文章编号:1671-9352(2026)05-0065-14 DOI:10.6040/j.issn.1671-9352.5.2025.006

基于快速超粒方生成算法的分类器模型

何怡[1],邵亚斌[1,2*],冯慧[1],郭瑞莲[1]
(1.重庆邮电大学数学与统计学院数学系, 重庆 400065; 2.网络空间大数据智能安全教育部重点实验室, 重庆 400065)

摘要:在空间划分时粒球计算方法存在半径敏感性、覆盖盲区与区域重叠缺陷等问题,本文提出基于 n 维超长方体的信息粒化方法。突破传统球形结构约束,采用 n 维超长方体几何模型,建立无盲区、无重叠的空间划分理论体系,提出快速超粒方生成(fast granular hypercube generation, FGHG)算法,通过维度自适应分割机制实现高效空间划分,与传统粒球生成算法相比,FGHG 算法在计算效率方面具有显著优势,设计快速超粒方分类器(fast granular hypercube classifier, FGHC)。为验证所提算法的有效性,选取 13 个真实数据集评估,FGHC 算法的分类精度和 $F1$ 分数均较高。本文建立的超粒方计算范式,为解决复杂数据空间划分问题提供新的理论框架。

关键词:粒计算;超粒方;信息粒化;分类器;粒球计算
中图分类号:TP311; TP309 **文献标志码**:A
引用格式:何怡,邵亚斌,冯慧,等. 基于快速超粒方生成算法的分类器模型[J]. 山东大学学报(理学版),2026,61(5):65-78.

A classifier model based on the fast granular hypercube generation algorithm

HE Yi[1], SHAO Yabin[1,2*], FENG Hui[1], GUO Ruilian[1]
(1. School of Mathematics and Statistics, Chongqing University of Posts and Telecommunications, Chongqing 400065, China; 2. Key Laboratory of Cyberspace Big Data Intelligent Security, Ministry of Education, Chongqing 400065, China)

Abstract: To address the problem of radius sensitivity in granular ball based spatial partitioning, which results in coverage gaps or region overlaps, an information granulation method is proposed based on n-dimensional hypercubes. The traditional constraint of spherical structures is overcome by introducing a novel n-dimensional hypercube geometric model, which establishes a theoretical framework for spatial partitioning without coverage gaps or overlapping regions. A fast granular hypercube generation (FGHG) algorithm is proposed, which utilizes a dimension-adaptive partitioning mechanism to enable efficient spatial division. Compared with traditional granular ball generation algorithms, FGHG algorithm demonstrates significant advantages in computational efficiency. A fast granular hypercube classifier (FGHC) is designed. To validate the effectiveness of the proposed algorithm, a systematic evaluation is conducted on 13 real-world datasets from the repository, where FGHC algorithm achieving improvements in both classification accuracy and $F1$ score. The granular hypercube computing paradigm established in this study provides a novel theoretical framework for tackling complex spatial partitioning problems in data analysis.

Key words: granular computing; granular hypercube; information granulation; classifier; granular ball computing

0 引言

粒计算(granular computing, GrC)[1]是一种应对复杂性、不确定性和大规模数据处理问题的计算范式。当人类处理大量复杂信息时,由于认知能力有限,往往会根据信息的特征将其划分为几个较为简单的部分,每个被划分的部分称为一个粒。这种处理信息的过程被称为信息粒化[2-3]。

在 GrC 中,信息粒的粒度直接影响算法的效率和抗噪声能力:较粗的粒度虽然提升执行速度与鲁棒性,但是可能影响决策结果或降低分类精度,而过细的粒度将每个数据点都作为最小计算单元,显著增加计算复

收稿日期:2025-05-23; **网络出版时间**:2026-03-12
基金项目:国家自然科学基金资助项目(12061067,62176033); 重庆市自然科学基金面上资助项目(CSTB2023NSCQ-MSX0707)
第一作者:何怡(2000—),女,硕士研究生,研究方向为机器学习、数据分析的不确定方法等. E-mail:karinahee@163.com
*通信作者:邵亚斌(1974—),男,教授,博士,研究方向为模糊分析学、粗糙集与粒计算、数据分析的不确定方法等. E-mail:shaoyb@cqupt.edu.cn

杂度[4-5],Xia 等[6]提出一种鲁棒的信息粒表示方式,即粒球(granular ball, GB)。在分类任务中,先使用粒球生成(granular ball generation, GBG)算法生成具有多粒度的粒球,然后使用粒球替代样本点,有效地降低数据集的维度和规模[7-8]。同时,GBG 算法与传统的信息粒化算法相比,时间复杂度显著降低。

尽管粒球在诸多领域中有广泛应用前景[9-10],但仍面临一些关键性的挑战。首先,粒球对数据集的覆盖效果有限,往往无法包含所有样本点,导致部分数据未能有效参与后续的分析或建模过程。其次,粒球之间可能存在重叠,尤其在边界区域,这种重叠使某些样本点同时被多个标签不同的粒球包含,引发类别混淆,影响分类或聚类的准确性。最后,粒球的生成结果差异较大,表现出不稳定性,难以保证多次实验中的一致性和可重复性。

首先,球的半径的选择导致粒球覆盖效果不佳和重叠问题:若采用样本点到球心的最大距离作为球的半径导致大量的粒球重叠问题[11-12];若采用样本点到球心的平均距离作为半径,导致粒球覆盖效果不佳,离球心远的数据点无法被覆盖在粒球内[13-14]。在这种情况下,粒球无法表示较远的数据点,甚至当作噪声点,影响粒球在后续应用中的性能。其次,粒球生成结果的不稳定性,由于粒球划分算法(如 k-means、k-division 算法[15-16])具有一定的随机性,例如,在迭代过程中,初始中心点的随机选取导致粒球数量和分布均不同,从而降低算法的可重复性与泛化能力[17-18]。

本文提出一种新的信息粒化算法,将 n 维超长方体结构引入信息粒化过程中,生成的信息粒不仅能完整覆盖整个数据集,而且无重叠现象,从而有效提升信息粒化的效率。在此基础上,进一步提出一种新的分类器。与传统的基于粒球计算的分类器相比,本文提出的分类器具有更强的适应性,且应对多种数据分布情形,分类精度和 $F1$ 分数提升较高。

本文提出“超粒方”的概念,定义中心点、边长、标签与纯度。为保证超粒方在后续划分与更新的稳定性,本文引入 Lipschitz 连续性条件,证明超粒方的中心点和边长满足 Lipschitz 连续性条件,确保超粒方生成过程的连续性与稳定性。针对传统的信息粒化算法存在覆盖不完全和重叠问题,设计一种快速高效的超粒方生成(fast granular hypercube generation, FGHG)算法。FGHG 算法在“造粒”过程中避免粒子间的重叠,有效降低在高维空间中生成结构粒的复杂度。实验结果表明,FGHG 算法在多个公开数据集上粒化效率高和耗费时间短,适用于大规模数据场景下的快速建模需求。在快速超粒方生成算法的基础上,提出快速超粒方分类器(fast granular hypercube classifier, FGHC),FGHG 利用粒子与待测样本点之间的空间关系判断样本归类,针对未被覆盖的样本引入局部近邻补偿机制,提升分类边界的准确性。实验表明,FGHC 的分类精度和 $F1$ 分数高于传统的粒球分类器,在不平衡数据、噪声数据等复杂场景中也具有更强的稳定性和泛化能力。

