 $\boldsymbol{D}$ 的第 u 行第 i 列的值,$\boldsymbol{S}_u$ 指 $\boldsymbol{S}$ 的第 u 个向量,$\boldsymbol{F}_i$ 指 $\boldsymbol{F}$ 的第 i 个向量。

2.2　空间划分

在对数据进行扩增前,先将对象子空间进行空间划分。本文提出的 Mannoy 算法是在 Annoy 算法的基础上进行改进的。Annoy 算法通过当前空间中随机选择的 2 个点划分空间,Mannoy 算法是通过选取密度最大和最小的 2 个向量,在 2 个向量之间建立超平面,将空间一分为二进行空间划分,通过密度的选点方式递归的划分空间,当划分的数据子空间中的样本数量少于设定阈值时则停止划分。

划分点的选择。首先通过计算每一个样本的 k 近邻距离半径内的样本数量,记作该样本的密度 ρ。对样本密度 ρ 进行排序,取出前 60%的样本点作为待选取点 $\boldsymbol{M}$,取出 $\boldsymbol{M}$ 中的密度最大的样本 p 和密度最小样本 q 作为空间划分的划分点。

算法 1 为 Mannoy 算法空间划分的伪代码。第①—④步判断输入数据集是否小于数据子空间最少样本数量的阈值,第⑤—⑩步是选择空间划分所需的划分点,第⑪—⑮步将 p 和 q 的中点作为划分子空间的根节点,第⑯—⑱步通过划分点确定超平面,第⑲—㉘步递归的划分数据子空间,并建立一棵二叉树。

算法 1　Mannoy 算法。

输入　对象子空间 $\boldsymbol{S}$,树节点 Root,叶节点的大小 g。

输出　二叉树 $\boldsymbol{T}$。

① if $|\boldsymbol{S}|\leqslant g$ then
② Each sample in $\boldsymbol{S}$ is a leaf node;
③ return;
④ end if
⑤ for each sample x_i in $\boldsymbol{S}$ do
⑥ r=The average distance of the k nearest neighbor samples of x_i;
⑦ ρ=Calculate the number of samples in a circle with x_i as the center and r as the radius;
⑧ end for

⑨ Sort the data set $\boldsymbol{D}$ in descending order according to the density;
⑩ $\boldsymbol{M}$ = Top 60% samples of dataset $\boldsymbol{D}$;
⑪ p is the largest value in $\boldsymbol{M}$;
⑫ q is the smallest value in $\boldsymbol{M}$;
⑬ if Root is NULL then
⑭ Root = the middle point of p and q;
⑮ end if
⑯ right child of Root = p;
⑰ left child of Root = q;
⑱ h = the hyperplane equidistant from p and q;
⑲ $\boldsymbol{S}_r$ = the points lies on the right of h;
⑳ if $\boldsymbol{S}_r \neq \varnothing$ then
㉑ Mannoy($\boldsymbol{S}_r, p, g$);
㉒ end if
㉓ $\boldsymbol{S}_l$ = the points lies on the left of h;
㉔ if $\boldsymbol{S}_l \neq \varnothing$ then
㉕ Mannoy($\boldsymbol{S}_l, q, g$);
㉖ end if
㉗ $\boldsymbol{T}$ = Root;
㉘ return $\boldsymbol{T}$;

2.3 数据扩增

数据扩增技术通过增加少数类样本的数量来实现数据平衡,本文算法通过在空间划分后得到数据子空间,再利用增强函数生成合成向量,将扩增后的数据子空间合并,与特征空间相乘得到合成向量。扩增公式为

$$\boldsymbol{S}_{\text{new}_\alpha} = \beta \boldsymbol{S}_i + (1-\beta)\boldsymbol{S}_{\text{nearst}_i}, \tag{3}$$

式中,$\boldsymbol{S}_{\text{new}_\alpha}$表示合成向量,$\boldsymbol{S}_i$ 表示正向量,$\boldsymbol{S}_{\text{nearst}_i}$是 $\boldsymbol{S}$ 中与 $\boldsymbol{S}_i$ 相邻最近的正向量,它与 $\boldsymbol{S}_i$ 之间的欧氏距离最小,β 是[0,1]中的随机值。

2.4 数据过滤

决策边界附近的向量很容易被错误分类。为了获得质量更好的合成向量,在数据过滤过程中同时考虑了类内和类间距离。数据过滤过程如图 2 所示。图 2 中 $\boldsymbol{S}_+$表示正向量集合,$\boldsymbol{S}_-$表示负向量集合,$\boldsymbol{S}_{\text{new}}$表示合成向量集合,蓝色表示正样本,橙色表示负样本,红色表示合成样本。

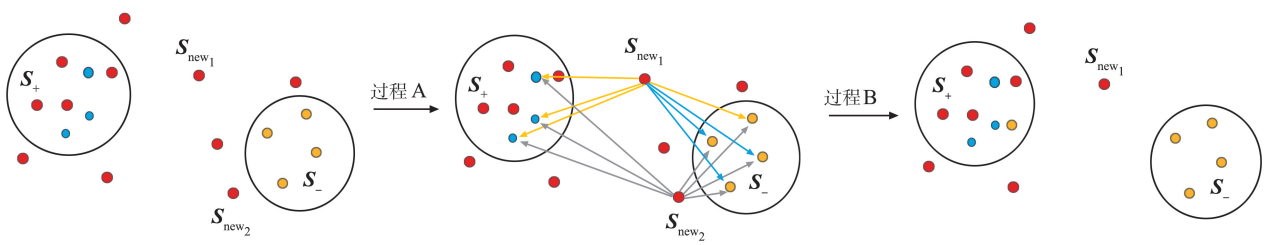


图 2 数据过滤
Fig.2 Data filtering

图 2 包含了距离计算和合成向量过滤 2 个过程。过程 A 用于计算每个合成向量与 $\boldsymbol{S}_+$和 $\boldsymbol{S}_-$集合内所有向量之间的加权距离和,得到加权距离集合 $\boldsymbol{H}$,并对 $\boldsymbol{H}$ 进行排序。过程 B 将距离最大的 $\tau \times b$ 个合成向量标记为正,过滤掉更接近 $\boldsymbol{S}_-$的合成向量。

合成向量 $\boldsymbol{S}_{\text{new}_\alpha}$到所有正向量和负向量的距离计算公式分别为

$$h_+ = \sum_{j=1}^{d} \left| \boldsymbol{S}_{\text{new}_\alpha} - \boldsymbol{S}_j \right|, \tag{4}$$

$$h_{-}=\sum_{m=1}^{c}\left|\boldsymbol{S}_{\mathrm{new}_{\alpha}}-\boldsymbol{S}_{m}\right|, \tag{5}$$

式中,h_{+}和 h_{-}分别为合成向量到所有正向量和负向量的距离。$\boldsymbol{S}_{\mathrm{new}_{\alpha}}\in\boldsymbol{S}_{\mathrm{new}}$, $\boldsymbol{S}_{j}\in\boldsymbol{S}_{+}$, $\boldsymbol{S}_{m}\in S_{-}$,$d$ 表示正向量的数量,c 表示负向量的数量。

合成向量 $\boldsymbol{S}_{\mathrm{new}_{\alpha}}$的加权距离为

$$H_{\mathrm{new}_{\alpha}}=\sigma h_{+}+(1-\sigma)h_{-}, \tag{6}$$

式中,$H_{\mathrm{new}_{\alpha}}$表示某个合成向量 $\boldsymbol{S}_{\mathrm{new}_{\alpha}}$加权距离,$\sigma$ 表示权重,$\sigma\in[0,1]$。

