计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.200800120

# 融合级联上采样与下采样的改进随机森林不平衡数据分类算法

郑建华[1,2]　李小敏[3]　刘双印[1,2]　李　迪[4]
1 仲恺农业工程学院信息科学与技术学院　广州 510225
2 广东省高校智慧农业工程技术研究中心　广州 510225
3 仲恺农业工程学院机电工程学院　广州 510225
4 华南理工大学机械与汽车工程学院　广州 510640
(zhengjianhua@mail.zhku.edu.cn)

**摘　要**　数据不平衡会严重影响传统分类算法的性能,不平衡数据分类是机器学习领域的一个热点和难点问题。为提高不平衡数据集中少数类样本的检出率,提出一种改进的随机森林算法。该算法的核心是对每一棵通过 Bootstrap 采样后的随机森林子树数据集进行混合采样。首先采用基于高斯混合模型的逆权重上采样,然后基于 SMOTE-borderline1 算法进行级联上采样,再用随机下采样方式进行下采样,得到每棵子树的平衡训练子集,最后以决策树为基学习器实现改进机随机森林不平衡数据分类算法。此外,以 G-mean 和 AUC 为评价指标,在 15 个公开数据集上将所提算法与 10 种不同算法进行比较,结果显示其两项指标的平均排名和平均值均为第一。进一步,在其中 9 个数据集上将其与 6 种 state-of-the-art 算法进行比较,在 32 次结果对比中,所提算法有 28 次取得的成绩都优于其他算法。实验结果表明,所提算法有助于提高少数类的检出率,具有更好的分类性能。

**关键词**:级联上采样;随机森林;不平衡数据;分类算法
**中图法分类号**　TP181

## Improved Random Forest Imbalance Data Classification Algorithm Combining Cascaded Up-sampling and Down-sampling

ZHENG Jian-hua[1,2], LI Xiao-min[3], LIU Shuang-yin[1,2] and LI Di[4]
1 College of Information Science and Technology, Zhongkai University of Agriculture and Engineering, Guangzhou 510225, China
2 Guangdong Engineering &Technology Research Center for Smart Agriculture, Guangzhou 510225, China
3 College of Mechanical and Electrical Engineering, Zhongkai University of Agriculture and Engineering, Guangzhou 510225, China
4 School of Mechanical and Automotive Engineering, South China University of Technology, Guangzhou 510640, China

**Abstract**　Data imbalance will seriously deteriorate the performance of traditional classification algorithms. Imbalance data classification has become a hot and difficult problem in the field of machine learning. In order to improve the detection rate of minority samples in imbalance data sets, an improved random forest algorithm is proposed in this paper. The core of the algorithm is to use hybrid sampling for each random forest subtree data set sampled by Bootsrap. Firstly, inverse weight up-sampling based on Gaussian mixture model is adopted, then cascade up-sampling based on SMOTE-borderline1 algorithm is carried out, and down-sampling is carried out in a random down-sampling way, so as to obtain a balanced training subset of each subtree. Finally, adecision tree-based improved random forest learner is used to implement the unbalanced data classification algorithm. In addition, this paper uses G-means and AUC as evaluation indexes, and compares them with 10 different algorithms on 15 public data sets. The results show that the average ranking and average value of the two indexes rank first. Furthermore, this paper compares with 6 state-of-the-art algorithms on 9 data sets. Among the 32 comparisons, the proposed algorithm achieves better results than that of

到稿日期:2020-08-19　返修日期:2020-09-21　本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家重点研发计划(2018YFB1700500);国家自然科学基金(61471133,61871475);广东省科技计划项目(2017A070712019,2017B010126001,2020A1414050062);广东省教育厅项目(2016KZDXM001,2017GCZX001,2020KZDZX1121);广州市科技计划项目(201704030098)
This work was supported by the National Key R&D Program of China(2018YFB1700500), National Natural Science Foundation of China(61471133, 61871475), Science and Technology Planning Project of Guangdong Province of China(2017A070712019, 2017B010126001, 2020A1414050062), Project of Educational Commission of Guangdong Province of China(2016KZDXM001, 2017GCZX001, 2020KZDZX1121) and Science and Technology Planning Project of Guangzhou (201704030098).
通信作者:李小敏(lixiaomin@zhku.edu.cn)

other algorithms for 28 times. The experimental results show that the proposed algorithm is helpful to improve the detection rate of minority class and has better classification performance.

**Keywords** Cascaded up-sampling, Random forest, Imbalance data, Classification algorithm

## 1 引言

精确检测出现实生活中的一些稀有事件非常重要，如信用卡欺诈检测[1]、癌症诊断[2]、网络入侵检测[3]等。在数据挖掘领域，对于这类稀有事件的检测是一个分类问题。但是由于这类问题中异常类(文中称作少数类)和正常类(文中称作多数类)的样本数量不一致，呈现出数据不平衡，甚至是高度不平衡的现象，而传统分类算法向多数类倾斜[4]，导致少数类样本检出率不高。因此如何检出更多的少数类样本，即提高少数类样本的召回率是一个非常值得关注的问题。

为提高少数类样本的检出率，许多学者进行了大量的研究，主要有两类解决方案[5]：基于数据预处理和基于算法改进。其中数据预处理又包括重采样和特征处理技术，而基于算法改进主要包括代价敏感方法和集成方法。对数据集进行特征处理是一种有效的方式，但很少单独使用，一般是集成到重采样[6]或者代价敏感[7]等方法中。代价敏感方法指为少数类中错分样本给予更高的误分代价，以此减小分类器对多数类的倾斜，但是精准确定代价因子是一个难点[8]。鉴于集成学习能显著提高分类性能性[9]以及重采样技术使用的普遍性[5]，本文重点关注重采样技术与集成学习相结合的不平衡数据分类器。

重采样技术中的上采样实际上是按照某种策略生成少数类样本的过程，常见的方法有 SMOTE，SMOTE-Borderline[10]，ADASYN，以及基于 GAN[11]的方法。但是对于不平衡问题，处理好类间数据不平衡比较容易，但是处理好因少数类的“小分离项”问题造成的少数类类内不平衡较为困难[12]。“小分离项”问题表现为少数类样本分块散布在空间的不同位置，不同小块样本数量不平衡且相互之间距离较远，这样容易使传统上采样算法生成的新样本陷入多数类样本密集分布区[13]，成为一个噪声样本。并且上采样数量越大，则引入更多噪声的可能性就越大，从而降低分类性能。为了精准控制上采样，Last 等[14]提出了基于聚类的上采样算法，其首先对数据集进行 $K$-means 聚类，然后在少数类较多的类簇进行上采样，旨在处理“小分离项”问题，但是 $K$-meams 聚类的形状不够灵活，而且当需要生成大量少数类样本时，仍然无法降低生成少数类样本引入的噪声所带来的影响。

重采样技术中的下采样主要是采用一定的策略从多数类样本中筛选部分样本作为代表样本，常见的有随机下采样算法(Random Under Sampling，RUS)。为了提高样本的代表性，研究者首先将多数类样本聚类[15-16]，然后选择聚类中心或者中心附件的样本作为代表性样本。但是下采样会造成多数类样本特征信息丢失[17]，从而有损于分类精度。

混合采样[18-19]是一种折中的方式，其先对少数类样本进行上采样，再对多数类样本进行下采样。这种处理方式在下采样中同样会损失多数类样本特征信息，而在上采样中会生成影响分类性能的噪声样本。

为了缓和上下采样所造成的问题，降低少数类样本生成噪声的影响，从而提升少数类样本的检出率，本文提出了融合级联上采样与下采样的改进随机森林不平衡数据分类算法。该算法的核心是对每一棵随机森林子树的数据集进行混合采样，首先采用基于高斯混合模型[20]的逆权重进行上采样，然后基于 SMOTE-borderline1 算法进行级联上采样，最后用随机下采样的方式进行下采样，以此构建每棵子树的平衡训练子集。随后分别训练多棵子树，采用投票的方式决定最终的分类结果。

本文提出的算法有 3 个特点：首先，该算法采用高斯混合模型进行数据分布的逆合，完成对少数类样本的聚类，然后利用基于高斯混合模型的逆权重进行上采样，这样有利于缓解少数类的“小分离项”问题。基于高斯混合模型的逆权重上采样算法与基于 SMOTE-borderlin1 的上采样算法进行级联上采样，避免每种上采样算法生成过多的少数类样本，从而有助于减少每种上采样算法在采样过程中引入噪声的概率。其次，该算法通过增加基分类器多样性来提升基于 Bagging 集成学习器的分类性能。算法中生成的每一棵子树数据集都具有较大的差异性，这有利于提升基分类器的多样性。最后，在某一棵子树中因下采样而丢失的多数类样本都可能在其他子树被利用，避免了多数类特征信息丢失。另外，在某一子树引入的噪声误差也可能在投票决策过程中被去掉，从而降低了生成少数噪声数据的影响，提升分类性能。