1 相关工作

1.1 粒球生成

常见的粒球生成算法有原始的 GBG 算法[16]和加速的 GBG 算法[16],原始的 GBG 算法的核心思想是用球形结构覆盖数据点,此球形结构称为粒球。

定义 1[6] D 是一个包含 n 个样本点的数据集,$D=\{D_1,D_2,\cdots,D_i\}$, $1\leqslant i\leqslant n'$,其中 D_i 是 D 的子集,包含 N_i 个数据点,其中 $N_1+N_2+\cdots+N_i=n$。G 是由 D 生成的粒球集合,$G=\{g_1,g_2,\cdots,g_i\}$, $1\leqslant i\leqslant n$,其中 g_i 是由 D_i 生成的粒球。每个粒球中心 c_i 是 D_i 中所有数据点 x_{ij} 的质心,半径 r_i 是 D_i 中所有数据点到中心点 c_i 的平均距离,$d(x_{ij},c_i)$ 为样本点 x_{ij} 到中心点 c_i 的欧氏距离。中心点 c_i 和半径 r_i 定义分别为

$$c_i=\frac{1}{n_i}\sum_{x_{ij}\in D_i}x_{ij}, \tag{1}$$

$$r_i=\frac{1}{n_i}\sum_{x_{ij}\in D_i}d(x_{ij},c_i)。 \tag{2}$$

定义 2[6] 给定一个数据集 D,对任意的 $g_j\in G$,有 $g_i=\{(x,y)\}$,其中 x 是 g_j 中的数据点,y 是 x 的标签。g_j 的标签 L_j 和纯度 P_j 定义分别为

$$L_j=\arg\max\{y\mid(x,y)\in g_j\}, \tag{3}$$

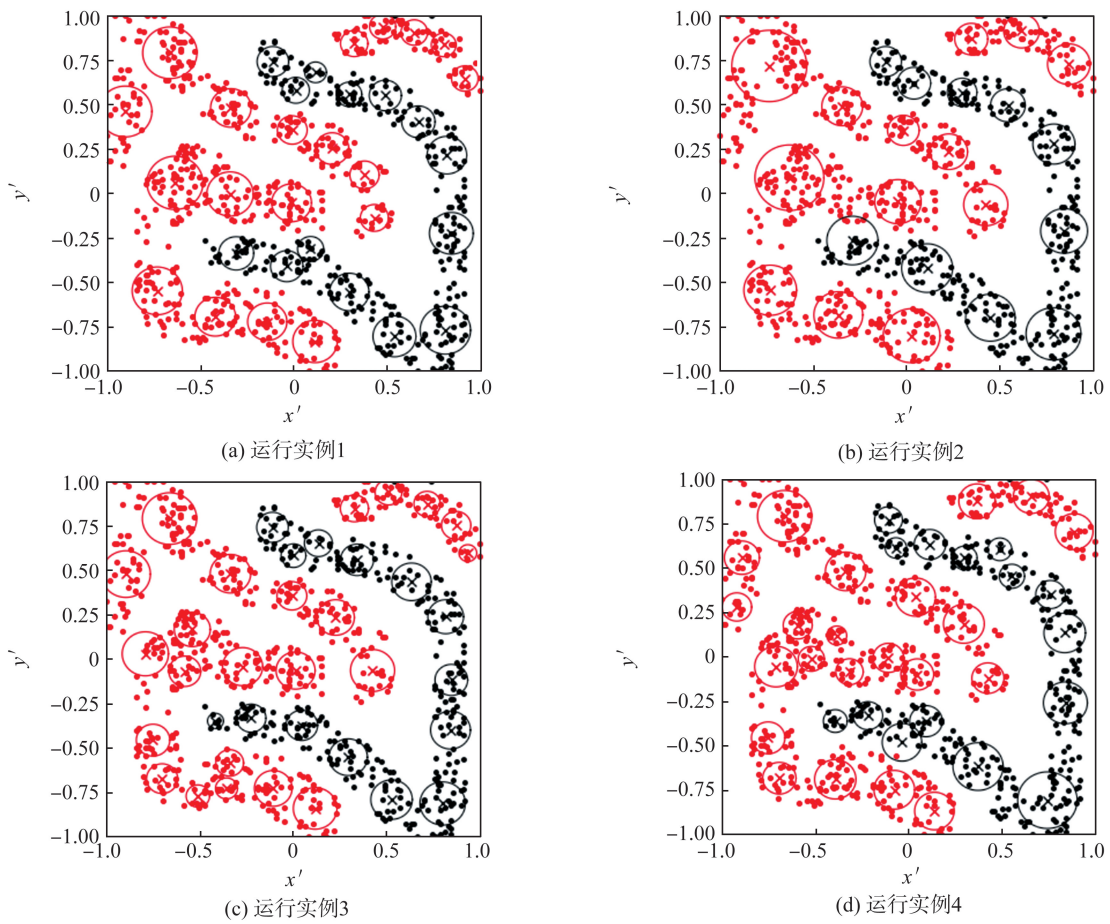
$$P_j=\frac{|g_{L_j}|}{|g_j|},\tag{4}$$

式中:$|g_{L_j}|$是 g_j 中标签为 L_j 的数据点的个数,$|g_j|$是 g_j 中所有数据点的个数。

对于一个初始数据集,将整个数据集视为一个初始粒球,采用 *k*-means 算法将初始粒球分割成更细小的粒球,再检查每个粒球的纯度是否达到设定的纯度,纯度越接近 1,则粒球对原始数据集的覆盖效果越好。如果所有粒球的纯度都符合要求,则流程结束;如果粒球的纯度未达标,则流程回到分裂的步骤,继续划分,直至所有粒球的纯度都达到预先设定好的纯度。这个方法通过循环执行分裂和检验步骤,逐步提高数据的分类准确性。

类似地,加速的 GBG 算法同样先将整个数据集作为一个初始粒球,在此基础上,该算法随机选择若干异类样本为中心,通过 *k*-division 算法将粒球进一步细分。每次划分后,检查每个粒球的纯度是否达到设定纯度,未达标则继续划分。若所有粒球纯度符合要求,进行一次全局划分,结束流程。在加速的 GBG 算法中,粒球的中心点、半径、标签和纯度的定义与原始的 GBG 算法相同。