所有合成向量的加权距离为

$$\boldsymbol{H}=(H_{1},H_{2},\cdots,H_{z}), \tag{7}$$

式中,$\boldsymbol{H}$ 表示所有合成向量的加权距离值,$z=1.5\times\tau\times b$。

加权距离越大,表示合成向量距离决策边界和负向量越远,选择 $\boldsymbol{H}$ 中距离最大的 $\tau\times b$ 个合成向量合并到 $\boldsymbol{S}$ 中,将扩增得到的矩阵 $\boldsymbol{S}$ 与特征矩阵 $\boldsymbol{F}$ 相乘,得到新的扩增矩阵。

算法 2 为数据扩增和数据过滤的伪代码。第①步是对相关变量进行初始化,第②—⑭步是对 $\boldsymbol{S}$ 矩阵中的数据进行扩增,分别计算合成样本与 $\boldsymbol{S}$ 中所有正向量和负向量之间的距离之和,并将这些距离分别添加到 h_{+_j}和 h_{-_j}中。第⑮—⑱步是通过使用加权公式,将 h_{+_j}和 h_{-_j}联系起来得到新的集合 $\boldsymbol{H}$,第⑲—㉒是对 $\boldsymbol{H}$ 进行排序,选择距离最大的 $\tau\times b$ 个合成向量,并将其与 $\boldsymbol{S}$ 矩阵合并。

算法 2　Augmentation 算法。

输入　正样本数量 b,负样本所占百分比 τ,对象子空间 $\boldsymbol{S}$。

输出　扩增后的数据集 S'。

① $i=1$, $\boldsymbol{S}_{\mathrm{new}}=\varnothing$, $H=\varnothing$;
② while ($i<1.5\times\tau\times b$) do
③　for ($j=1$ to b) do
④　$\beta=0$ to 1 direct random value;
⑤　Select vector $\boldsymbol{S}_{\mathrm{nearst}_j}$ closest to $\boldsymbol{S}_{j}$;
⑥　Generate $\boldsymbol{S}_{\mathrm{new}_i}$ according to Eq. (3);
⑦　$\boldsymbol{S}_{\mathrm{new}}=\boldsymbol{S}_{\mathrm{new}}\cup\boldsymbol{S}_{\mathrm{new}_i}$;
⑧　Calculate h_{+} according to Eq. (4);
⑨　$h_{+_j}=h_{+_j}\cup h_{+}$;
⑩　Calculate h_{-} according to Eq. (5);
⑪　$h_{-_j}=h_{-_j}\cup h_{-}$;
⑫　$i=i+1$;
⑬　end for
⑭ end while
⑮ for ($z=1$ to $1.5\times\tau\times b$) do
⑯　Calculate $H_{\mathrm{new}_{\alpha}}$ according to Eq. (6);
⑰　$\boldsymbol{H}=\boldsymbol{H}\cup\boldsymbol{H}_{\mathrm{new}_{\alpha}}$;
⑱ end for
⑲ Rank $\boldsymbol{H}$ in descending order;
⑳ $\boldsymbol{S}_{\mathrm{new}}$=Select the top-($\tau\times b$) distances in $\boldsymbol{H}$;
㉑ $\boldsymbol{S}'=\boldsymbol{S}\cup\boldsymbol{S}_{\mathbf{new}}$;
㉒ return $\boldsymbol{S}'$;

2.5　MFSP 算法

MFSP 算法伪代码如算法 3 所示。第①—③步是对 $\boldsymbol{S}$、$\boldsymbol{F}$、M_{former}、M_{current}和根节点的初始化,第④—⑩步是对矩阵 $\boldsymbol{S}$ 和 $\boldsymbol{F}$ 的更新操作,使得它们的乘积更加接近原始数据集 $\boldsymbol{D}$,第⑪步是进行空间划分,并建立二叉树,第⑫—⑮步是对每个数据子空间进行数据扩增和过滤,第⑯步是将扩增后的数据子空间合并起来,然后与特征子空间 $\boldsymbol{F}$ 相乘得到新的数据集 $\boldsymbol{D}'$。

算法 3 MFSP 算法。

输入 原始数据集 $\boldsymbol{D}$,学习率 α,正则化参数 γ,隐藏因子 l,正样本数量 b,负样本占比 τ,叶节点大小 g。

输出 增强数据集 $\boldsymbol{D}'$。

① Initialize matrix $\boldsymbol{S}$, $\boldsymbol{F}$, $\boldsymbol{T}$;
② Root = NULL;
③ $M_{\text{former}}=\varphi$, $M_{\text{current}}=0$;
④ $\boldsymbol{S}$, $\boldsymbol{F}$ = UpdateMatrix($\boldsymbol{S}$, $\boldsymbol{F}$, α, λ, l);
⑤ Calculate M_{current} according to Eq. (2);
⑥ while ($M_{\text{former}}>M_{\text{current}}$) do
⑦ $M_{\text{former}}=M_{\text{current}}$;
⑧ $\boldsymbol{S}$, $\boldsymbol{F}$ = UpdateMatrix($\boldsymbol{S}$, $\boldsymbol{F}$, α, λ, l);
⑨ Calculate M_{current} according to Eq. (2);
⑩ end while
⑪ $\boldsymbol{T}$ = Mannoy($\boldsymbol{S}$, Root, g);
⑫ Num = The number of leaf nodes of $\boldsymbol{T}$;
⑬ for (i = 1 to Num)
⑭ $\boldsymbol{T}_i$ = Augmentation($\boldsymbol{T}_i$, b, τ);
⑮ end for
⑯ $\boldsymbol{S}''$ = Merge the amplified subspaces;
⑰ Get a new dataset $\boldsymbol{D}'$ according to Eq. (1);
⑱ return $\boldsymbol{D}'$;

3 实验结果及分析

为了有效评估算法,本文选取以下先进算法进行比较:①最近邻对的合成少数类过采样(synthetic minority oversampling technique tomeklinks, SMOTE_TomekLinks)算法[28];②SMOTE 算法[12];③最近邻算法合成少数类过采样(synthetic minority oversampling technique of edited nearest neights, SMOTE_ENN)算法[28];④用于不平衡学习的自适应合成采样(adaptive synthetic sampling approach for imbalanced learning, ADASYN)算法[29];⑤使用 SMOTE 算法和局部线性嵌入不平衡数据分类(classification of imbalanced data by using the SMOTE algorithm and locally linear embedding, LLE_SMOTE)算法[30];⑥KMeans_SMOTE 算法[13];⑦CURE_SMOTE 算法[14];⑧MFCS 算法[1];⑨KGA 算法[26]。采用 3 种分类器对 9 种算法进行对比实验,实验结果表明 MFSP 算法能够正确识别更多的阳性样本。

3.1 实验数据集

在本节中,采用 9 个数据集进行实验,以验证 MFSP 算法的有效性。数据集来自 AutoML 网站(http://39.100.246.211:8050/dataset)。数据集的具体信息见表 1。

表 1 不平衡微生物数据集

Table 1 Unbalanced microbial datasets

数据集	特征数	少类数	多类数	不平衡比
D008881	153	490	1 170	2.39
D001714	149	315	1 216	3.86
D003863	145	228	1 170	5.13
D001255 9	145	224	1 170	5.22
D001289	142	162	1 170	7.22
D002318	140	116	1 170	10.09
D000067877	140	86	1 170	13.60
D008107	136	62	1 170	18.87
D002446	137	58	1 170	20.17