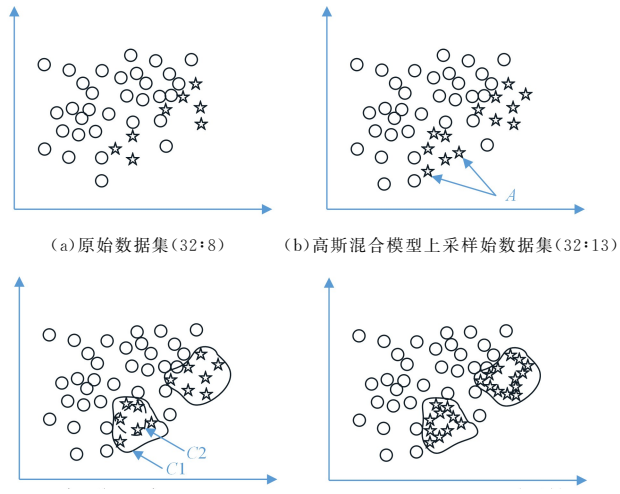
## 2 融合级联上采样与下采样的改进随机森林算法设计

### 2.1 级联上采样原理分析

上采样是一种有效的重采样技术，能够快速地平衡数据集，但是少数类的“小分离项”问题以及上采样会引入噪声样本，制约了上采样技术的应用。为此，本文拟针对少数类样本采用级联上采样方法，首先通过高斯混合模型进行聚类，区分少数类的“小分离项”，并通过该模型对少数类进行上采样，构建类内相对平衡数据集；随后采用经典的 SMOTE-Borderline1 上采样算法，进一步缓解类间不平衡的问题。

图 1 给出了一次级联上采样过程。图 1(a)表示原始数据，图 1(b)中对少数类样本采用高斯混合模型以确定高斯模型的概率分布参数，并形成 2 个小类，对 2 个小类依据样本数量的倒数作为权重，采用已经获得的高斯模型参数分别生成 3 个和 2 个少数类样本，从而通过为数量更少的小块生成更多的样本来缓解少数类“小分离项”问题以及解决类内不平衡问题，同时扩充了少数类样本，缓解了类间不平衡问题。与传统基于 SMOTE 的上采样方式不同，由于高斯混合模型中方差的存在，通过高斯混合模型生成的样本还可以扩大少数类的几何范围，如图 1(b)中的 $A$ 点。图 1(c)给出了两种不同的边界范围，圈 $C1$ 表示采用高斯混合模型生成的样本集范围，而 $C2$ 表示原始样本集范围，显然 $C1$ 的范围比 $C2$ 大。但是

值得注意的是，如果生成的少数类样本过多，则圈 $C2$ 与多数类样本重叠的可能性增加，这样反而会降低分类性能。由于 SMOTE_Borderline1 采样算法[10]首先是从处于“danger”状态的样本中随机选择样本，然后再与附近的少数类样本进行拟合生成新的少数类样本，这样生成的新样本更能提供足够的分类信息。故本文采用 SMOTE_Borderline1 上采样算法扩充少数类样本，进一步缓解类间不平衡问题。图 1(d)给出了对第一次上采样后的子类簇进行 SMOTE_Borderline1 上采样的效果。采用这种级联上采样策略，则少数类样本的生成将由两种上采样方式完成，避免了单独一种方式生成过多少数类样本而导致噪声增加的可能性，从而有利于降低少数类样本中的噪声量。




(a)原始数据集(32∶8)　(b)高斯混合模型上采样始数据集(32∶13)

(c)边界范围示意　(d)SMOTE-Borderline1 上采样(32∶24)

图 1　级联上采样原理示意图

Fig. 1　Schematic diagram of cascade up-sampling principle

## 2.2　基于高斯混合模型的逆权重上采样算法

高斯混合模型 GMM 是 Stauffer 等[21]提出的一个概率统计模型，是对单一高斯密度函数的扩展，可以模拟实现数据的真实概率分布。GMM 假设样本集有若干个内在的概率分布，然后利用不同的概率分布来划分聚类簇[22]，从而实现样本聚类。除了作为聚类模型外，GMM 实质上是一种密度估算算法，因此可以将其作为描述数据分布的生成概率模型，故本文利用各高斯分量的概率分布参数来生成新的少数类样本。相比 $K$-means 聚类方式，GMM 具有以下有优点：1) 聚类形状比较灵活，且有每个样本聚类分配的概率值；2)能够利用求解的概率分布参数快速生成新的样本。

设有包含 $N$ 个训练样本的数据集 $X\{\boldsymbol{x}_i \in R^m, i=1,2,\cdots,N\}$，$m$ 表示输入数据特征维度，该数据集的概率密度函数可以表示为：

$$P(X|\boldsymbol{\Theta})=\sum_{k=1}^{K}\alpha_k f(X|\theta_k)=\prod_{i=1}^{N}\sum_{k=1}^{K}\alpha_k f(\boldsymbol{x}_i|\theta_k) \tag{1}$$

其中，$K$ 是高斯分量的个数，$\theta_k$ 表示第 $k$ 个高斯分量的参数，$\theta_k=(\mu_k,\boldsymbol{\Sigma}_k)$，$\mu_k$ 和 $\boldsymbol{\Sigma}_k$ 分别表示第 $k$ 个高斯分量的均值和协方差矩阵，$\alpha_k$ 为第 $k$ 个高斯分量所占的比例，且 $\sum_{k=1}^{K}\alpha_k=1$，$0<\alpha_k<1$。

通过 EM 算法[23]可以求解每个高斯分量的 $\alpha_k$ 和 $\theta_k=(\mu_k,\boldsymbol{\Sigma}_k)$，每个高斯分量即成为一个聚类类别，随后可以利用每个分量参数生成新的少数类样本。为了得到最优的聚类数目，本文采用 BIC (Bayesian Information Criterion)的最小值作为衡量少数类样本最佳聚类数目的标准。

设 $X$ 中少数类样本 $X^-\{\boldsymbol{x}_i^- \in R^m, i=1,2,\cdots,N\}$，待生成的少数类数量为 $n^{-*}$。

(1)利用 EM 算法求得少数类的最佳聚类数目 $K$，并求解每个类别的模型参数 $\boldsymbol{\Theta}$。

(2)确定每个类别生成的样本数量。针对少数类中的“小分离项”问题，先统计每个类别中元素数量的占比 $\beta_i=n_i^-/\sum_{i=1}^{K}n_i^-$，随后对每个类别的占比取倒数，归一化后确定每个类别的上采样权重 $w_i=1/\beta/\sum_{i=1}^{K}1/\beta_i\ (\sum_{i=1}^{K}w_i=1)$，最后求取每个少数类需要生成的样本数量 $n_i^{-*}=\lceil w_i * n^{-*}\rceil$，其中$\lceil\ \rceil$表示向上取整，这样处理的目的是让样本数量较少的类别生成较多的样本，以缓解少数类样本类内不平衡的问题。

(3)依据每个类别的逆型参数 $\boldsymbol{\Theta}$ 和待生成的样本数量 $n_i^{-*}$，利用高斯模型生成新的样本，最后将生成好的样本加入原有的少数类样本。

基于混合高斯模型的逆权重上采样算法(GMM_IWUS)的伪代码如算法 1 所示。

**算法 1**　基于混合高斯模型的逆权重上采样算法(GMM_IWUS)

输入：少数类样本 $X^-\{\mathbf{x}_i^- \in R^m, i=1,2,\cdots,N\}$，$n^{-*}$，$G^- \leftarrow \varnothing$

输出：$G^-$

1. 执行 EM 算法，求解最佳聚类数据 K，得到每个类别的相关参数 $\boldsymbol{\Theta}=\{\alpha_k,\mu_k,\boldsymbol{\Sigma}_k\}_{k=1}^{K}$

2. 计算每个类别的 $\beta_i=n_i^-/\sum_{i=1}^{K}n_i^-$，并计算每个类别的采样权重 $w_i=1/\beta/\sum_{i=1}^{K}1/\beta_i/(\sum_{i=1}^{K}w_i=1)$