在相同参数设置下,分别采用 *k*-means(图 1(a)、(b))与 *k*-division(图 1(c)、(d))算法分别在 Fourclass 数据集上生成粒球的示意图。可以看出,无论是使用 *k*-means 还是 *k*-division 算法,粒球的生成结果都存在明显的差异。首先,存在覆盖不完全的现象,部分样本点未被任何粒球包含,特别是在样本分布边缘区域,这种现象较为严重,导致部分数据在后续建模中被忽略。其次,粒球之间存在较为严重的重叠,导致某些样本点同时被多个粒球覆盖,且这些粒球可能对应不同的类别标签,引发标签冲突问题。此外,从同一算法的两次运行结果(图 1(a)、(b)、(c)、(d))中可以看出,即使在相同的参数条件下,生成的粒球数量、位置和覆盖结构也存在明显差异,说明现有方法具有较强的随机性,缺乏稳定性和可重复性。这些问题共同反映现有粒球生成算法在覆盖性、分离性和稳定性方面的不足,难以满足高质量信息粒结构的实际需求,文中 x、y 分别为粒球的位移。



(a) 运行实例1　(b) 运行实例2

(c) 运行实例3　(d) 运行实例4

图 1　在 Fourclass 数据集上生成的粒球示意图

Fig.1　Schematic of the granular ball generated on the Fourclass dataset

在后续的研究中,为了解决上述随机中心选择和计算量大的问题,Xie 等[19]提出基于注意力机制的粒球生成算法,即 GBG++算法,通过局部离群点检测与冲突消解显著提升效率与稳定性;同时还提出 GBKNN++(granular ball k nearest neighbor++, GBKNN++)分类算法,在预测阶段,将球心与样本距离与粒球纯度加权投票,并配合冲突消解策略给出最终标签,提升对噪声与类不平衡的鲁棒性与分类精度。

1.2 粒球分类器

在机器学习领域,分类任务是最常见的问题之一,分类任务的本质是将实例数据归入预定义的类别。例如 k-最近邻(k-nearest neighbor, KNN)分类器[20]基于最近邻的原则,测量新数据点与训练集中每个点之间的距离,选择距离最近的 k 个数据点,并根据这些邻近点的类别通过投票方式分类。这种算法直观且易于实现,但随着数据量的增加,特别是在特征维度高的情况下,计算距离的成本显著提高。目前还有很多分类器是基于信息粒化的思想对数据点分类[21-22],首先数据集被分解为一系列较小的、信息丰富的信息粒,每个粒代表数据集中的一个集群或子集。这些粒可以是基于相似性、功能、位置或任何其他有意义的标准构建的。基于这些粒化后的数据,分类器通过学习不同信息粒之间的区别进行模式识别和数据分类。

Xia 等[15]将粒球分类器(granular ball classifier, GBC)的思想引入 KNN 分类器,利用粒球的几何特性,计算数据点到每个粒球球心的距离,选择最近的球心的标签作为数据点的标签,提升分类器的准确性。随后,为了解决 KNN 分类器中的 k 值需要预先设定的问题,Li 等[21]引入自然邻居的概念,提出基于自然邻居局部中心搜索的粒球 KNN(local optima ridge exploration granular ball KNN, LOREGBKNN)算法,利用数据点之间的邻居关系,自适应的得到 k,提升算法的稳定性。Xia 等[22]进一步提出粒球支持向量机(granular-ball support vector machine, GBSVM)分类器,将支持向量机(support vector machine, SVM)与粒球计算结合,利用数据的多粒度特性,提高分类器的稳健性和效率。Xie 等[23]提出多粒度邻域关系来优化 KNN 分类器,并提出多粒度邻域 KNN(multi-granularity neighbor for KNN, MGKNN)分类器。该算法自适应地为每个数据点确定邻居大小,利用粒球模型计算。这种算法解决传统 KNN 的局限性,即固定的值无法适应具有不同密度和分布的数据集。Ganaie 等[24]为解决多类分类器在面对噪声数据时鲁棒性不足且计算开销高的问题,提出粒球 k 类孪生支持向量机(granular ball k-class twin support vector classifier, GB-TWKSVC),将粒球计算与孪生支持向量机相结合,以粒球中心及半径代替原始样本点进行多类划分,在多个基准数据集上同时实现高于现有算法的分类准确率和提高运算速度。值得注意的是,粒计算中出现了一些基于不同粒化表示的分类方法,如旋转粒支持向量机分类器[25],该方法从旋转粒构造的角度将粒化思想与支持向量机的结合,为粒计算在分类任务中的应用提供新思路。

2 快速超粒方生成算法及分类器

由于现有的粒球计算方法在覆盖效果与稳定性上存在局限,如何在不降低计算效率的前提下,更有效地覆盖数据空间、更精准地表达数据特征,成为粒计算领域亟需解决的关键问题。受此启发,本文提出一种基于 n 维超长方体的信息粒化表示方法。与传统的球形信息粒相比,n 维超长方体的几何结构具有更强的空间划分能力与覆盖能力,生成的超粒方有效解决 GBC 中粒球重叠和覆盖不完全问题。此外,超粒方的生成过程采用确定性的递归划分策略,提高算法的稳定性和可重复性。

2.1 超粒方的定义及其数学模型

结合超长方体的几何特性,即在各个维度空间中完整的划分数据集且不会产生任何重叠,构造超粒方模型。接下来,给出超粒方的定义及其数学模型。

设 D_i 是 D 的一个子集,$D_j=\{X^{(1)},X^{(2)},\cdots,X^{(n)}\}$,每个数据点 $X^{(t)}=(x_1^{(t)},x_2^{(t)},\cdots,x_m^{(t)})\in\mathbf{R}^m$, m 是数据集的维数,n 为数据集 D 中样本点的数量。数据点 $X\in D$ 的标签记作 $l(X)$。

定义 3 由数据集 D 构成的初始超粒方 H_0,将 H_0 定义为数据集在各维度取值范围的笛卡尔积,即

$$H_0=\prod_{p=1}^{m}[a_p,b_p],\quad p=1,2,\cdots,m, \tag{5}$$

式中:$a_p=\min\limits_{1\leqslant q\leqslant n}x_p^{(q)}$, $b_p=\max\limits_{1\leqslant q\leqslant n}x_p^{(q)}$ 是第 p 维的上界和下界。