3.2　评价指标

本文使用了表 2 所示的混淆矩阵,基于混淆矩阵,T_P 指样本真实类别是正类,并且模型将其识别为正类;F_N 指样本真实类别是正类,但模型将其识别为负类;F_P 指样本真实类别是负类,但模型将其识别为正类;T_N 指样本真实类别是负类,并且模型将其识别为负类。

表 2　混淆矩阵
Table 2　Confusion matrix

原始数据集	正类(预测)	负类(预测)
正类(真实)	T_P	F_N
负类(真实)	F_P	T_N

特异性 S_p 指正确预测的真阴性样本的比例,即

$$S_p=\frac{T_N}{T_N+F_P}。\tag{8}$$

召回率 R_e 指正确分类的正例个数占实际正例个数的比例,即

$$R_e=\frac{T_P}{T_P+F_N}。\tag{9}$$

G_m 也是一种综合考虑召回率和特异性的标准,当 G_m 较大时说明模型性能较好,即

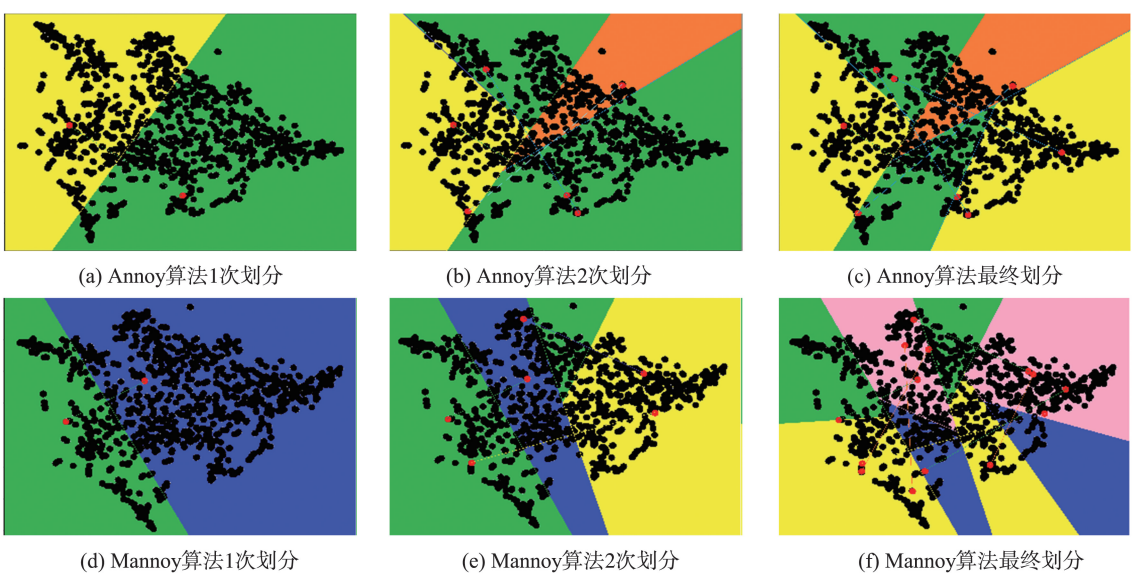
$$G_m=\sqrt{R_eS_p}。\tag{10}$$

A_u 指受试者工作特征曲线(receiver operating characteristic curve, ROC)下面积的值,用于评估分类器区分不平衡数据的分类性能。

3.3　实验结果与分析

3.3.1　空间划分结果

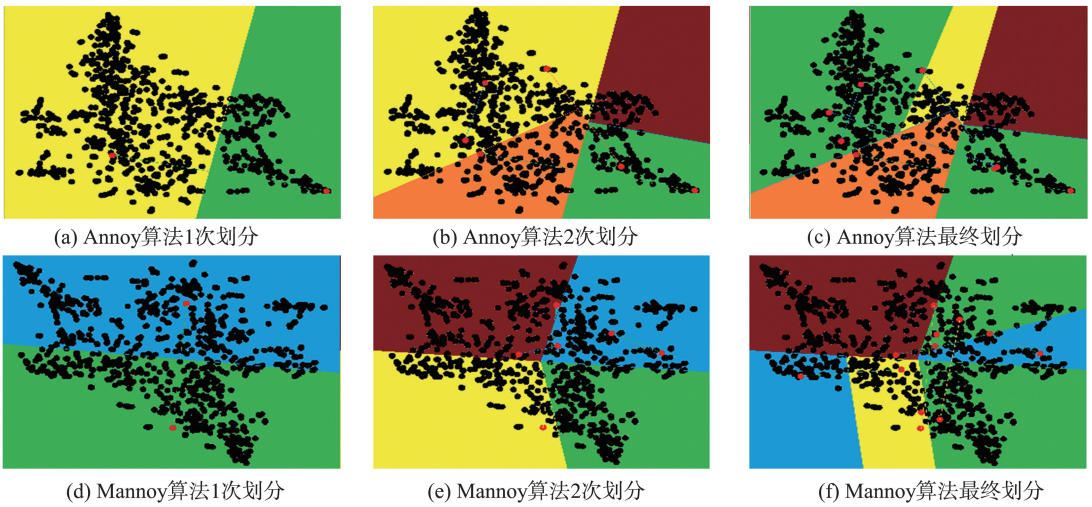
本节选取不平衡比分别为 2.39、13.6 和 20.17 的数据集 D008881、D000067877 和 D002446 进行实验。对这 3 个数据集分别使用 Mannoy 算法和 Annoy 算法进行空间划分,空间划分操作结束后,再降维并绘制分布图,结果如图 3—5 所示。从图 3—5 的空间划分结果可以看出,Annoy 算法划分的各个空间中的样本数量差异较大,并且相对密度较大的区域也没有得到有效的划分。Mannoy 算法基于样本密度进行空间划分,使得划分线主要经过样本密度较高的区域。考虑样本密度的划分方式使得各数据子空间样本分布较为均匀,并且各个空间中的样本数量相差不大。



(a) Annoy算法1次划分　(b) Annoy算法2次划分　(c) Annoy算法最终划分

(d) Mannoy算法1次划分　(e) Mannoy算法2次划分　(f) Mannoy算法最终划分

图 3　Annoy 算法和 Mannoy 算法在数据集 D008881 上的划分过程
Fig.3　The division process of Annoy algorithm and Mannoy algorithm on dataset D008881

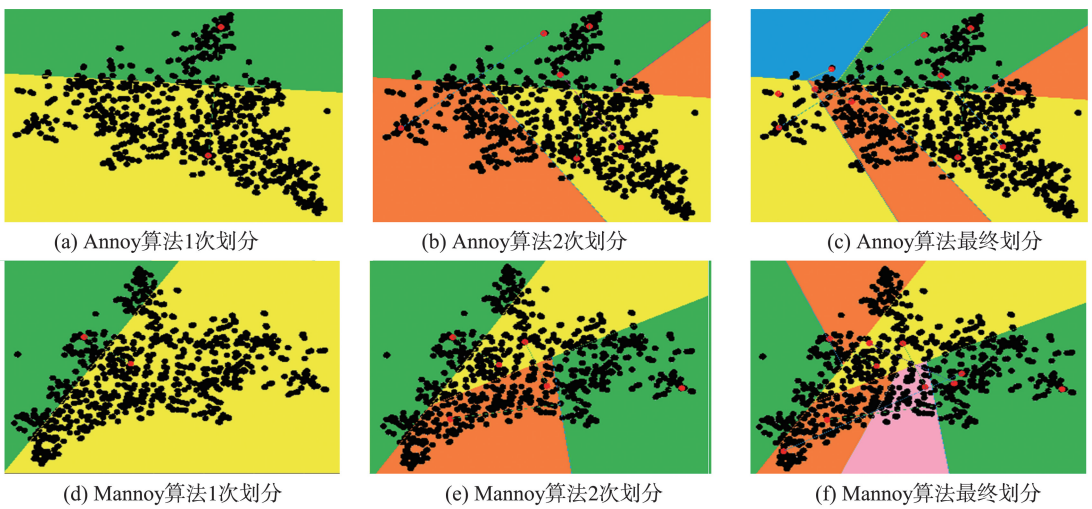


(a) Annoy算法1次划分 (b) Annoy算法2次划分 (c) Annoy算法最终划分

(d) Mannoy算法1次划分 (e) Mannoy算法2次划分 (f) Mannoy算法最终划分

图 4 Annoy 算法和 Mannoy 算法在数据集 D000067877 上的划分过程

Fig.4 The division process of Annoy algorithm and Mannoy algorithm on dataset D000067877