3. for i←1 to k，$G_i^- \leftarrow \varnothing$

4. $n_i^{-*} \leftarrow w_i * n^{-*}$

5. for j←1 to $n_i^{-*}$

6. 　　从高斯分布 $\mathbf{X}\sim N(\mu_i,\boldsymbol{\Sigma}_i)$ 生成一个新的样本 $\mathbf{x}_{ij}^{-*}$

7. $G_i^- \leftarrow G_i^- \cup \{\mathbf{x}_{ij}^{-*}\}$

8. $G^- \leftarrow G^- \cup G_i^-$

9. $G^- \leftarrow G^- \cup X^-$

10. 返回 $G^-$

## 2.3　融合级联上采样与下采样的改进随机森林算法框架设计

集成学习通过结合多个基学习器来完成学习任务，通常可以获得显著优于单一学习器的泛化性能，常见的集成方式包括 Bagging，Boosting，Stacking 等。Bhagat 等[24]进一步指出，提高基分类器的多样性是 Bagging 集成学习获得较好性能的关键因素。随机森林是以决策树为基分类器的 Bagging 集成学习方式的一个扩展变体，其在每棵树应用重采样技术并随机选择不同的特征[18]，以保证每一棵决策树的多样性。正是由于这种多样性，使得随机森林显示出强大的性能，在现实任务中被广泛应用。

针对不平衡数据的分类应用，上采样会引入噪声，下采样会丢失多数类样本特征，这两个问题严重影响了分类算法的

性能。文献[25]通过 SMOTE 对原始整体数据进行上采样，构建一个平衡的训练集，然后将该训练集应用到随机森林模型中，但并没有解决上述两个问题。Zheng 等[18]提出对每一棵随机森林子树采用混合采样，从而构建平衡数据集，通过增加子树的多样性，提升了分类性能，但是该方法无法解决少数类中的“小分离项”问题。本文在文献[14,18]的基础上，结合第 1.1 节中介绍的级联上采样的原理，提出对每一棵随机森林子树采用融合级联上采样与下采样的混合采样策略，并以此构建平衡数据集，最终实现不平衡数据分类算法。具体实现框架如图 2 所示。

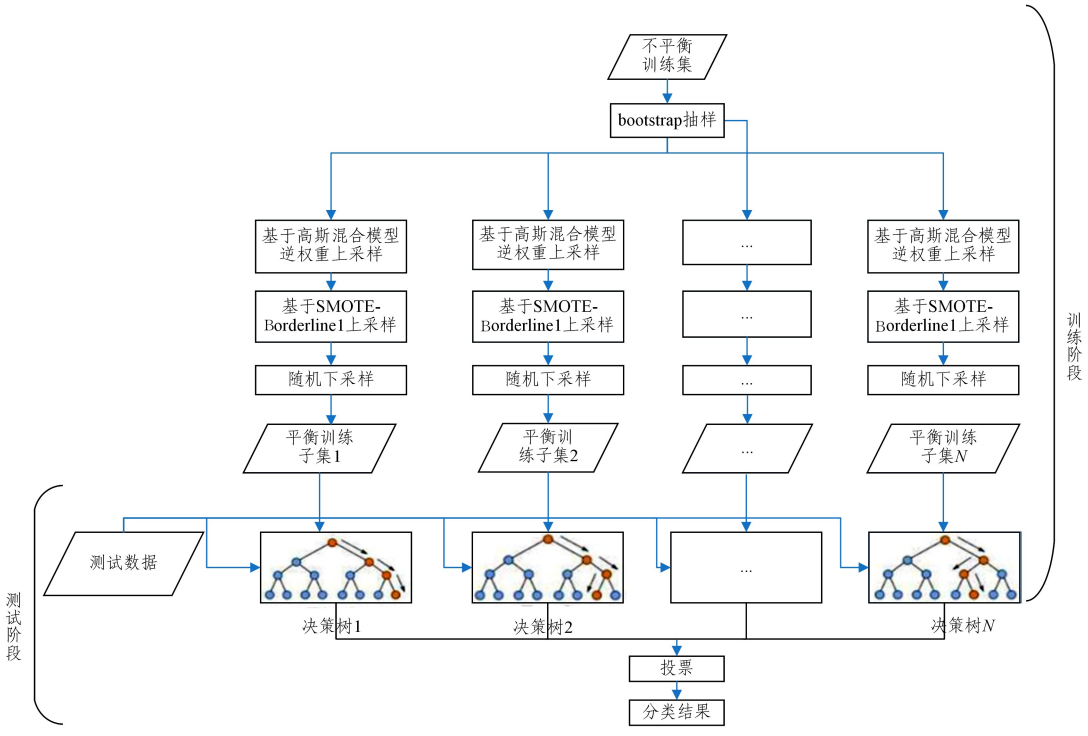



图 2 改进的随机森林算法的框架

Fig. 2 Framework of improved random forest algorithm

融合级联上采样与下采样的改进随机森林算法的整个过程分为训练阶段和测试阶段。训练阶段采用 Bootstrap 抽样，得到每一棵子树的训练集，接着对每一棵子树的训练集采用高斯混合模型逆权重上采样算法进行第一次上采样，然后采用 SMTOE-Borderline1 算法进行第二次上采样，最后采用随机下采样算法，得到平衡训练子集。该算法采用决策树作为基分类器，可以得到 $N$ 棵训练好的决策树模型。在测试阶段，先通过 $N$ 棵决策树预测测试数据的结果，然后将 $N$ 个结果中票数最高的作为该测试数据的结果。

本文提出的框架具有以下特点：1)采用级联上采样策略，针对少数类的“小分离项”问题，通过高斯混合模型逆权重上采样做了类内不平衡处理，同时采用两次上采样可以减少每次上采样的数量，避免引入过多的噪声数据。2)利用级联上采样和下采样融合的混合采样方式增加了集成学习中基分类器的多样性，从而有助于提升分类算法性能。此外，这种模式能够尽量减少上采样所造成的特征信息丢失问题，例如，$A$ 子树丢失的多数类样本可能在 $B$ 子树被利用。同时少数类上采样引入的噪声所带来的影响也会在最终的多棵决策树投票过程中被降至最低。

### 2.4 HyCUD_RF 算法描述

为了清晰地表述整个框架的执行过程，设包含 $N$ 个训练样本的数据集 $X$ 为 $\{\boldsymbol{x}_i \in R^m, i=1,2,\cdots,N\}$，$X^+$ 表示多数类，$X^-$ 分别表示少数类，$|X|$ 表示类别的样本个数，则数据集的不平衡率为 $IR=|X^+|/|X^-|$。对每一棵随机森林的子树，设级联上采样过程中第一次上采样的系数为 $\alpha=coef1*IR$，则其需要上采样的量为 $n_1^{-*}=\alpha*|X^{-sub}|$，其中 $|X^{-sub}|$ 表示子树中少数类的数量；设级联上采样过程中第二次上采样的系数为 $\beta=coef2*IR$，则其需要上采样的量为 $n_2^{-*}=\alpha*|X^{-sub}|$。经过两次上采样后此时每一棵子树数据集的不平衡率的计算公式如式(2)所示。本文认为每棵子树的不平衡率与总数据集的不平衡率相等。

$$IR^{sub}=\frac{|X^{+sub}|}{|X^{-sub}|(1+IR(coef1+coef2))}\approx\frac{IR}{1+IR(coef1+coef2)} \tag{2}$$

显然，$coef1$，$coef2$ 越大，则 $IR^{sub}$ 越小，表示上采样的量越大。

本文将融合级联上采样与下采样的改进随机森林不平衡数据分类算法命名为 HyCUD_RF 算法，算法的训练过程伪代码设计如算法 2 所示。

**算法 2** HyCUD_RF 算法伪代码

输入：数据集 $X\{x_i \in R^m, i=1,2,\cdots,N\}$，coef1，coef2，$n_{tree}$

输出：子树集合 $H\{H_i, i=1,2,\cdots,n_{tree}\}$

1. H←Ø
2. for i←1 to $n_{tree}$
3. $G_i$←Ø
4. 通过 bootstrap 生成子树训练集 $X^{sub}$ 并得到多数类样本 $X^{+sub}$ 和

少数类样本 $X^{-sub}$，计算 IR

5. 确定第一次上采样的少数类样本数量 $n_1^{-*}=coef1*IR*|X^{-sub}|$

6. 采用 GMM_IWUS 算法计算得到第一次上采样数据集 $G_i^1$

7. $G_i \leftarrow G_i \cup G_i^1$

8. 确定第二次上采样的少数类样本数量 $n_2^{-*}=coef2*IR*|X^{-sub}|$

9. 在 $G_i$ 的基础上，采用 SMOTE-Borderline1 算法得到第二次上采样数据集 $G_i^2$

10. $G_i \leftarrow G_i \cup G_i^2$

11. 在 $G_i$ 的基础上，采用随机下采样算法 RUS，得到平衡数据集 $G_i'$

12. 针对 $G_i'$，训练决策树 $H_i$

13. $H \leftarrow H \cup H_i$

14. 返回 H

对测试样本 $\boldsymbol{x}_t$，决策树 $H_i$ 输出 $H_i(\boldsymbol{x}_t)$，则最终结果 $f(\boldsymbol{x}_t)=majorityVote\{H_i(\boldsymbol{x}_t)\}_{i=1}^{n_{tree}}$。