定义 4 对于任意一个超粒方 $H=\prod\limits_{s=1}^{m}[a_s,b_s]$,对应的数据子集 $D(H)$ 定义为

$$D(H)=\{x^{(s)}\in D\mid x^{(s)}\in[a_s,b_s],\ s\in\mathbf{N}\}。\tag{6}$$

定义 5　对于任意一个超粒方 H,其中心点 c 与各个维度上的边长 d 定义为

$$c=\left(\frac{a_1+b_1}{2},\frac{a_2+b_2}{2},\cdots,\frac{a_m+b_m}{2}\right),\tag{7}$$

$$d=\{b_1-a_1,b_2-a_2,\cdots,b_m-a_m\}。\tag{8}$$

定义 6　超粒方 H 的标签 L_H 定义为其对应的数据子集 $D(H)$ 中出现次数最多的标签,即

$$L_H=\arg\max_{l_k}\{x\in D(H)\mid l(x)=l_k\},\tag{9}$$

式中 l_k 是 $D(H)$ 中所有样本点的标签,$k\in\mathbf{N}$。

定义 7　超粒方 H 的纯度 P_H 定义为对应的数据子集 $D(H)$ 中标签为 L_H 的数据点所占的比例,即

$$P_H=\frac{\max\limits_{l_k}|\{x\in D(H)\mid l(x)=l_k\}|}{|D(H)|},\tag{10}$$

式中$|\cdot|$表示其内部的数据点个数。

当超粒方 H 的纯度没有达到一定纯度时,将针对其标准差最大的一维进行二等分。假设将在第 v 维上对超粒方 H 进行划分,划分后的子超粒方的定义如下。

定义 8　对于超粒方 $H=\prod\limits_{u=1}^{m}[a_u,b_u]$,以第 v 维的中点 $c_v=\dfrac{a_v+b_v}{2}$将 H 划分为 2 个新的子超粒方,即

$$H_1=\left(\prod_{\substack{u=1\\u\neq v}}^{m}[a_u,b_u]\right)\times[a_v,s],\tag{11}$$

$$H_2=\left(\prod_{\substack{u=1\\u\neq v}}^{m}[a_u,b_u]\right)\times[s,b_v]。\tag{12}$$

任意一个超粒方 $H=\prod\limits_{o=1}^{m}[a_o,b_o]$都可以由中心点和边长唯一确定,相比于传统的、以单个数据点作为基本计算单位的方法,以超粒方作为输入单位可以减少后续算法的数据输入规模和运行的时间。

接下来将评估超粒方的几何参数对输入数据扰动的敏感度,以验证本文算法的稳健性。定理 1 说明当数据集发生小幅波动时,重新计算得到的超粒方中心点和边长仅在可控范围内变化,且该变化与扰动幅度呈线性关系。具体而言,对于一个给定的超粒方,其中心点的边长对数据点的扰动是 Lipschitz 连续的,若数据集 D 发生小幅度扰动,则重新计算得到的中心点和边长的变化有上界,且这种变化与扰动幅度成线性关系。

定理 1　设 $U=\{X^1,X^2,\cdots,X^n\}\in\mathbf{R}^m$为超粒方中的数据点集合,其中 $X^e=(x_1^e,x_2^e,\cdots,x_m^e)$, $1\leqslant e\leqslant n$,对于第 f 维,有

$$c_f=\frac{1}{n}\sum_{e=1}^{n}x_f^e,$$

$$d_f=\max_{1\leqslant e\leqslant n}x_f^e-\min_{1\leqslant e\leqslant n}x_f^e。$$

设 U 受到扰动,得到新的数据点集合

$$U'=\{X^1+\Delta X^1,X^2+\Delta X^2,\cdots,X^n+\Delta X^n\}\in\mathbf{R}^m,$$

其中,每个扰动都满足$\|\Delta X^e\|\leqslant\delta$, $\forall 1\leqslant e\leqslant n$,则对于第 f 个维度,记新的中心和边长为

$$c'_f=\frac{1}{n}\sum_{e=1}^{n}(x_f^e+\Delta x_f^e),$$

$$d'_f=\max_{1\leqslant e\leqslant n}(x_f^e+\Delta x_f^e)-\min_{1\leqslant e\leqslant n}(x_f^e+\Delta x_f^e),$$

则 $\forall\delta>0$,存在常数 K_1、K_2,使得

$$\|\vec{c'}-\vec{c}\|\leqslant K_1\delta,$$

$$\|d'_i-d_i\|\leqslant K_2\delta,$$

即满足 Lipschitz 连续性。

证明　(1) 中心点的稳定性。

对于维度 f,有

$$c'_f-c_f=\frac{1}{n}\sum_{e=1}^{n}\Delta x_f^e,$$

因此有

$$|c'_f-c_f|\leqslant\frac{1}{n}\sum_{e=1}^{n}|\Delta x_f^e|\leqslant\frac{1}{n}\sum_{e=1}^{n}\|\Delta x_f^e\|\leqslant\delta。$$

对于中心点向量 $\vec{c}=(c_1,c_2,\cdots,c_m)$ 和 $\vec{c'}=(c'_1,c'_2,\cdots,c'_m)$,有

$$\|\vec{c'}-\vec{c}\|=\sqrt{(c'_1-c_1)^2+(c'_2-c_2)^2+\cdots+(c'_m-c_m)^2},$$

式中 $|c'_f-c_f|\leqslant\delta$,有 $(c'_f-c_f)^2\leqslant\delta^2$,所以

$$\|\vec{c'}-\vec{c}\|=\sqrt{\sum_{f=1}^{m}(c'_f-c_f)^2}\leqslant\sqrt{m\cdot\delta^2}=\sqrt{m}\delta。$$

取 $K_1=\sqrt{m}$,有

$$\|\vec{c'}-\vec{c}\|\leqslant K_1\delta。$$

(2) 边长的稳定性。

考虑第 f 维上的最值,设

$$M_f=\max_e x_f^e,\quad m_f=\min_e x_f^e,$$

扰动后,新的最大值为 $M'_f=\max_e(x_f^e+\Delta x_f^e)$,新的最小值为 $m'_f=\min_e(x_f^e+\Delta x_f^e)$,满足

$$M'_f\leqslant M_f+\delta,\quad m'_f\geqslant m_f-\delta。$$

新的边长 $d'_f=M'_f-m'_f$,满足

$$d'_f\leqslant(M_f+\delta)-(m_f-\delta)=(M_f-m_f)+2\delta=d_f+2\delta。$$

同理可得

$$M'_f\geqslant M_f-\delta,\quad m'_f\leqslant m_f+\delta,$$

于是有

$$d'_f\geqslant(M_f-\delta)-(m_f+\delta)=(M_f-m_f)-2\delta=d_f-2\delta。$$

综上,有 $d_f-2\delta\leqslant d'_f\leqslant d_f+2\delta$,即 $|d'_f-d_f|\leqslant2\cdot\delta$。

对于 $\vec{d}=(d_1,d_2,\cdots,d_m)$ 和 $\vec{d'}=(d'_1,d'_2,\cdots,d'_m)$,有

$$\|d'_i-d_i\|\leqslant\sqrt{m}(2\delta)=2\sqrt{m}\delta。$$

取 $K_2=2\sqrt{m}$,有

$$\|d'_i-d_i\|\leqslant K_2\delta。$$

2.2 快速超粒方生成算法

超粒方的生成过程本质上是一个不断细分数据空间以满足信息粒纯度要求的过程。然而,传统的信息粒化算法通常存在信息粒重叠较多、覆盖效果不佳、以及稳定性不足等问题。本节提出一种快速、稳定的超粒方生成算法,该算法通过确定性划分策略,在保证信息粒纯度的同时,也显著提高算法的效率和稳定性。

本文算法首先将整个数据集视为一个初始超粒方,通过计算纯度,判断初始超粒方是否满足设定的纯度。当超粒方的纯度未达到要求时,算法将选择标准差最大的维度,并在该特征上取中点对超粒方进行一分为二的递归划分,随后继续检查子超粒方的纯度,直至所有超粒方均达到纯度为止。具体流程如算法 1 所示。