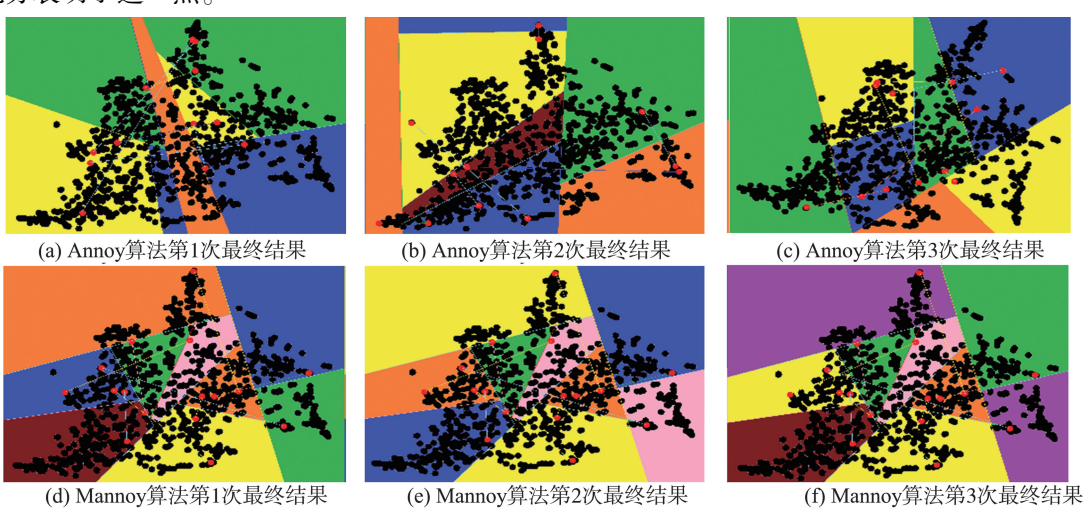
(a) Annoy算法1次划分 (b) Annoy算法2次划分 (c) Annoy算法最终划分

(d) Mannoy算法1次划分 (e) Mannoy算法2次划分 (f) Mannoy算法最终划分

图 5 Annoy 算法和 Mannoy 算法在数据集 D002446 上的划分过程

Fig.5 The division process of Annoy algorithm and Mannoy algorithm on dataset D002446

图 6 给出了 Annoy 算法和 Mannoy 算法分别在数据集 D008881 上的 3 次划分结果。从 Annoy 算法的特性可知,该算法空间划分选点采用随机选取的方式,导致每一次的空间划分结果具有较大的差异。而 Mannoy 算法的空间划分选点方式相对 Annoy 算法稳定了许多,使得空间划分的结果也更加稳定,图 6 中的结果也充分表明了这一点。



(a) Annoy算法第1次最终结果 (b) Annoy算法第2次最终结果 (c) Annoy算法第3次最终结果

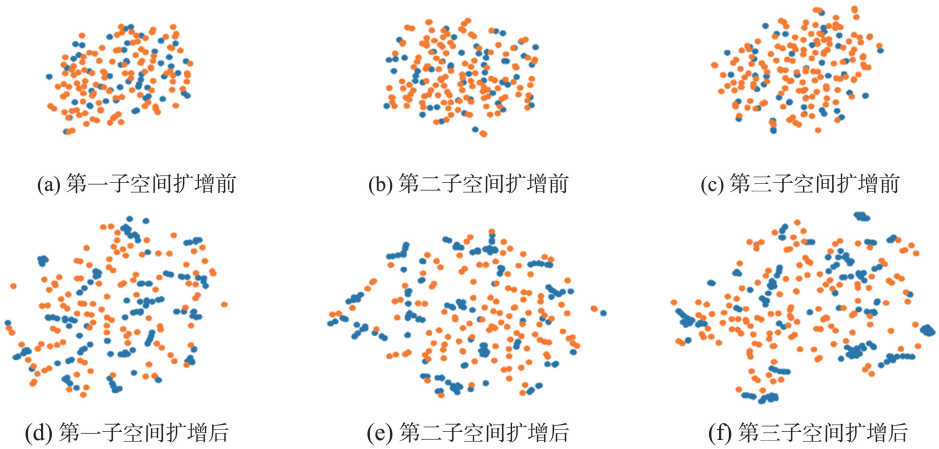
(d) Mannoy算法第1次最终结果 (e) Mannoy算法第2次最终结果 (f) Mannoy算法第3次最终结果

图 6 Annoy 算法和 Mannoy 算法在数据集 D008881 上 3 次最终划分结果

Fig.6 Annoy algorithm and Mannoy algorithm on dataset D008881 on the third final division result

3.3.2　子空间扩增结果

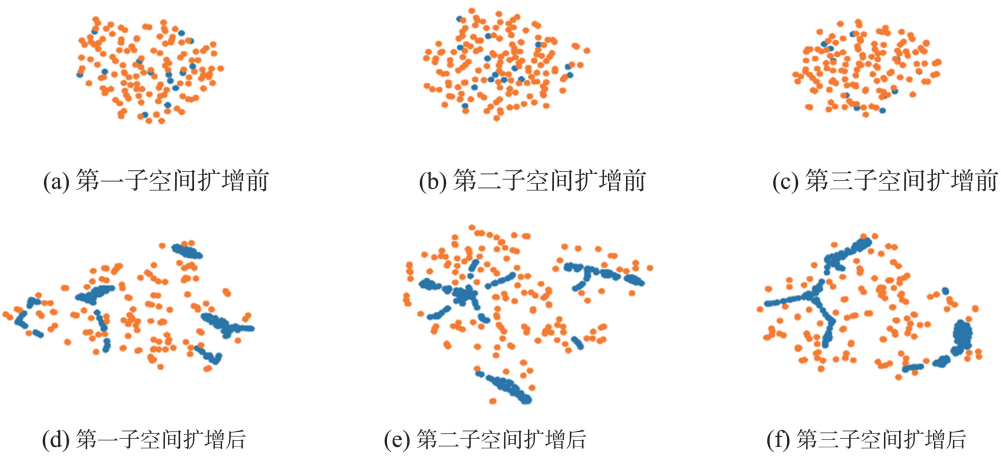
图 7—9 为在数据集 D008881、D000067877 和 D002446 使用 Mannoy 算法空间划分后的数据子空间的扩增效果前后对比。