## 3 实验设计

### 3.1 实验数据集

本文拟采用来自 KEEL，UCI（UCI Machine Learning Repository）和 Kaggle 的 15 个数据集来验证本文算法，数据集的不平衡率最小是 1.9，最大为 577.87。数据集中少数类样本的数量最多有 1586，最少只有 20，如表 1 所列。其中，数据分布字段表示数据集中少数类样本和多数类样本的数量，IR 字段表示该数据集的不平衡率，特征数量字段表示该数据集中特征的数量。在后续的实验分析中，为了表述的简洁，本文用简写代码来表示每个数据集。

表 1 实验数据集

Table 1 Experimental datasets

| 简写 | 数据集名称 | 数据分布 | IR | 特征数量 | 数据来源 |
|---|---|---|---|---|---|
| D1 | wisconsin | 241/458 | 1.9 | 9 | KEEL |
| D2 | phoneme | 1586/3818 | 2.41 | 5 | KEEL |
| D3 | haberman | 81/225 | 2.77 | 3 | KEEL |
| D4 | segment0 | 329/1979 | 6.01 | 19 | KEEL |
| D5 | page-blocks-1-3_vs_4 | 28/444 | 8.71 | 10 | KEEL |
| D6 | page-blocks0 | 559/4913 | 8.78 | 10 | KEEL |
| D7 | ecoli-0-1_vs_2-3-5 | 24/220 | 9.16 | 7 | KEEL |
| D8 | pen_digits | 1055/9937 | 9.41 | 16 | UCI |
| D9 | vowel0 | 90/898 | 9.97 | 13 | KEEL |
| D10 | car_eval_34 | 134/1596 | 11.91 | 21 | UCI |
| D11 | dermatology-6 | 20/338 | 16.9 | 34 | KEEL |
| D12 | car-good | 84/1788 | 21.28 | 6 | KEEL |
| D13 | poker-8-9_vs_6 | 25/1460 | 58.4 | 10 | KEEL |
| D14 | kaggle_credict_fraud_50 | 400/20000 | 50 | 30 | Kaggle |
| D15 | kaggle_credict_fraud | 492/284315 | 577.87 | 30 | Kaggle |

### 3.2 算法评价指标

本文仅仅涉及二分类问题，对此混淆矩阵可以较好地表示分类结果，如表 4 所列。对于分类问题，常见的性能评价指标有 Accuracy，Precision，Recall，Specificity，F1，AUC 等。但是这些指标并非都适用于不平衡数据分类，如 $Accuracy=(TP+TN)/(P+N)$。设多数类样本为 10000，少数类样本为 10，假如有一个分类器将少数类全部分错，则该分类器的正确率 $Accuracy=10000/10010=0.999$。对于一个将少数类全部错分的分类器却给出如此高的评价，这显然是不合理的。

对于不平衡应用问题，比如在疾病诊断中，能够及时确诊病情对于患者非常重要，因此提高分类问题的召回率非常重要。Kubat 等[26]提出的 G-mean 是一种鲁棒性较好的不平衡数据分类方法的评价指标，该指标主要关注少数类和多数类的召回率情况。其定义如下：

$$G\text{-}mean=\sqrt{\frac{TP}{TP+FN}\cdot\frac{TN}{TN+FP}} \tag{3}$$

受试者工作特征曲线（Receiver Operating Characteristic Curve，ROC）是根据一系列不同的二分类方式（分界值），以真阳性率（敏感性）为纵坐标，假阳性率（1-特异性）为横坐标绘制的曲线。AUC（Area Under Curve）被定义为 ROC 下的面积，一般用于衡量分类器的性能优劣。*AUC* 越接近 1，表示该分类器的性能越好。

本文采用 G-mean 和 AUC 作为分类算法的性能评价指标。

表 2 混淆矩阵

Table 2 Confusion matrix

| | 预测为多数类（Positive） | 预测为少数类（Negative） |
|---|---|---|
| 实际为多数类（Positive） | *TP* | *FN* |
| 实际为少数类（Negative） | *FP* | *TN* |

### 3.3 实验环境与手段

为了验证 HyCUD_RF 的性能，本文将 HyCUD_RF 以及对比算法应用于 15 个数据集，然后比较 G-mean 和 AUC。本次实验在操作系统 Windows7，CPU 主频为 3.6 GHz，内存 32 GB 的台式机上完成，编程语言为 Python3.6。台式机要求安装以下包：Pandas，Numpy，Sklearn 和 Imblearn。

在实验过程中，为了取得公平的结果，对所有数据集均采用多折交叉验证方法（由于 D5，D7，D11，D13 的少数类样本太少，采用的是 3 折交叉验证，其他数据集则采用 5 折交叉验证），然后执行上述过程 5 次，取 5 次执行结果的平均值作为该算法的结果。

## 4 实验结果与分析

### 4.1 HyCUD_RF 与单一上采样算法的对比

本文提出在每次构建随机森林子树时采用融合级联上采样与下采样的混合采样技术，为了验证级联上采样优于单一上采样，本文对比了单一上采样的 SMOTE 上采样＋下采样、高斯混合模型上采样＋下采样的混合重采样策略，以此构建的随机森林算法分别命名为 HySmote_RF，HyGauss_RF，每种算法的上采样数量为少数类样本的 $0.2*IR$ 倍。对于 HyCUD_RF 算法，设定 $coef1=coef2=0.1$。实验结果如表 3 和图 3 所示。

表 3 不同算法的 *G-mean* 与 *AUC*

Table 3 *G-mean* and *AUC* of different algorithms

| 数据集 | 指标 | HySmote_RF | HyGauss_RF | HyCUD_RF |
|---|---|---|---|---|
| D1 | G-Mean | 0.9731 | 0.9718 | **0.9754** |
| | AUC | 0.9731 | 0.9719 | **0.9754** |
| D2 | G-Mean | 0.8862 | 0.8899 | **0.8900** |
| | AUC | 0.8865 | 0.8902 | **0.8905** |
| D3 | G-Mean | 0.6433 | 0.6455 | **0.6533** |
| | AUC | 0.6526 | 0.6589 | **0.6625** |
| D4 | G-Mean | 0.9937 | 0.9952 | **0.9953** |
| | AUC | 0.9937 | 0.9952 | **0.9953** |
| D5 | G-Mean | **0.9955** | 0.9921 | 0.9951 |
| | AUC | **0.9955** | 0.9922 | 0.9951 |
| D6 | G-Mean | 0.9542 | 0.9528 | **0.9545** |
| | AUC | 0.9542 | 0.9531 | **0.9546** |
| D7 | G-Mean | 0.8990 | 0.8713 | **0.9130** |
| | AUC | 0.9032 | 0.8773 | **0.9147** |
| D8 | G-Mean | 0.9843 | **0.9920** | 0.9915 |
| | AUC | 0.9844 | **0.9920** | 0.9916 |
| D9 | G-Mean | 0.9831 | 0.9815 | **0.9832** |
| | AUC | 0.9831 | 0.9816 | **0.9833** |
| D10 | G-Mean | **0.9840** | 0.9783 | 0.9832 |
| | AUC | **0.9841** | 0.9784 | 0.9833 |
| D11 | G-Mean | 0.9707 | **0.9998** | **0.9998** |
| | AUC | 0.9720 | **0.9998** | **0.9998** |
| D12 | G-Mean | **0.9595** | 0.9383 | 0.9579 |
| | AUC | **0.9601** | 0.9401 | 0.9584 |
| D13 | G-Mean | 0.4814 | 0.8147 | **0.8645** |
| | AUC | 0.6196 | 0.8373 | **0.8784** |
| D14 | G-Mean | 0.9265 | 0.9334 | **0.9436** |
| | AUC | 0.9291 | 0.9354 | **0.9449** |
| D15 | G-Mean | 0.9111 | 0.9251 | **0.9395** |
| | AUC | 0.9151 | 0.9279 | **0.9412** |