算法 1 快速超粒方生成算法。

输入 数据集 D,纯度 P;

输出 超粒方集合 H_{list}。

(1) 初始化 $H_{\text{list}}\leftarrow0$;

(2) 将整个数据集 D 视作一个初始的超粒方,记作 H_0;

(3) 根据定义 5—7 计算初始超粒方 H_0 的中心点 c_0,边长 d_0,标签 L_0 和纯度 P_0,并将 H_0 加入 H_{list};

(4) Repeat

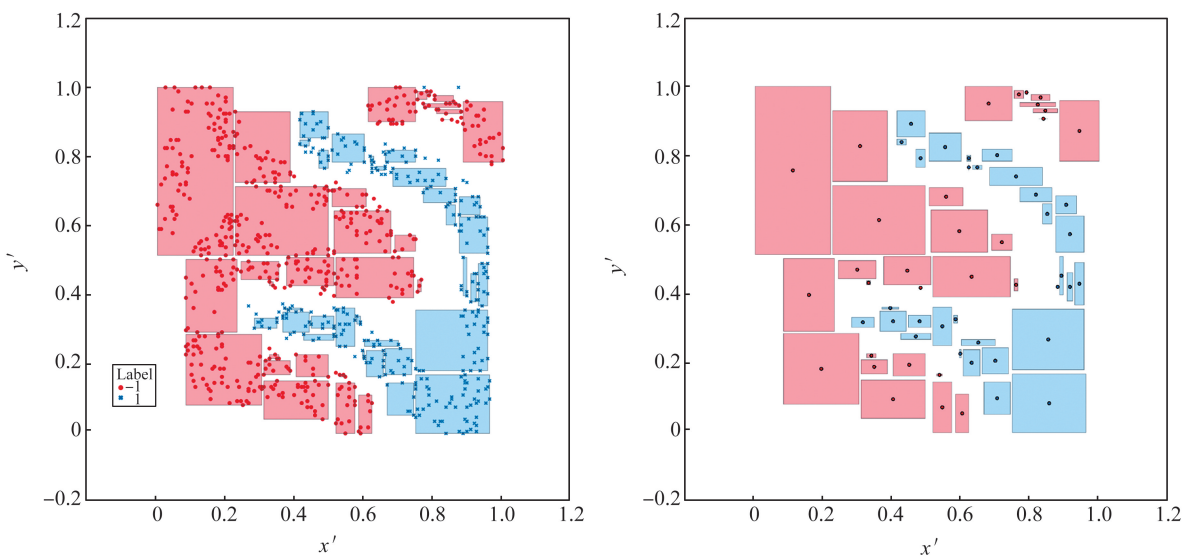
(5) For $H_i\in H_{\text{list}}$ do

(6) 根据定义 5 计算超粒方 H_i 的中心点和边长;
(7) 根据定义 6、7 计算超粒方 H_i 的标签和纯度;
(8)　If $P_{H_i}<P$ then
(9)　　选择标准差最大的一维对超粒方 H_i 进行二等分;
(10)　　根据定义 5 计算新的超粒方的中心点和边长;
(11)　　将子超粒方加入集合 H_{list};
(12)　End If
(13) End For
(14) Until H_{list}中所有超粒方的纯度均满足纯度 P;
(15) 输出超粒方集合 H_{list}。

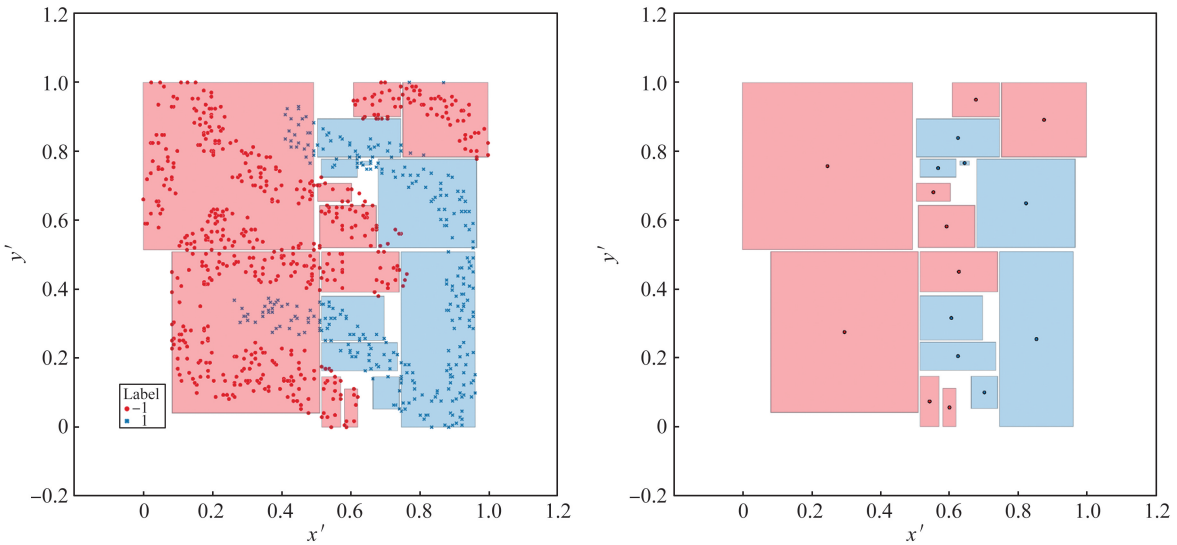
为便于评估算法 1 在高维与大样本条件下的可扩展性,下面给出时间复杂度推导。设数据集中的样本数为 n、维度为 m、划分的层数为 L。在每一层中,各粒方的样本互不重叠。对任一粒方 H 计算中心、半径与纯度并据某一维做二等分判定的代价为 $O(|D(H)|m)$,通过累加得到时间复杂度 $O(nm)$,总时间复杂度为 $O(nmL)$。由此可见,复杂度对维度 m 呈线性增长而非指数增长。高维时“维度灾难”的影响主要体现在纯度增益变小,可能使 L 略有增加,但不改变 m 的线性关系。

图 2 是算法 1 在 Fourclass 数据集上生成的超粒方示意图。图 2(a)、(b)为当纯度设置为 1 时的数据分布,(c)、(d)为当纯度设置为 0.8 时的数据分布。通过图 2 可知,当纯度为 0.8 时,生成的超粒方数量较少且体积较大,覆盖大量的样本空间。然而,由于每个超粒方内部允许 20%的异类样本混入,后续的分类任务重,容易造成若干边界样本被误划分。当纯度为 1 时,超粒方数量激增且粒度显著变细,保证内部纯净度,直接分类的误差几乎消失。但是覆盖范围却下降。导致后续的分类任务中大量样本落在超粒方之间的空隙地带,这些样本的分类结果更易受相邻样本分布的影响而出现波动。2 种纯度的对比直观地揭示超粒方在覆盖率、纯度与分类稳定性之间的相互制约关系。同时可以看出,超粒方能够紧密的贴合数据集,完整的展示数据的分布形式,极大的减少数据量,为后续的分类任务提高效率。