(a) 第一子空间扩增前　(b) 第二子空间扩增前　(c) 第三子空间扩增前

(d) 第一子空间扩增后　(e) 第二子空间扩增后　(f) 第三子空间扩增后

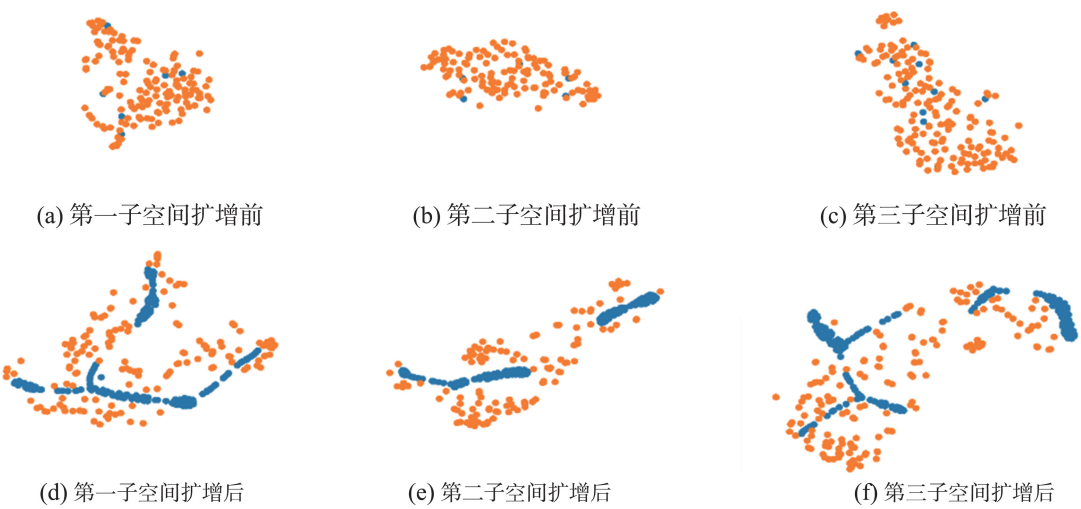
图 7　Mannoy 算法在数据集 D008881 上的数据子空间扩增结果(正类:蓝色;负类:橙色)
Fig.7　Data subspace augmentation results of Mannoy algorithm on dataset D008881
(Positive class: blue; Negative class: orange)



(a) 第一子空间扩增前　(b) 第二子空间扩增前　(c) 第三子空间扩增前

(d) 第一子空间扩增后　(e) 第二子空间扩增后　(f) 第三子空间扩增后

图 8　Mannoy 算法在数据集 D000067877 上的数据子空间扩增结果(正类:蓝色;负类:橙色)
Fig.8　Data subspace augmentation results of Mannoy algorithm on dataset D000067877
(Positive class: blue; Negative class: orange)



(a) 第一子空间扩增前　(b) 第二子空间扩增前　(c) 第三子空间扩增前

(d) 第一子空间扩增后　(e) 第二子空间扩增后　(f) 第三子空间扩增后

图 9　Mannoy 算法在数据集 D002446 上的数据子空间扩增结果(正类:蓝色;负类:橙色)
Fig.9　Data subspace augmentation results of Mannoy algorithm on dataset D002446
(Positive class: blue; Negative class: orange)

图 7 为数据集 D008881 的数据子空间扩增前、后的分布图。从图 7 可以看出,D008881 原始正样本数量远少于负样本数量,正、负样本随机分布且相互重叠使得正、负类样本难以区分。使用 MFSP 算法对数据子空间扩增之后,合成样本均匀地分布在少数类样本区域,扩增后正、负样本产生较明显的边界,使得正、负样本的区分更加容易。

图 8—9 为在数据集 D000067877、D002446 的数据子空间扩增前、后的分布图。数据集 D000067877 和 D002446 中的正类样本数量远少于负样本数量,并且正负类样本分布没有任何规律使得正、负类样本难以区分。通过 MFSP 算法对数据子空间进行扩增后,合成样本主要出现在正、负样本的边界区域,并且合成样本呈现聚集现象。MFSP 算法通过在空间划分后的数据子空间中进行扩增,能够将原始矩阵扩增问题简化,在数据子空间中扩增能够更加准确识别正类样本,并在正类样本周围合成新样本。

3.3.3 分类结果

实验将所选数据集进行 5 次矩阵分解,从中选取 3 组结果分别进行 5 次实验。每组数据集得到 15 个实验结果,并将 15 个结果求平均,以减轻随机性多实验结果的影响。然后在 9 个数据集上验证了 MFSP 算法和 9 种采样方法的分类有效性。表 3—5 分别为各个采样方法采用 3 种分类器时的分类结果。原始数据集(original data set,ODS),黑体表示在对应数据集及对应指标下某个算法取得的最佳性能。

表 3 中,SMO 表示 SMOTE 算法,STL 表示 SMOTE_TomekLinks 算法,SOME 表示 SMOTE_ENN 算法,ADA 表示 ADASYN 算法,LSMO 表示 LLE_SMOTE 算法,KSMO 表示 KMeans_SMOTE 算法,CSMO 表示 CURE_SMOTE 算法。

表 3 不同分类器下不同数据集的 R_e 对比

Table 3 R_e comparison of different data sets under different classifiers

分类器	数据集	ODS	SMO	STL	SMOE	ADA	LSMO	KSMO	CSMO	MFCS	KGA	MFSP
GBDT	D008881	0.10	0.63	0.64	0.01	**0.72**	0.50	0.01	0.60	0.38	0.15	0.70
	D001714	0	**0.62**	0.60	0	0.58	0.59	0.02	**0.62**	0.12	0.14	0.42
	D003863	0	0.64	0.63	0	**0.73**	0.42	0.08	0.62	0.36	0.18	0.32
	D0012559	0	0.60	0.60	0	0.55	0.56	0	**0.62**	0.27	0.23	0.54
	D001289	0	0.36	0.33	0	**0.49**	0.47	0	0.38	0.19	0.15	0.41
	D002318	0	0.48	0.51	0	0.48	0.45	0	0.45	**0.51**	0	0.48
	D0067877	0	0.50	0.50	0	0.45	0.30	0	**0.60**	0.19	0.41	0.49
	D008107	0	0.32	0.26	0	0.32	0.21	0	0.11	0.16	0.33	**0.89**
	D002446	0	0.47	0.47	0	0.47	0.41	0	0.47	0.05	0.27	**0.80**
SVM	D008881	0.10	0.63	0.64	0.01	**0.72**	0.50	0.01	0.60	0.38	0.15	0.70
	D001714	0	**0.62**	0.60	0	0.58	0.59	0.02	**0.62**	0.12	0.14	0.42
	D003863	0	0.64	0.63	0	**0.73**	0.42	0.08	0.62	0.36	0.18	0.32
	D0012559	0	0.6	0.6	0	0.55	0.56	0	**0.62**	0.27	0.23	0.54
	D001289	0	0.36	0.33	0	**0.49**	0.47	0	0.38	0.19	0.15	0.41
	D002318	0	0.48	0.51	0	0.48	0.45	0	0.45	**0.51**	0	0.48
	D0067877	0	0.50	0.50	0	0.45	0.30	0	**0.60**	0.19	0.41	0.49
	D008107	0	0.32	0.26	0	0.32	0.21	0	0.11	0.16	0.33	**0.89**
	D002446	0	0.47	0.47	0	0.47	0.41	0	0.47	0.05	0.27	**0.80**
决策树	D008881	0.40	0.47	0.39	0.37	0.42	0.37	0.36	0.39	0.10	0.41	**0.52**
	D001714	0.26	0.36	0.36	0.26	0.33	0.23	0.25	0.30	0.08	0.27	**0.45**
	D003863	0.19	0.23	0.30	0.15	0.36	0.23	0.21	0.29	0.14	0.18	**0.45**
	D0012559	0.22	0.36	0.29	0.18	0.18	0.20	0.22	0.20	0.15	0.16	**0.70**
	D001289	0.11	0.16	0.31	0.09	0.20	0.29	0.04	0.07	0.23	0.07	**0.63**
	D002318	0.12	0.18	0.12	0.09	0.12	0.09	0.15	0.12	0.04	0.17	**0.45**
	D0067877	0.10	0.1	0.10	0.10	0.2	0.15	0.10	0.10	0.23	0.18	**0.59**
	D008107	0	0.11	0.11	0	0.16	0.05	0.05	0.11	0.04	0.08	**0.58**
	D002446	0.06	0.12	0.12	0.06	0.18	0	0.06	0.06	0.11	0.09	**0.51**