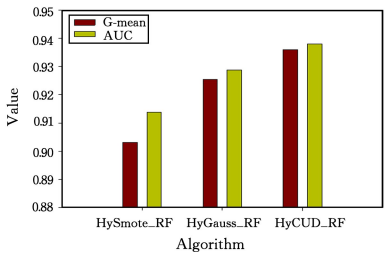



图 3 不同算法的性能对比

Fig. 3 Performance comparison of different algorithms

图 3 给出了不同算法在 15 个数据集上的 *G-mean* 和 *AUC* 平均值的对比。从图 3 可知，HyCUD_RF 算法在 G-mean 和 AUC 上都优于单一上采样的 HySmote_RF 和 HyGauss_RF 算法。具体而言，相比 HySmote_RF，HyGauss_RF 的 *G-mean* 和 *AUC* 分别提升了 2.48%和 1.64%，这说明对少数类数据进行聚类后再上采样有助于减少引入噪声数据，从而提升少数类的分类精准度。而 HyCUD_RF 在 *G-mean* 值上比 HySmote_RF 提升达 3.65%，在 *AUC* 上提升了 2.64%，同样优于 HyGauss_RF 算法，这说明本文提出的级联上采样策略对于提升性能有显著的贡献。

表 3 列出了不同算法在 15 个数据集上的具体 *G-mean* 和 *AUC* 值。其中 HyCUD_RF 在 G-mean 和 AUC 指标上分别获得 11 个第一(1 个并列第一)和 4 个第二的好成绩。而 HyGauss_RF，HySmote_RF 在以上两个指标上获得第一的个数分别为 3 和 2。3 种算法的 *AUC* 值平均排名如下：HySmote_RF 为 2.33，HyGauss_RF 为 2.33，HyCUD_RF 为 1.26。

以上两种比较表明 HyCUD_RF 的总体性能最好，级联上采样比单一上采样更有利于提升分类性能。

### 4.2 HyCUD_RF 性能对比分析

为了验证 HyCUD_RF 的分类性能，本节拟将其与不同的重采样算法进行比较。下采样策略的相关算法有以下 4 种：1)RUS 是被广泛应用的下采样算法，为了与 HyCUD_RF 保持一致，本文将 RUS 与随机森林算法组合构建了 RUS_RF 分类器。2)RUSBoostClassifier[27] 是将 RUS 与基于 boosting 集成学习融合构建的分类器，本文将其简称为 RUSBoost。3)BalancedBaggingClassifier[28] 是在 scikit-learn 的 BaggingClassifier 的基础上应用 RUS 进行平衡处理的分类器，简称为 RUSBagging。4)EasyEnsembleClassifier[29] 算法则从多数类中下采样抽取子集，然后与少数类合并形成训练子集，本文将其简称为 EasyEnsemble。上采样策略的相关算法有 SMOTE+RF，SMOTE_Borderline1+RF 和 ADASYN+RF。考虑到 HyCUD_RF 涉及对少数类的聚类，因此也将其与 K-means+SMOTE[14] 的上采样算法进行了对比。在比较过程中，将所有随机森林子树数量都设置为 50。为了便于表述，对于所有数据集，设定 HyCUD_RF 算法中的 $coef1=coef2=0.1$，高斯混合模型的最大聚类数量除 D5，D7，D11 和 D13 设置为 3 外，其他都设置为 10。实验结果如表 4 和表 5 所列。

表 4 不同算法的 *G-mean* 值

Table 4 *G-mean* value of different algorithms

| | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | D10 | D11 | D12 | D13 | D14 | D15 | Average |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RF | 0.9617 | 0.8707 | 0.4982 | 0.9886 | 0.9566 | 0.9114 | 0.7990 | 0.9789 | 0.9630 | 0.8959 | 0.9380 | 0.7761 | 0.1782 | 0.9137 | 0.8843 | 0.8343 |
| SMOTE_RF | 0.9722 | 0.8876 | 0.5690 | 0.9920 | 0.9658 | 0.9412 | 0.8780 | 0.9861 | 0.9746 | 0.9373 | 0.9388 | 0.8984 | 0.6731 | 0.9209 | 0.8952 | 0.8953 |
| SMOTE_B1_RF | 0.9637 | 0.8898 | 0.5811 | 0.9930 | 0.9658 | 0.9370 | 0.8661 | 0.9771 | 0.9809 | 0.9314 | 0.9024 | 0.8766 | 0.2928 | 0.9103 | 0.8807 | 0.8632 |
| ADASYN_RF | 0.9721 | 0.8897 | 0.6080 | 0.9946 | 0.9829 | 0.9395 | 0.8719 | 0.9817 | 0.9751 | 0.9452 | 0.9710 | 0.8838 | 0.6778 | 0.9141 | 0.8907 | 0.8999 |
| SMOTEBoost | 0.9230 | 0.8406 | 0.5576 | 0.9890 | 0.9892 | 0.9347 | 0.8191 | 0.9720 | 0.9247 | 0.9420 | 0.9009 | 0.8044 | 0.7162 | 0.9056 | 0.8751 | 0.8729 |
| KmeanSmoteRF | 0.9657 | 0.8770 | 0.5945 | 0.9934 | 0.9712 | 0.9216 | 0.8571 | 0.9806 | 0.9753 | 0.9126 | 0.9453 | 0.8774 | 0.2368 | 0.9172 | 0.8835 | 0.8606 |
| RUS_RF | 0.9680 | 0.8798 | 0.6553 | 0.9925 | 0.9513 | 0.9536 | 0.8780 | 0.9909 | 0.9731 | 0.9702 | **1** | 0.9479 | 0.7229 | 0.9347 | 0.9372 | 0.9170 |
| RUSBagging | 0.9677 | 0.8819 | 0.6474 | 0.9905 | 0.9898 | 0.9525 | 0.8825 | 0.9869 | 0.9715 | **0.9839** | 0.9374 | **0.9768** | 0.6451 | 0.9306 | 0.9372 | 0.9121 |
| RUSBoost | 0.9541 | 0.8332 | 0.5090 | 0.9925 | 0.9835 | 0.7877 | 0.8823 | 0.9758 | 0.9566 | 0.9542 | 0.9345 | 0.8760 | 0.5775 | 0.8997 | 0.8946 | 0.8674 |
| EasyEnsemble | 0.9671 | 0.8194 | 0.5797 | 0.9943 | 0.9795 | 0.9515 | 0.8854 | 0.9688 | 0.9768 | 0.9817 | 0.9352 | 0.9517 | 0.4129 | 0.9421 | 0.8751 | 0.8814 |
| HyCUD_RF | **0.9754** | **0.8900** | 0.6533 | **0.9953** | **0.9951** | **0.9545** | **0.9130** | **0.9915** | **0.9832** | 0.9832 | 0.9998 | 0.9579 | **0.8645** | **0.9436** | **0.9395** | 0.9360 |

表 5 不同算法的 *AUC* 值

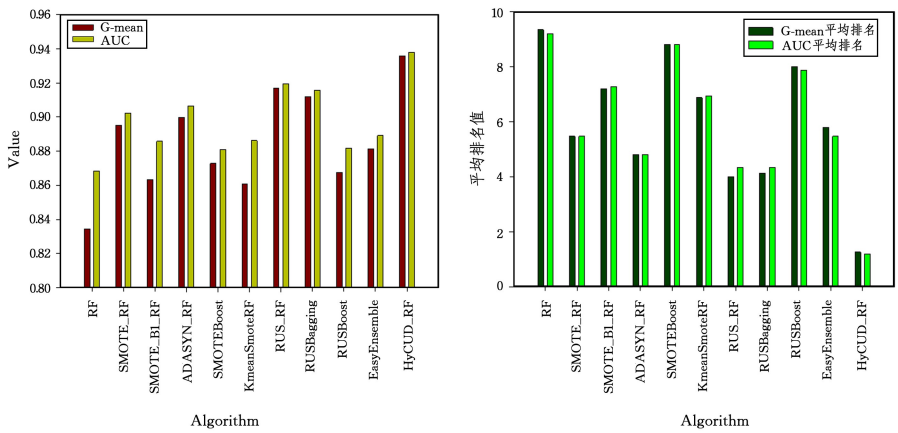
Table 5 *AUC* value of different algorithms

| | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | D10 | D11 | D12 | D13 | D14 | D15 | Average |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RF | 0.9619 | 0.8733 | 0.5744 | 0.9886 | 0.9583 | 0.9145 | 0.8233 | 0.9791 | 0.9637 | 0.9012 | 0.9444 | 0.8021 | 0.5299 | 0.9174 | 0.8911 | 0.8682 |
| SMOTE_RF | 0.9723 | 0.8880 | 0.5818 | 0.9920 | 0.9667 | 0.9420 | 0.8862 | 0.9862 | 0.9749 | 0.9391 | 0.9444 | 0.9035 | 0.7326 | 0.9240 | 0.9007 | 0.9023 |
| SMOTE_B1_RF | 0.9637 | 0.8899 | 0.5937 | 0.9930 | 0.9667 | 0.9379 | 0.8767 | 0.9774 | 0.9810 | 0.9337 | 0.9167 | 0.8835 | 0.5704 | 0.9143 | 0.8879 | 0.8858 |
| ADASYN_RF | 0.9723 | 0.8898 | 0.6127 | 0.9946 | 0.9833 | 0.9401 | 0.8794 | 0.9818 | 0.9754 | 0.9467 | 0.9722 | 0.8898 | 0.7456 | 0.9177 | 0.8967 | 0.9065 |
| SMOTEBoost | 0.9236 | 0.8416 | 0.5762 | 0.9890 | 0.9894 | 0.9358 | 0.8310 | 0.9722 | 0.9267 | 0.9436 | 0.9152 | 0.8233 | 0.7558 | 0.9090 | 0.8827 | 0.8810 |
| KmeanSmoteRF | 0.9658 | 0.8779 | 0.6096 | 0.9934 | 0.9722 | 0.9237 | 0.8683 | 0.9808 | 0.9757 | 0.9168 | 0.9500 | 0.8842 | 0.5640 | 0.9206 | 0.8904 | 0.8862 |
| RUS_RF | 0.9680 | 0.8803 | 0.6600 | 0.9925 | 0.9527 | 0.9537 | 0.8819 | 0.9910 | 0.9732 | 0.9707 | **1** | 0.9493 | 0.7461 | 0.9355 | 0.9380 | 0.9195 |
| RUSBagging | 0.9677 | 0.8820 | 0.6560 | 0.9905 | 0.9899 | 0.9527 | 0.8844 | 0.9870 | 0.9716 | **0.9841** | 0.9430 | **0.9771** | 0.6798 | 0.9322 | 0.9382 | 0.9157 |
| RUSBoost | 0.9544 | 0.8370 | 0.5588 | 0.9925 | 0.9838 | 0.8153 | 0.8875 | 0.9760 | 0.9576 | 0.9557 | 0.9422 | 0.8850 | 0.6769 | 0.9043 | 0.8998 | 0.8818 |
| EasyEnsemble | 0.9672 | 0.8201 | 0.5863 | 0.9943 | 0.9797 | 0.9518 | 0.8865 | 0.9689 | 0.9769 | 0.9819 | 0.9430 | 0.9529 | 0.4460 | 0.9425 | 0.9385 | 0.8891 |
| HyCUD_RF | **0.9754** | **0.8905** | **0.6625** | **0.9953** | **0.9951** | **0.9546** | **0.9147** | **0.9916** | **0.9833** | 0.9833 | 0.9998 | 0.9584 | **0.8784** | **0.9449** | **0.9412** | 0.9379 |

表 4 列出了不同算法在 15 个数据集上的 *G-mean* 值。可知，HyCUD_RF 算法在 11 个数据集上取得第一名，在 D3，D10，D11，D12 这 4 个数据集上取得第二名，且在这 4 个数据集上的结果与第一名分别仅相差 0.305%，0.071%，0.020%，1.935%，除在 D12 上相差略大外（实际上，在 3.3 节中，当 $coef1=coef2=0.04$ 时，HyCUD_RF 算法在 D12 上的 *G-mean* 值为 0.9799，要优于当前的第一名），其他的差距均很小。

表 5 列出了不同算法在 15 个数据集上的 *AUC* 值。可知，本文提出的 HyCUD_RF 算法在 12 个数据集上取得第一名，在 D10，D11，D12 这 3 个数据集上取得第二名，且在这 3 个数据集上的结果与第一名分别仅相差 0.081%，0.02%，1.914%。与 G-mean 类似，HyCUD_RF 算法的 AUC 值只有在 D12 数据集上与第一名差距略大（实际上，在 3.3 节中，当 $coef1=coef2=0.04$ 时，所提算法在 D12 上的 *AUC* 值为 0.9801，要优于当前的第一名），其他的差距均很小。

为了更详细地对比不同算法的性能，图 4 给出了不同算法的 G-mean 和 AUC 的对比情况。图 4(a)给出了不同算法在 15 个数据集上 *G-mean* 和 *AUC* 平均值的对比情况，图 4(b)给出了不同算法在 15 个数据集上的 *G-mean* 与 *AUC* 平均值排名情况。图 4(a)和图 4(b)均显示随机森林算法（图中的 RF）在 11 种算法中的分类性能最差，表现为 *G-mean* 和 *AUC* 值明显较低，排名最差，这表明不平衡数据对分类性能影响较大。当算法进行上采样或者下采样后，分类性能提升明显，其中随机下采样算法（RUS_RF）表现较为突出。本文提出的 HyCUD_RF 则进一步提升了算法的分类性能，在 15 个数据集中的平均 *G-mean* 和 *AUC* 均取得最好的成绩，且排名也是最好的。这证明了本文提出的融合级联上采样与下采样的改进随机森林算法对于不平衡数据集较传统的重采样技术具有明显的优越性。




(a)G-mean 与 AUC 平均值比较　　(b)G-mean 与 AUC 排名比较

图 4 不同算法的 G-mean 和 AUC 对比

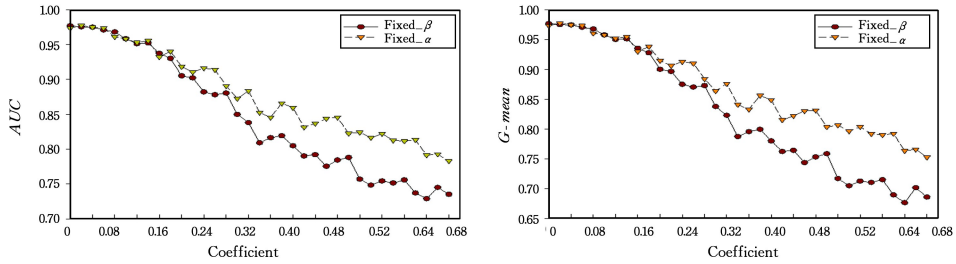
Fig. 4 Comparison of G-mean and AUC of different algorithms

### 4.3 HyCUD_RF 算法参数对性能的影响分析

HyCUD_RF 算法的核心是在构建随机森林子树时采用了融合级联上采样与下采样的重采样策略，级联过程中两次上采样量：高斯混合模型的上采样量和 SMOTE-Borderline1 算法的上采样量非常关键，由于 $\alpha=coef1*IR$，$\beta=coef2*IR$，显然 $coef1$ 和 $coef2$ 的变化将直接影响每种上采样的少数类样本数量。为探究 $coef1$ 和 $coef2$ 对算法性能的影响，本节以 Car-good 和 ecoli-0-1_vs_2-3-5 两个数据集为代表进行深入分析。

图 5 给出了 Car-good 数据集上的算法性能随参数的变化情况，图中的横轴表示 $coef1$ 和 $coef2$ 的变化，统一用 coefficient 表示。图 5(a)给出了 AUC 的变化情况，图中“Fixed_α”曲线表示固定 $coef1=0.1$，随后 $coef2$ 在 0～0.68 之间变化。可知，$coef2$ 在[0，0.06]区间时，本文算法的 *AUC* 值取得最大值，且波动不是很大；当 $coef2$ 增大时，*AUC* 呈线性下降趋势，这是因为当 $coef2$ 增大，基于 SMOTE-Borderline1 上

采样的少数样本就越多，由于上采样容易引入噪声样本，生成更多的少数类样本，引入噪声样本的可能性就越大，从而导致整个数据集的错分率越高，AUC 下降。图 5(a)中“Fixed_β”曲线表示固定 $coef2=0.1$，随后 $coef1$ 在 0～0.68 之间变化。该曲线呈现的趋势与“Fixed_α”曲线保持一致，原因也相同。但是，从图中可以发现，当 *coefficient* 在[0, 0.16]区间时，“Fixed_α”曲线与“Fixed_β”曲线的吻合度较高，而当 coefficient 大于 0.16 时，“Fixed_β”曲线的下降速率比“Fixed_α”曲线快，这表明增加高斯混合模型的上采样量比增加基于 SMOTE-Borderline1 的上采样量更容易造成 *AUC* 下降，这是因为采用高斯混合模型生成新样本时是基于各个高斯分量模型的均值和方差生成新的数据，由于高斯分量模型方差的存在，使得生成的数据超越了原来少数类样本的范围，显然，生成的新样本越多，则越多的样本可能超越原始样本范围，增加与多数类样本重叠的可能性，反而加大了数据可分难度，降低了 *AUC* 值。类似地，图 5(a)给出了 G-mean 的变化情况，可以发现，AUC 的变化趋势和 G-mean 的变化趋势保持一致。在此不再赘述。