如图 2 所示,提高纯度会产生数量更多、粒度更细的超粒方,内部标签一致性更高;降低纯度则得到数量更少、粒度更粗的超粒方,但允许异类混入,导致一致性下降。当类别不平衡时,较高的纯度(如 0.99)会对混合区域要求更苛刻的停止条件,若进一步细分以更好分离少数类,但同时增加超粒方的数量与算法运行时间;而较低的纯度(如 0.95)则更早停止,可抑制多数类区域的过度切分,但可能降低对极小少数类的分离能力。针对类别不平衡,文本采用统一且不过拟合的纯度策略:默认纯度为 0.97;若纯度在 0.97 时出现粒方数量急剧增多或计算量大,则纯度为 0.95 以抑制过度划分;为增强少数类分离效果(尤其在极端不平衡情形),图 1(a)给出纯度为 1 的数据分布,作为高纯度上界的参考。上述选择不依赖标签信息,仅依据算法内部量(粒方数量与运行时间)进行折中,并在实验中对 3 种纯度做并列敏感性测试。



(a) 纯度为1时数据分布　　(b) 纯度为1时超粒方分布



(c) 纯度为0.8时数据分布　　(d) 纯度为0.8时超粒方分布

图 2　在 Fourclass 数据集上生成的超粒方示意图

Fig.2　Schematic of the granular hypercube generated on the Fourclass dataset

2.3　基于快速超粒方生成算法的分类器

本节进一步提出快速超粒方分类器(fast granular hypercube classifier, FGHC)。FGHC 以粒化后的超粒方作为基本输入单位,从整体结构描述数据空间。这种做法不仅能够显著减少数据输入规模,还能有效抑制噪声点的影响,提高分类的稳定性和准确性。

利用算法 1 得到的高纯度超粒方集合,通过判断待分类数据点与超粒方的几何位置关系,快速确定所属类别。当某个数据点位于某个超粒方的内部时,直接利用该超粒方的标签,当该数据点未被任何超粒方覆盖时,则借助高效的近邻搜索结构,快速确定最可能的类别标签。具体流程详见算法 2。

算法 2　快速超粒方分类器。

输入　超粒方集合 H_{list},待分类的数据点 X;

输出　数据点 X 的标签 $l(X)$。

(1) For $H_i \in H_{\text{list}}$ do

(2)　If $X \in D(H_i)$ then

(3)　　$l(X)=l(H_i)$;

(4)　Else

(5)　$l(X)$设定为最近的 k 个样本点的标签中的大多数类;

(6) End For

(7) 返回数据点 X 的标签 $l(X)$。

在算法 2 中,若待测样本点未落入任何一个已知的超粒方中,则在训练集中进行 k 近邻检索并基于多数投票原则决定样本点的类别,其中 k 默认设定为数据集中的类别数量。算法 2 首先以欧氏距离为度量,计算待测样本与所有训练样本之间的距离,并快速返回距离最小的 k 个邻居;随后统计这 k 个邻居的真实标签出现频率,将出现频率最大的标签作为该样本的预测结果。此投票机制既能有效填补超粒方覆盖范围的空白,又能融合局部样本分布的信息,确保边界或稀疏区域的分类决策更加稳定,提高整体分类的一致性和鲁棒性。

3　实验结果及分析

为了验证 FGHG 算法与 FGHC 的有效性,实验分别从超粒方生成效率和分类性能展开,再与现有的粒球计算方法对比分析。此外,由于纯度是粒生成过程中的重要参数,设计不同纯度的对比实验,以验证不同的纯度对生成效率、粒的数量和分类性能的影响。本文选取 13 个公开数据集评估,不仅包括常用的基准集,还有面向真实业务场景的结构化数据集,例如 Wisconsin diagnostic breast cancer(WDBC)数据集与 Credit 数据

集,White blood cell(WBC)数据集用于乳腺癌良恶性诊断与临床决策支持,Credit 数据集用于信用风险评估。

所有实验在统一的预处理流程下检验本文所提算法在真实场景中的适用性与稳健性,数据集的信息见表 1 所示。本文实验配置为:内存为 16GB DRAM 和 Intel(R),处理器为 Core(TM) i5-6500。实验的所有算法都在 Python 3.11.7 中使用 PyCharm 2024.1 开发环境,确保实验一致性。

表 1　数据集信息

Table 1　Informations of data set

数据集	样本数量/个	特征数量/个	数据集	样本数量/个	特征数量/个
Cancer	683	9	Sensorreadings	5 456	24
Fourclass	862	2	WBC	683	9
Heart1	294	13	Wine2	178	13
Iris	150	4	Zoo	101	16
Letter	20 000	16	Credit	690	16
PenDigits	10 992	16	WDBC	569	31
Phoneme	5 404	5			

3.1　信息粒化时间对比

为了验证 FGHG 算法的生成效率和造粒有效性,与基于 *k*-division 的 GBG 算法、基于自然邻居的局部最优中心搜索的粒球生成(local optima ridge exploration granular ball generation, LOREGBG)算法对比。为评估生成效率与造粒有效性,在纯度为 0.95、0.97、0.99 对每个数据集,依次运行算法生成信息粒,记录信息粒化的时间和数量,在相同纯度与相同数据划分条件下完成对照,保证可比性。

表 2 展示了信息粒化的时间对比。FGHG 在所有数据集上信息粒化时间最少。随着纯度提高,信息粒化的时间增加,而 FGHG 算法的增幅在大多数据集上最小,出现部分基线算法未能在该数据集下生成有效粒球。其中,在 Heart1 数据集、Wine2 数据集与 Phoneme 数据集上,LOREGBG 算法均未产生有效粒球;在 Zoo 数据集上,GBG 算法未产生有效粒球。主要原因是这些数据集在高纯度要求下类别分布不均导致算法分裂出大量单样本或空簇,无法满足生成条件。值得关注的是,在 PenDigits 数据集(纯度为 0.99)上,FGHG 算法的信息粒化的时间为 3.127 s,而 GBG 算法信息粒化的时间为 33.994 s、LOREGBG 算法信息粒化的时间为 6 975.449 s,时间分别下降约 90.8%与 99.96%;在 Letter 数据集(纯度为 0.99)上,FGHG 算法信息粒化的时间为 274.651 s,而 GBG 算法信息粒化的时间为 2 115.656 s、LOREGBG 算法信息粒化的时间为 10 784.122 s,分别下降约 87.0%与 97.5%。FGHG 算法的生成阶段不进行迭代聚类(如 *k*-division)也无法构建自然邻居图,计算量随样本与维度近似线性增长,因此在大样本、多类别数据上信息粒化的时间更少。

表 2　不同纯度、不同粒生成算法信息粒化的时间对比

Table 2　Generation time of information-granule algorithms with varying purity thresholds　　单位:s