表 4　不同分类器下不同数据集的 G_m 对比

Table 4　G_m comparison of different data sets under different classifiers

分类器	数据集	ODS	SMO	STL	SMOE	ADA	LSMO	KSMO	CSMO	MFCS	KGA	MFSP
GBDT	D008881	0.33	0.52	0.50	0.32	**0.56**	0.42	0.36	0.31	0.32	0.3	0.50
	D001714	0.38	0.49	0.49	0.38	0.46	0.34	0.39	0.33	0.30	0.20	**0.51**
	D003863	0.16	**0.45**	0.40	0.16	0.37	0.28	0.20	0	0	0.20	0.40
	D0012559	0.23	0.31	0.36	0.20	0.29	0.19	0.19	0.13	0	0.15	**0.39**
	D001289	0.15	0.14	0.20	0.15	0.21	0.15	0.15	0.15	0.14	0.18	**0.31**
	D002318	0	**0.34**	0.29	0	0.29	0	0	0	0	0	0.10
	D0067877	0.22	0.31	0.22	0.31	0.31	0.22	0.22	0.22	0	0.23	**0.48**
	D008107	0	0.23	0.23	0	0.23	0	0	0	0	0	**0.41**
	D002446	0	0.34	0.34	0	0.34	0	0	0	0	0.28	**0.47**
SVM	D008881	0.31	0.52	0.53	0.09	0.52	**0.55**	0.09	0.55	0.34	0.37	0.50
	D001714	0	0.57	0.56	0	0.55	**0.63**	0.14	0.58	0.30	0.34	0.44
	D003863	0	0.54	0.53	0	**0.58**	0.50	0.23	0.55	0.36	0.39	0.33
	D0012559	0	0.53	0.53	0	0.53	0.50	0	0.50	0.30	0.44	**0.53**
	D001289	0	0.48	0.46	0	0.48	0.52	0	**0.49**	0.39	0.32	0.44
	D002318	0	0.52	0.53	0	0.52	0.50	0	0.51	**0.54**	0	0.45
	D0067877	0	0.56	0.56	0	0.52	0.47	0	**0.57**	0.34	0.52	0.53
	D008107	0	**0.48**	0.44	0	0.47	0.41	0	0.32	0.31	0.42	0.38
	D002446	0	0.55	0.55	0	0.54	**0.57**	0	0.52	0.10	0.40	0.46
决策树	D008881	0.53	**0.55**	0.51	0.52	0.53	0.53	0.51	0.52	0.15	0.55	0.54
	D001714	0.46	0.53	0.52	0.46	0.51	0.45	0.45	0.50	0.17	0.45	**0.53**
	D003863	0.39	0.42	0.47	0.35	**0.53**	0.44	0.41	0.49	0.20	0.37	0.46
	D0012559	0.42	**0.52**	0.48	0.38	0.38	0.40	0.42	0.40	0.24	0.36	0.48
	D001289	0.31	0.36	0.50	0.28	0.41	0.50	0.20	0.24	0.20	0.24	**0.52**
	D002318	0.33	0.40	0.33	0.28	0.32	0.28	0.37	0.34	0.11	0.38	**0.43**
	D0067877	0.30	0.30	0.30	0.30	0.42	0.37	0.31	0.30	0.20	0.39	**0.47**
	D008107	0	0.31	0.10	0	0.38	0.22	0.22	0.32	0.11	0.26	0.50
	D002446	0.23	0.33	0.33	0.23	0.41	0	0.23	0.23	0.12	0.28	**0.44**

表 5　不同分类器下不同数据集的 A_u 对比

Table 5　A_u comparison of different data sets under different classifiers

分类器	数据集	ODS	SMO	STL	SMOE	ADA	LSMO	KSMO	CSMO	MFCS	KGA	MFSP
GBDT	D008881	0.52	0.58	0.56	0.52	**0.60**	0.56	0.52	0.52	0.50	0.52	0.54
	D001714	0.54	**0.59**	0.58	0.56	0.57	0.55	0.56	0.55	0.51	0.48	0.56
	D003863	0.51	**0.57**	0.55	0.51	0.54	0.53	0.50	0.49	0.50	0.51	0.53
	D0012559	0.52	0.50	0.53	0.51	0.49	0.50	0.52	0.50	0.50	0.49	**0.53**
	D001289	0.50	0.48	0.49	0.50	0.50	0.50	0.50	0.50	0.51	0.45	**0.52**
	D002318	0.49	**0.53**	0.52	0.49	0.52	0.49	0.49	0.50	0.50	0.5	0.51
	D0067877	0.52	0.54	0.51	0.54	0.54	0.52	0.52	0.52	0.50	0.47	**0.56**
	D008107	0.50	0.52	0.52	0.50	0.52	0.49	0.49	0.50	0.50	0.44	**0.56**
	D002446	0.50	0.54	0.54	0.50	**0.55**	0.50	0.50	0.50	0.50	0.49	0.54
SVM	D008881	0.54	0.53	0.54	0.50	0.55	0.56	0.50	0.55	0.51	0.54	**0.57**
	D001714	0.50	0.57	0.56	0.50	0.55	**0.63**	0.49	0.58	0.50	0.49	0.53
	D003863	0.50	0.55	0.54	0.50	**0.59**	0.51	0.52	0.55	0.51	0.51	0.51
	D0012559	0.50	0.53	0.53	0.5	0.53	0.50	0.50	0.51	0.51	0.53	**0.53**
	D001289	0.50	0.50	0.49	0.50	0.48	0.52	0.50	0.51	0.51	0.43	**0.52**
	D002318	0.50	0.53	0.53	0.50	0.52	0.50	0.50	0.52	0.52	0.49	**0.54**
	D0067877	0.50	0.56	0.57	0.50	0.52	0.52	0.50	**0.57**	0.51	0.54	0.55
	D008107	0.50	0.52	0.49	0.50	0.51	0.50	0.50	0.52	0.49	0.43	**0.53**
	D002446	0.50	0.55	0.55	0.50	0.55	**0.60**	0.50	0.52	0.49	0.43	0.55