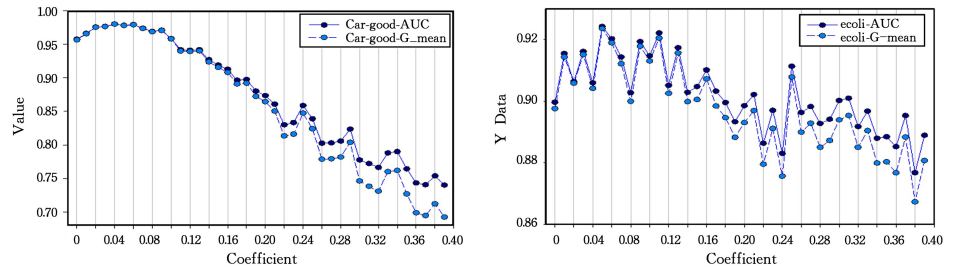

(a)AUC (b)G-mean

注：α 和 β 单独变化

图 5 Car-good 数据集上的算法性能随参数变化情况

Fig. 5 Performance changes of Car-good dataset with parameters

本文在 4.1 节和 4.2 节进行比较时设定 α 的 $coef1$ 和 β 的 $coef2$ 均为 0.1，为了探究 $coef1$，$coef2$ 同时变化对分类性能的影响，本文在 Car-good 和 ecoli-0-1_vs_2-3-5 两个数据集上进行实验。为了简化描述，实验过程中设定 $coef1$ 等于 $coef2$，且统一用 coefficient 表示，结果如图 6 所示。图 6 给出了当 *coefficient* 从 0 变化到 0.4 时两个数据集 AUC 和 G-mean 的变化情况。由图可知，在两个数据集中，当 *coefficient* 在[0.04, 0.06]区间时，*AUC* 和 *G-mean* 均取得最优值，比如 Car-good 的最优 *G-mean* 值为 0.9779，ecoli-0-1_vs_2-3-5 的最优 *G-mean* 值为 0.9235，均比表 4 和表 5 中的结果更好。以上结果说明，如果不生成少数类样本，分类算法向多数类倾斜，使得分类性能不佳；而增加少量的少数类样本有助于平衡数据集，避免算法向多数类倾斜，提升了数据集的可分性，使得 *G-mean* 较好。如果上采样量过大，引入的噪声样本较多，则 *G-mean* 下降。对比图 6(a)和图 6(b)可以发现，图6(a)的趋势更加稳定，而图 6(b)则波动较大，这是因为 ecoli 的少数类样本仅为 24，测试集中少数类样本仅为 8，误分 1 个少数类样本将会导致 *G-mean* 发生较大的变化，而 Car-good 中少数类样本为 84，受此影响较小。




(a)Car-good 数据集 (b)ecoli-0-1_vs_2-3-5 数据集

注：α 和 β 同时变化

图 6 不同数据集上算法性能随参数的变化情况

Fig. 6 Performance changes of different data sets with parameters

### 4.4 HyCUD_RF 与 State-of-art 算法的性能对比

为了将本文提出的 HyCUD_RF 算法与当前最新的研究成果进行对比，本节选择了与 2019－2020 年发表的 5 篇论文的原始结果进行比较。由于本文的测试数据集都是来自 UCI，KEEL，Kaggle 公开数据集，并不需要对数据进行额外的处理，故实验结果可以直接进行对比；另一方面，由于难以获得原始论文算法的各种具体参数值，而原文给出的一般都是该算法的最优值，因此直接对比论文给出的实验结果更具有比较意义。鉴于本文算法采用了聚类策略，我们选择将其与文献[14]进行对比，但是该文献并没有给出每个数据集的结果，因此本文通过运行其源码得到实验结果。

在优化重采样的策略方面，2017 年，Last 等对整个数据集进行 *K*-means 聚类，然后选择少数类较少的类簇进行上采样，构建了 K-means＋SMOTE[14] 采样策略。本文将对采样

后的数据集采用随机森林进行分类。2019 年，REN 等[30]对多数类和少数类采用 *K*-means 算法进行聚类，然后在聚类中心分别进行上下采用实现数据集平衡，最后使用 SVM 算法进行分类，构建 CPG 算法。

在采用新的分类器方面：为了获得不平衡数据分类的最优 *G-mean* 值，2020 年，Ri 等[31]通过构建新的极限学习机(Extreme Learning Machine，ELM)的损失函数，使得 ELM 可以用于不平衡分类算法，以此构建 GELM 算法。2020 年，Richhariya 等[32]充分利用数据分布的先验信息，将 universum learning 与支持向量机相结合，提出了简化的通用双支持向量机不平衡数据分类算法 RUTSVM-CIL。

在引入分类代价方面：2019 年，Ahmed 等[33]在每一次 boot 迭代过程中采用下采样进行重采样，同时为每一个实例附加代价项，并构建了 LIUBoost 算法。对于大数据量及高度不平衡的数据集，2020 年，Liu 等[34]引入了“分类硬度分布”的概念，并构建了 Self-paced Ensemble (SPE)不平衡数据分类框架。

各算法的对比结果如表 6 所列。由于每个算法所采用的数据集不完全相同，故本文仅仅从以上文献中摘取了与本文数据集相同的评价指标的值，对于原论文不存在的数据集或者指标在表 6 中用“—”标注。

表 6 本文算法与最新算法的性能比较

Table 6 Performance comparison between the proposed algorithm and state-of-the-art algorithms

| | 指标 | CPG | GELM | LIUBoost | RUTSVM-CIL | SPE | KmeanSmoteRF | HyCUD_RF |
|---|---|---|---|---|---|---|---|---|
| D1 | G-mean | — | **0.9802** | — | — | — | 0.9657 | 0.9754 |
| | AUC | — | — | — | — | — | 0.9658 | 0.9754 |
| D3 | G-mean | — | 0.6488 | — | 0.4874 | — | 0.5945 | **0.6533** |
| | AUC | — | — | 0.6470 | 0.6270 | — | 0.6096 | **0.6625** |
| D4 | G-mean | 0.9930 | — | — | — | — | 0.9934 | **0.9953** |
| | AUC | — | — | — | — | — | 0.9934 | **0.9953** |
| D6 | G-mean | — | 0.8916 | — | — | — | 0.9216 | **0.9545** |
| | AUC | — | — | **0.9880** | — | — | 0.9237 | 0.9546 |
| D7 | G-mean | — | **0.9397** | — | 0.5 | — | 0.8571 | 0.9130 |
| | AUC | — | — | — | 0.7 | — | 0.8683 | **0.9147** |
| D9 | G-mean | — | 0.9493 | — | — | — | 0.9753 | **0.9832** |
| | AUC | — | — | — | — | — | 0.9757 | 0.9833 |
| D12 | G-mean | — | **1** | — | — | — | 0.9453 | 0.9998 |
| | AUC | — | — | — | — | — | 0.95 | **0.9998** |
| D15 | G-mean | — | — | — | — | — | 0.2368 | **0.8645** |
| | AUC | — | — | 0.7920 | — | — | 0.5640 | **0.8784** |
| D17 | G-mean | — | — | — | — | 0.8550 | 0.8835 | **0.9395** |
| | AUC | — | — | — | — | — | 0.8904 | **0.9412** |

由表 6 可知，HyCUD_RF 在 D4 数据集上的 *G-mean* 值优于 CPG 算法。与 GELM 算法相比，HyCUD_RF 算法在 3 个数据集上均取得较优的结果。在与 LIUBoost 算法的对比中，有 3 个数据集相似，HyCUD_RF 在 2 个数据集上取得最优。HyCUD_RF 算法在 D3，D7 两个数据集上的 *G-mean* 和 *AUC* 值均优于 RUTSVM-CIL 算法，而 HyCUD_RF 算法在 D17 数据集上的 *G-mean* 值要优于 SPE 算法。尤其是与 KmeanSmoteRF 相比，HyCUD_RF 在所有数据集上均取得较好的成绩。综上，在总共 32 次对比中，HyCUD_RF 有 28 次取得的成绩都优于对比算法。因此本文算法与 state-of-the-art 算法相比显示出更好的性能。