数据集	纯度为 0.95			纯度为 0.97			纯度为 0.99		
	GBG	LOREGBG	**FGHG**	GBG	LOREGBG	**FGHG**	GBG	LOREGBG	**FGHG**
Cancer	41.444	332.615	**2.361**	40.247	332.364	**4.253**	40.696	468.553	**4.812**
fourclass	32.630	41.956	**0.122**	30.348	42.933	**0.176**	34.605	47.529	**0.162**
heart1	136.915		**13.779**	138.106		**15.050**	140.612		**14.004**
iris	12.755	62.006	**0.040**	12.877	61.993	**0.042**	13.340	62.239	**0.042**
letter	2 145.351	10 626.833	**188.203**	2 074.111	10 637.291	**228.887**	2 115.656	10 784.122	**274.651**
PenDigits	34.360	6 931.383	**0.168**	17.418	6 945.261	**1.494**	33.994	6 975.449	**3.127**
phoneme	1 004.450		**29.548**	1 003.600		**32.928**	1 006.484		**36.480**
sensorReadings	1 050.699	1 761.909	**9.764**	1 105.144	1 817.984	**10.483**	1 042.081	1 895.127	**12.592**
WBC	39.483	325.994	**2.277**	38.205	325.102	**3.654**	38.826	325.584	**5.210**
wine2	14.048		1.178	9.823		1.301	13.813		1.344
zoo		24.458	**3.453**		24.312	**3.863**		24.667	**3.979**
credit	308.788	398.747	**11.868**	299.858	398.966	**12.378**	316.275	401.384	**13.680**
WDBC	71.804	154.798	**0.179**	59.033	160.238	**0.175**	80.637	166.284	**0.245**
平均时间	408	1 878	**20**	402	1 886	**24**	406	1 923	**28**

为了进一步展示 FGHG 算法的粒化有效性,本文还对比 3 种算法在不同纯度下生成信息粒的数量,具体结果如表 3 所示。从表 3 中发现,在相同纯度条件下,FGHG 算法生成的超粒方数量明显少于 GBG 算法生成的粒球数量,这表明 FGHG 算法的粒化结果更加紧凑。例如,在 Sensorreading 数据集中,GBG 算法产生超过 1 500 个粒球,而 FGHG 算法显著减少超粒方的数量。这种更紧凑、高效的信息粒分布,能够有效提高后续数据处理和分析的准确性与稳定性。

表 3 不同纯度、不同粒生成算法生成信息粒的数量对比

Table 3 Comparison of the number of information granules generated at various purity thresholds in different algorithms

数据集	纯度为 0.95			纯度为 0.97			纯度为 0.99		
	GBG	LOREGBG	FGHG	GBG	LOREGBG	FGHG	GBG	LOREGBG	FGHG
Cancer	52	58	10	47	58	10	52	61	19
Fourclass	43		38	40		38	43		46
Heart1	178	28	20	177	31	20	178	34	20
Iris	19	29	9	18	29	9	19	36	9
Letter	2 721	3 768	416	2 583	3 777	416	2 721	416	477
PenDigits	45	554	26	22	558	26	45	26	46
Phoneme	1 307		315	1 242		315	1 307	1 333	361
Sensorreadings	1 600	781	109	1 586	816	109	1 600	109	139
WBC	52	58	10	48	58	10	52	10	19
Wine2	19		7	13		7	19		13
Zoo		21	3		21	3		3	3
Credit	384	129	27	383	129	40	383	132	46
WDBC	94	84	13	73	88	13	69	96	19

3.2 分类精度

为对 FGHC 的性能进行全面评估,本文采用 GBKNN、GBKNN++、LOREGBKNN 算法以及 KNN、SVM、朴素贝叶斯(naive bayes, NB)、PLKNN 分类器[20] 作为对比算法。由于 FGHC、GBKNN、GBKNN++和 LOREGBKNN 算法都要设定纯度,分别比较这 4 种算法不同纯度的分类精度,最后将不同纯度下的平均分类精度与 4 种基于经典机器学习的分类器做比较。

从表 4—7 的结果可见,FGHC 在多数数据集上都表现出较高的分类精度。在 Wine2 数据集上,FGHC 的分类精度 100%,在 PenDigits 数据集上,分类精度为 0.998 6。当纯度为 0.99 时,FGHC 在 Iris 数据集上为 100%的分类精度。相比之下,GBKNN 算法和 LOREGBKNN 算法在同样的测试条件下,分类精度仍略有不足,例如在 Heart1 数据集上,GBKNN 和 LOREGBKNN 算法的分类精度分别为 0.566 4 和 0.666 7,而 FGHC 的分类精度为 0.694 9。

表 4 当纯度为 0.95 时,不同分类器的分类精度比较

Table 4 Accuracy comparison of different classifiers with the value of purity 0.95

数据集	GBKNN	GBKNN++	LOREGBKNN	**FGHC**
Cancer	0.922	0.909	0.897	**0.971**
Fourclass	0.987	**1.000**	0.994	0.994
Heart1	0.566	0.690	0.667	**0.695**
Iris	0.912	0.929	0.924	**1.000**
Letter	0.966	0.949	**0.975**	0.968
PenDigits	0.989	0.998	0.986	**0.999**
Phoneme	0.873	0.837	**0.891**	0.883
Sensorreadings	0.873	0.829	0.887	**0.908**
WBC	0.922	0.909	0.964	**0.971**
Wine2	0.955	**1.000**	0.944	**1.000**
Zoo		**0.800**		0.762
Credit	0.646	0.652	0.725	**0.797**
WDBC	0.898	0.964	0.965	**0.974**
平均分类精度	0.876	0.882	0.902	**0.917**

表 5　当纯度为 0.97 时,不同分类器的分类精度比较

Table 5　Accuracy comparison of different classifiers with the value of purity 0.97

数据集	GBKNN	GBKNN++	LOREGBKNN	**FGHC**
Cancer	0.922	0.932	0.897	**0.971**
Fourclass	0.987	0.988	**0.994**	**0.994**
Heart1	0.566	0.690	0.667	**0.695**
Iris	0.912	0.929	0.924	**1.000**
Letter	0.966	0.949	**0.975**	0.968
PenDigits	0.989	0.996	0.986	**0.999**
Phoneme	0.873	0.843	**0.891**	0.883
Sensorreadings	0.873	0.828	0.887	**0.908**
WBC	0.922	0.932	0.964	**0.971**
Wine2	0.955	0.944	0.944	**1.000**
Zoo		**0.762**		**0.762**
Credit	0.646	0.551	0.725	**0.797**
WDBC	0.898	0.929	0.970	**0.974**
平均分类精度	0.876	0.867	0.902	**0.917**

表 6　当纯度为 0.99 时,不同分类器的分类精度比较

Table 6　Accuracy comparison of different classifiers with the value of purity 0.99