续表

分类器	数据集	ODS	SMO	STL	SMOE	ADA	LSMO	KSMO	CSMO	MFCS	KGA	MFSP
决策树	D008881	0.55	0.56	0.53	0.55	0.54	**0.56**	0.54	0.54	0.50	0.55	0.54
	D001714	0.54	0.56	0.55	0.54	0.56	0.55	0.53	0.56	0.50	0.51	**0.56**
	D003863	0.50	0.50	0.52	0.48	**0.58**	0.53	0.51	0.56	0.54	0.48	0.53
	D0012559	0.52	**0.55**	0.54	0.49	0.48	0.50	0.51	0.52	0.50	0.49	0.54
	D001289	0.49	0.49	0.56	0.49	0.53	**0.57**	0.46	0.48	0.49	0.43	0.55
	D002318	0.51	**0.54**	0.51	0.49	0.50	0.48	0.52	0.53	0.50	0.51	0.53
	D0067877	0.51	0.49	0.49	0.51	0.54	0.53	0.52	0.51	0.49	0.51	**0.54**
	D008107	0.47	0.51	0.51	0.48	0.53	0.49	0.50	0.53	0.50	0.46	**0.54**
	D002446	0.49	0.52	0.52	0.50	**0.56**	0.48	0.50	0.50	0.50	0.47	0.54

由表 3—5 可知,在梯度提升决策树(gradient boosting decision tree, GBDT)中,MFSP 算法的 R_e 指标在大多数数据集上排名第一,且其他 2 个指标相对稳定。MFSP 算法在数据集 D008107 上的 3 个评价指标效果最好,但在数据集 D008881 上表现出的结果相对弱于其他算法。最新采样算法 KGA 和 MFCS 算法在分类性能上较为良好,但存在部分评价指标为 0 的情况,表明 MFSP 算法能够鉴定出更多的阳性样本。

对于支持向量机(support vector machine, SVM)分类器,MFSP 算法在 9 个数据集上得到的评价指标较为均衡,不存在评价指标为 0 的情况。MFSP 算法在数据集 D008107 上表现出的结果相对其他算法比较有优势,但在其他数据集上得到的结果相对较差。最新的采样算法 KGA 和 MFCS 算法通过 SVM 分类器所得到的结果相比于 GBDT 分类器有所提高。

对于决策树分类器,MFSP 算法所得评价指标一半以上排名第一,在数据集 D001714 上得到的评价指标效果最好,并且 R_e 指标相比于其他算法高 40%。决策树分类器所得实验结果相比于 SVM 更加均衡,且优于其他算法。实验结果表明, MFSP 算法在 3 个分类器中的分类结果优于现有的最新采样算法。

3.3.4 消融实验

为了证明矩阵分解和空间划分的有效性,在 3 个数据集上进行消融实验,实验结果如表 6 所示。Non 表示不经过处理的原始数据集,Mannoy+Augmentation 表示不经过矩阵分解下,原始数据集在进行空间划分后直接进行扩增和过滤,MF+Augmentation 表示不经过空间划分处理,原始数据集在矩阵分解后进行扩增和过滤,Augmentation 表示将原始数据集直接进行扩增和过滤操作。

表 6 采用 3 个数据集时的消融实验

Table 6 Ablation experiments by using three datasets

数据集	模型	R_e	G_m	A_u
D008881	Non	0.21	0.34	0.54
	Mannoy+Augmentation	0.37	0.47	0.54
	MF+Augmentation	0.31	0.33	0.51
	Augmentation	0.26	0.45	0.54
	MFSP	**0.52**	**0.51**	**0.55**
D000067877	Non	0.05	0.18	0.51
	Mannoy+Augmentation	0.21	0.39	0.53
	MF+Augmentation	0.07	0.14	0.50
	Augmentation	0.12	0.32	0.53
	MFSP	**0.61**	**0.49**	**0.55**
D002446	Non	0.02	0.07	0.50
	Mannoy+Augmentation	0.04	0.16	0.49
	MF+Augmentation	0.33	0.02	0.50
	Augmentation	0.08	0.21	0.52
	MFSP	**0.67**	**0.46**	**0.54**

由表 6 可知,相较于原始数据集,Mannoy+Augmentation、MF+Augmentation 和 Augmentation 分类性能有所提升,能够识别出更多的正样本,Mannoy+Augmentation 和 MF+Augmentation 分类效果要好于

Augmentation 的分类效果,说明矩阵分解和空间划分使得实验结果有所改进。在空间划分后的数据子空间中采用过采样方法,是 MFSP 算法分类性能提升的重要原因。

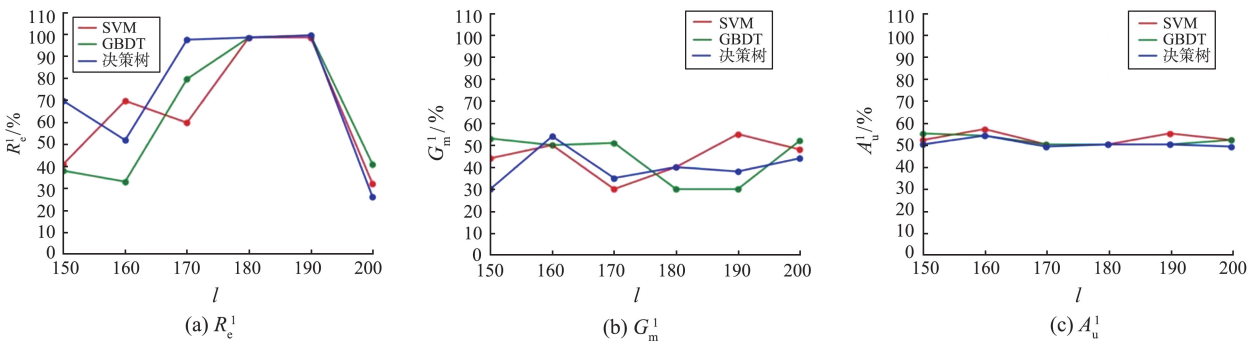
MFSP 模型能够取得最好的分类效果,原因在于原始数据矩阵分解后再进行空间划分,在划分好的每一个数据空间中扩增,通过这种在数据空间的数据扩增,能够有效地解决数据集正负类不平衡问题,从而更加有效地区分正、负类样本。

3.3.5　参数实验

MFSP 算法有 2 个关键参数:隐藏因子 l 和叶节点大小 g。本节通过实验来验证 2 个关键参数取值不同时对实验结果的影响。实验使用数据集 D008881 分别在 3 个分类器上进行实验。不同隐藏因子在 3 个分类器上的评估指标为 R_e^1、G_m^1 和 A_u^1,不同叶节点大小在 3 个分类器上的评估指标为 R_e^2、G_m^2 和 A_u^2。

当 $5 \leqslant l \leqslant 30$ 时,MFSP 算法在 3 种分类器上的性能如图 10 所示。实验结果表明,当 $20 \leqslant l \leqslant 25$ 时,R_e^1 具有一定的稳定性。随着 l 的变化,G_m^1 在 GBDT 分类器上相对较为稳定,在 SVM 和决策树分类器上 G_m^1 波动相对较大。当 l 取不同值时,3 种分类器上 A_u^1 相对稳定,并在较小的范围内浮动。