**结束语** 处理好不平衡数据集对于精确检测出一些稀有事件非常重要。本文阐述了级联上采样的原理，并基于高斯混合模型逆权重上采样与 SMOTE-Borderline 上采样进行级联上采样，随后提出了一种融合级联上采样与下采样的改进随机森林不平衡数据分类算法。对比分析实验显示本文所提算法具有较好的优势，并得出以下结论：

(1) 级联上采样能够缓解少数类“小分离项”问题，能够较好地处理类内不平衡和类间不平衡问题。

(2)在 17 种数据集中，融合级联上采样与下采样的改进随机森林不平衡数据分类算法在与经典的重采样算法以及 state-of-the-art 算法的比较中均取得更优的成绩。

(3)在本文提出的算法中，少数类上采样样本数量对分类性能影响较大，当两次上采样系数 *coefficient* 均在[0.04，0.06]区间取值时，*G-mean* 和 *AUC* 取得最优值，随后 *coefficient* 增加，分类性呈下降趋势。

本文所提算法为二分类问题提供了一种有效的解决方案，但本文本质上是通过解决原始数据集的数据不平衡分布问题来提升算法的分类性能，因此本文提出的级联上采样＋下采样的混合采样策略能够推广至多分类场景，在后续的工作中可以针对多分类数据进行处理，再修改分类算法的源码以应用于多分类问题。其次，本文通过级联上采样和集成学习来减少生成的少数类噪声，后续拟继续研究如何采用更优的生成模型来减少少数类中的噪声生成量。

## 参 考 文 献


[1] RANDHAWA K，LOO C K，SEERA M，et al. Credit Card Fraud Detection Using AdaBoost and Majority Voting[J]. IEEE Access，2018，6：14277-14284.

[2] MOHD F，JALIL M A，NOORA N M M，et al. Improving Accuracy of Imbalanced Clinical Data Classification Using Synthetic Minority Over-Sampling Technique[C]// Advances in Data Science，Cyber Security and IT Applications. Cham：Springer International Publishing，2019：99-110.

[3] ZHANG Y，ZHANG H，ZHANG X，et al. Deep Learning Intrusion Detection Model Based on Optimized Imbalanced Network Data[C]// 2018 IEEE 18th International Conference on Communication Technology (ICCT). IEEE，2018：1128-1132.

[4] LIU P Z, HONG M, HUANG D T, et al. Joint ADASYN and AdaBoostSVM for Imbalanced Learining[J]. Journal of Beijing University of Technology, 2017, 43(3): 368-375.
[5] HUANG Y Y, LI Y J, GU M Y, et al. Learning from class-imbalanced data: Review of methods and applications[J]. Expert Systems with Applications, 2017, 73: 220-239.
[6] LIU Y, WANG Y, REN X, et al. A Classification Method Based on Feature Selection for Imbalanced Data[J]. IEEE Access, 2019, 7: 81794-81807.
[7] KHAN S H, HAYAT M, BENNAMOUN M, et al. Cost-sensitive learning of deep feature representations from imbalanced data[J]. IEEE Transactions on Neural Networks and Learning Systems, 2018, 29(8): 3573-3587.
[8] WOZNIAK M, KRAWCZYK B, SCHAEFER G. Cost-sensitive decision tree ensembles for effective imbalanced classification [J]. Applied Soft Computing, 2013, 14(1): 554-562.
[9] KRAWCZYK B, SCHAEFER G. An improved ensemble approach for imbalanced classification problems[C]// SACI 2013 - 8th IEEE International Symposium on Applied Computational Intelligence and Informatics. IEEE, 2013: 423-426.
[10] HAN H, WANG W Y, MAO B H. Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning[C]// International Conference on Intelligent Computing. Springer, 2005: 878-887.
[11] LEE J, PARK K. GAN-based imbalanced data intrusion detection system[J/OL]. Personal and Ubiquitous Computing. http://doi.org/10.1007/s00779-019-01332-y.
[12] WEISS G M. The Impact of Small Disjuncts on Classifier Learning[M]// Data Mining. Springer, Boston, MA, 2010: 193-226.
[13] BELLINGER C, SHARMA S, JAPKOWICZ N, et al. Framework for extreme imbalance classification: SWIM-sampling with the majority class[J]. Knowledge and Information Systems, 2020, 62: 841-866.
[14] LAST F, DOUZAS G, BACAO F. Oversampling for Imbalanced Learning based on K-Means and SMOTE[J]. arXiv: 1711.00837, 2017.
[15] LIN W C, TSAI C F, HU Y H, et al. Clustering-based undersampling in class-imbalanced data[J]. Information Sciences, 2017, 409: 17-26.
[16] CHEN G, LIU Y, GE Z. K-means Bayes algorithm for imbalanced fault classification and big data application[J]. Journal of Process Control, 2019, 81: 54-64.
[17] ZHAO N, ZHANG X F, ZHANG L J. Overview of Imbalanced Data Classification[J]. Computer Science, 2018, 45(6A): 22-27, 57.
[18] ZHENG J H, LIU S Y, HE C B, et al. Improved Random Forest Classification Algorithm for Imbalance Data Based on Hybrid Sampling Strategy[J]. Journal of Chongqing University of Technology(Natural Science), 2019, 33(7): 113-123.
[19] SHI H, GAO Q, JI S, et al. A Hybrid Sampling Method Based on Safe Screening for Imbalanced Datasets with Sparse Structure [C]// International Joint Conference on Neural Networks (IJCNN). 2018: 1-8.
[20] HAN X, JIA N, ZHU N. Gauss mixture undersampling algorithm for credit imbalance data[J]. Computer Engineering and Design, 2020, 41(1): 65-70.
[21] STAUFFER C, GRIMSON W E L. Adaptive background mixture models for real-time tracking[C]// Proceedings of IEEE Conf. Computer Vision Patt. Recog. 1999.
[22] ZHANG Y L, ZHOU Y J. A review of cluster algorithms[J]. Journal of Computer Applications, 2019, 39(7): 1869-1882.
[23] DEMPSTER A P, LAIRD N M, RUBIN D B. Maximum likelihood from incomplete data via the EM algorithm[J]. Journal of the Royal Statistical Society: Series B (Methodological), 1977, 39(1): 1-22.
[24] BHAGAT R C, PATIL S S. Enhanced SMOTE algorithm for classification of imbalanced big-data using Random Forest[C]// 2015 IEEE International Advance Computing Conference (IACC). 2015: 403-408.
[25] TAN X P, SU S J, HUANG Z P, et al. Wireless Sensor Networks Intrusion Detection Based on SMOTE and the Random Forest Algorithm[J]. Sensors, 2019, 19(1): 203.
[26] KUBAT M, HOLTE R, MATWIN S. Learning when negative examples abound[C]// European Conference on Machine Learning. Springer, 1997: 146-153.
[27] SEIFFERT C, KHOSHGOFTAAR T M, VAN HULSE J, et al. RUSBoost: A hybrid approach to alleviating class imbalance[J]. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 2010, 40(1): 185-197.
[28] Balanced Bagging Classifier imbalanced-learn 0.5.0 documentation[EB/OL]. [2020-02-18]. http://imbalanced-learn.org/stable/references/generated/imblearn.ensemble.BalancedBaggingClassifier.html.
[29] LIU X Y, WU J, ZHOU Z H. Exploratory undersampling for class-imbalance learning[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2009, 39(2): 539-550.
[30] REN H, YANG B. Clustering-Based Prototype Generation for Imbalance Classification[C]// 2019 International Conference on Smart Grid and Electrical Automation (ICSGEA). 2019: 422-426.
[31] RI J, KIM H. G-mean based extreme learning machine for imbalance learning[J]. Digital Signal Processing, 2020, 98: 102637.
[32] RICHHARIYA B, TANVEER M. A reduced universum twin support vector machine for class imbalance learning[J]. Pattern Recognition, 2020, 102: 107150.
[33] AHMED S, RAYHAN F, MAHBUB A, et al. LIUBoost: Locality Informed Under-Boosting for Imbalanced Data Classification [C]// Emerging Technologies in Data Mining and Information Security. Springer, 2019: 133-144.
[34] LIU Z N, CAO W, GAO Z F, et al. Self-paced Ensemble for Highly Imbalanced Massive Data Classification[J]. arXiv: 1909.03500, 2019.

**ZHENG Jian-hua**, born in 1977, Ph.D, associate professor, master supervisor. His main research interests include big data mining, machine learning and AI in smart agricultural.

**LI Xiao-min**, born in 1981, Ph.D, associate professor. His main research interests include cyber-physical systems, smart manufacturing, big data and wireless network.