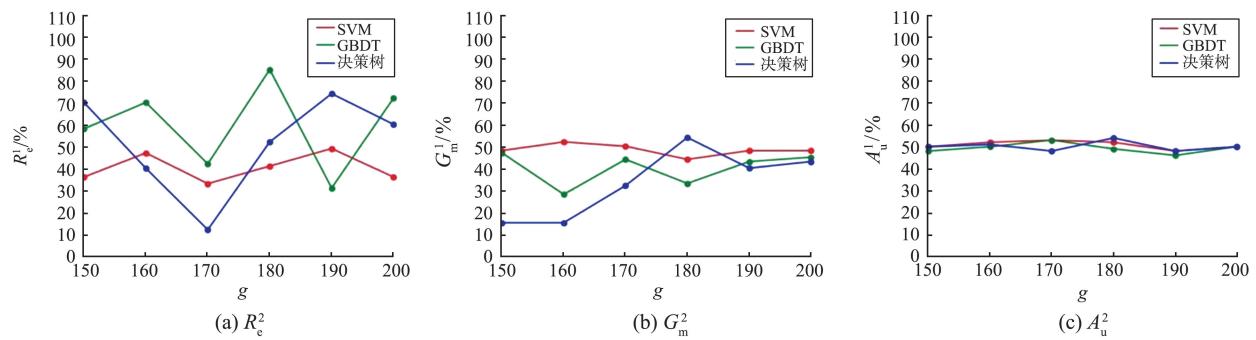
当 $150 \leqslant g \leqslant 200$ 时,MFSP 算法在 3 种分类器上的性能如图 11 所示。实验结果表明,当 g 取不同值时,3 种分类器生成的 R_e^2 实验结果波动较大。当 $180 \leqslant g \leqslant 200$ 时,G_m^2 在 3 种分类器上较为稳定,g 取不同值时,3 种分类器的 A_u^2 实验结果均在小范围内浮动。



(a) R_e^1　(b) G_m^1　(c) A_u^1

图 10　不同隐藏因子时在 3 个分类器上的性能

Fig.10　Performance on three classifiers at different hidden factors



(a) R_e^2　(b) G_m^2　(c) A_u^2

图 11　不同叶节点大小时在 3 个分类器上的性能

Fig.11　Performance on three classifiers at different leaf node sizes

4　总结

本文提出一种融合矩阵分解和空间划分的数据扩增算法。在 9 个微生物数据集上对所提出的 MFSP 算法和 9 种先进的采样算法的性能进行了对比。从分类性能结果来看,MFSP 算法的 A_u 指标在大多数数据要优于其他算法。在 GBDT 和决策树分类器上,MFSP 算法的各项指标一半以上排名第一,相比其他算法有较大优势。说明所提 MFSP 算法能够有效解决类别不平衡和数据高稀疏性问题。在消融实验中验证了矩阵分解和空间划分的有效性。实验结果表明,与原始数据的性能进行对比,Mannoy + Augmentation、MF + Augmentation 和 Augmentation 在分类性能上具有较大的提升。MFSP 算法的评估指标最优。本文为微生物

数据扩增提供了一条新的研究思路,后续将设计其他有关不平衡数据的数据扩增方法和数据过滤方法。

参考文献:

[1] WEN Liuying, WANG Xi, MIN Fan. Cost-sensitive microbial data augmentation through matrix factorization[J]. Applied Intelligence, 2022, 53(10):12684-12700.

[2] ZHANG Chong, TAN Kaychen, LI Haizhou, et al. A cost-sensitive deep belief network for imbalanced classification[J]. IEEE Transactions on Neural Networks and Learning Systems, 2018, 30(1):109-122.

[3] LAPIERRE N, WANG W, ZHOU G, et al. Metapheno: a critical evaluation of deep learning and machine learning in metagenome-based disease prediction[J]. Methods, 2019, 15(166):74-82.

[4] ZHANG Yong, ZHANG Heping. Microbiota associated with type 2 diabetes and its related complications[J]. Food Science and Human Wellness, 2013, 2(3):167-172.

[5] 张玉凤,荆功超,李劲华,等. 基于微生物组大数据搜索的疾病检测[J]. 科学, 2021, 73(2):24-30.
ZHANG Yufeng, JING Gongchao, LI Jinhua, et al. Disease detection based on microbiome big data search[J]. Science, 2021, 73(2):24-30.

[6] GARCIA V, SANCHEZ S J, MARTIN F R, et al. Surrounding neighborhood-based smote for learning from imbalanced data sets[J]. Progress in Artificial Intelligence, 2013, 1(4):347-362.

[7] SPELMEN V S, PORKODI R. A review on handling imbalanced data[C]// International Conference on Current Trends Towards Converging Technologies (ICCTCT), Coimbatore. New Delhi: IEEE, 2018:1-11.

[8] NGUYEN H T, TRAN T B, BUI M Q, et al. Enhancing disease prediction on imbalanced metagenomic dataset by cost-sensitive[J]. International Journal of Advanced Computer Science and Applications, 2020, 11(7):1-6.

[9] PETROSINO J F. The microbiome in precision medicine: the way forward[J]. Genome Medicine, 2018, 10(1):1-4.

[10] PENG Minlong, ZHANG Qi, XING Xiaoyu, et al. Trainable undersampling for class-imbalance learning[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2019, 33(1):4707-4714.

[11] BARUA S, ISLAM M M, YAO X, et al. MWMOTE-majority weighted minority oversampling technique for imbalanced data set learning[J]. IEEE Transactions on Knowledge and Data Engineering, 2014, 26(2):405-425.

[12] CHAWLA V N, BOWYER W K, HALL O L, et al. SMOTE: synthetic minority over-sampling technique[J]. The Journal of Artificial Intelligence Research, 2002, 16(1):321-357.

[13] LI Wenjie. Imbalanced data optimization combining *k*-means and smote[J]. International Journal of Performability Engineering, 2019, 15(8):2173-2181.

[14] LI Ma, FAN Suohai. Cure-smote algorithm and hybrid algorithm for feature selection and parameter optimization based on random forests[J]. BMC Bioinformatics, 2017, 18(1):1-16.

[15] BUNKHUMPORNPAT C, SINAPIROMSARAN K. DBMUTE: density-based majority under-sampling technique[J]. Knowledge and Information Systems, 2017, 50(3):827-850.

[16] 赵增,李明勇,胡航飞. 基于邻居聚类的近似最近邻搜索[J]. 智能计算机与应用, 2020, 10(11):70-78.
ZHAO Zeng, LI Mingyong, HU Hangfei. Approximate nearest neighbor search based on neighbor clustering[J]. Intelligent Computers and Applications, 2020, 10(11):70-78.

[17] 卫泽刚, 侯一凡, 张小丹, 等. 微生物操作分类单元划分算法研究[J]. 宝鸡文理学院学报, 2022, 42(1):80-88.
WEI Zegang, HOU Yifan, ZHANG Xiaodan, et al. Research on the algorithm for division of microbial operation taxonomic units[J]. Journal of Baoji University of Arts and Sciences, 2022, 42(1):80-88.

[18] ANDREAS H, VAKHTANG K. SVD approach to data unfolding[J]. Nuclear Instruments & Methods in Physics Research Section (Aaccelerators Spectrometers Detectors and Associated Equipment), 1996, 372(3):469-481.

[19] LI Wuzhou, LIANG Zhiwen, CAO Yi, et al. Estimating intrafraction tumor motion during fiducial-based liver stereotactic radiotherapy via an iterative closest point (ICP) algorithm[J]. Radiation Oncology, 2019, 14(1):1-8.

[20] YUE Xiaokui, LIU Qicheng. Improved funkSVD algorithm based on RMSProp[J]. Journal of Circuits, Systems and Computers, 2022, 31(8):1-14.

[21] RAJEEV K, VERMA B K, SHYAM R S. Social popularity based SVD++ recommender system[J]. International Journal of Computer Applications, 2014, 87(14):33-37.

[22] 徐彭娜,魏静,林劼,等. 基于位置信息熵的局部敏感哈希聚类方法[J]. 计算机应用与软件, 2018, 35(3):230-235.
XU Pengna, WEI Jing, LIN Jie, et al. Locality-sensitive hash clustering method based on location information entropy[J]. Computer Applications and Software, 2018, 35(3):230-235.

(下转第 44 页)