数据集	GBKNN	GBKNN++	LOREGBKNN	**FGHC**
Cancer	0.922	0.932	0.897	**0.971**
Fourclass	0.987	0.988	**0.994**	**0.994**
Heart1	0.566	0.633	0.667	**0.695**
Iris	0.912	0.933	0.924	**1.000**
Letter	0.966	0.941	**0.975**	0.968
PenDigits	0.989	0.997	0.986	**0.999**
Phoneme	0.873	0.856	**0.891**	0.883
Sensorreadings	0.873	0.846	0.887	**0.908**
WBC	0.922	0.911	0.964	**0.971**
Wine2	0.955	0.944	0.944	**1.000**
Zoo		**0.762**		**0.762**
Credit	0.642	0.551	0.754	**0.797**
WDBC	0.905	0.946	0.970	**0.974**
平均分类精度	0.876	0.865	0.904	**0.917**

表 7　不同数据集不同算法的分类精度对比

Table 7　Comparison of classification accuracy of different algorithms on different datasets

数据集	KNN	SVM	NB	GBKNN	LOREGBKNN	PLKNN	GBKNN++	FGHC
Cancer	0.963	0.964	0.956	0.911	0.915	0.949	0.924	**0.966**
Fourclass	**0.998**	0.757	0.746	0.992	0.994	0.837	0.992	0.994
Heart1	0.722	0.746	**0.797**	0.566	0.667	0.576	0.671	0.695
Iris	0.947	0.933	0.867	0.915	0.945	0.967	0.930	**1.000**
Letter	0.975	0.729	0.736	0.969	0.975	0.731	0.946	0.970
PenDigits	0.999	0.990	0.975	0.993	0.989	0.961	0.997	**0.999**
Phoneme	0.875	0.759	0.744	0.879	**0.905**	0.803	0.845	0.883
Sensorreadings	0.857	0.720	0.511	0.874	0.891	0.752	0.834	**0.907**
WBC	0.966	0.964	0.956	0.911	0.961	0.964	0.917	**0.966**
Wine2	0.969	0.972	0.944	0.955	0.962	0.944	0.963	**1.000**
Zoo	0.800	**0.850**	0.700			**0.850**	0.775	0.762
Credit	0.830	**0.841**	0.768	0.645	0.734	0.826	0.585	0.797
WDBC	0.968	0.974	0.904	0.900	0.968	0.904	0.946	**0.974**
平均分类精度	0.913	0.861	0.816	0.876	0.909	0.851	0.871	**0.916**

3.3 *F*1 分数

为了进一步说明 FGHC 的优势,本文对比不同纯度下的 FGHC 算法、GBKNN 算法、GBKNN++算法和 LOREGBKNN 算法二分类数据集的 *F*1 分数,其中 Iris 数据集和 Sensorreadings 数据集为多分类数据集,所以在本实验中没有出现。同时与 3 种基于机器学习的分类器比较平均 *F*1 分数。

从表 8—11 可以看出,FGHC 的 *F*1 分数同样展现显著的优势。例如,在 Cancer 数据集、PenDigits 数据集以及 WBC 数据集上,FGHC 的 *F*1 分数始终保持在较高水平,进一步说明该算法在实际分类任务中具有稳定且优异的综合性能。特别是在 PenDigits 数据集中,FGHC 的 *F*1 分数为 0.998 6(纯度为 0.95),而基于粒球的其他分类器则略低一些。可以推断,在更高维度或更复杂分布的数据集上,FGHC 依然能够兼顾分类精度与稳定性。

表 8 当纯度为 0.95 时,不同分类器的 *F*1 分数对比

Table 8 *F*1 score comparison of different classifiers with the value of purity 0.95

数据集	GBKNN	GBKNN++	LOREGBKNN	**FGHC**
Cancer	0.921	0.909	0.905	**0.971**
Fourclass	0.986	**1.000**	0.988	0.994
Heart1	0.541	**0.676**	0.667	0.663
Letter	0.966	0.949	0.965	**0.968**
PenDigits	0.969	0.998	0.981	**0.999**
Phoneme	0.848	0.792	0.878	**0.879**
WBC	0.921	0.909	0.964	**0.971**
Wine2	0.950	**1.000**	0.944	**1.000**
Zoo		**0.762**		0.721
Credit	0.638	0.369	0.729	**0.797**
WDBC	0.889	0.961	0.965	**0.974**
平均值	0.863	0.848	0.899	**0.903**

表 9 当纯度为 0.97 时,不同分类器的 *F*1 分数对比

Table 9 *F*1 score comparison of different classifiers with the value of purity 0.97

数据集	GBKNN	GBKNN++	LOREGBKNN	**FGHC**
Cancer	0.907	0.931	0.914	**0.963**
Fourclass	0.992	0.987	**0.994**	**0.994**
Heart1	0.541	**0.676**	0.658	0.663
Letter	0.969	0.949	**0.976**	0.970
PenDigits	0.986	0.990	0.991	**0.999**
Phoneme	0.856	0.806	**0.896**	0.879
WBC	0.907	0.931	0.954	**0.963**
Wine2	0.950	0.943	0.953	**1.000**
Zoo		**0.800**		0.721
Credit	0.638	0.313	0.727	**0.797**
WDBC	0.907	0.922	0.933	**0.974**
平均值	0.865	0.841	0.900	**0.902**

表 10 当纯度为 0.99 时,不同分类器的 *F*1 分数对比

Table 10 *F*1 score comparison of different classifiers with the value of purity 0.99

数据集	GBKNN	GBKNN++	LOREGBKNN	**FGHC**
Cancer	0.900	0.931	0.904	**0.963**
Fourclass	0.993	0.987	0.994	**0.994**
Heart1	0.541	0.597	**0.678**	0.663
Letter	0.972	0.941	**0.985**	0.972
PenDigits	0.990	0.993	0.991	**1.000**
Phoneme	0.859	0.824	0.871	**0.879**

表 10(续)

数据集	GBKNN	GBKNN++	LOREGBKNN	**FGHC**
WBC	0.900	0.911	0.960	**0.963**
Wine2	0.950	0.940	0.970	**1.000**
Zoo		**0.762**		0.721
Credit	0.633	0.313	0.768	**0.797**
WDBC	0.894	0.943	0.973	**0.974**
平均值	0.863	0.831	**0.909**	0.902

表 11　不同数据集、不同算法的 $F1$ 分数对比

Table 11　Comparison of $F1$ score of different algorithms on different datasets

数据集	KNN	SVM	NB	GBKNN	LOREGBKNN	PLKNN	GBKNN++	FGHC
Cancer	0.963	**0.975**	0.969	0.910	0.908	0.944	0.924	0.966
Fourclass	**0.994**	0.604	0.621	0.990	0.992	0.824	0.991	**0.994**
Heart1	**0.716**	0.571	0.700	0.541	0.667	0.366	0.649	0.663
Letter	**0.975**	0.739	0.757	0.969	0.976	0.720	0.946	0.970
PenDigits	0.999	0.995	0.986	0.982	0.988	0.907	0.993	**0.999**
Phoneme	0.874	0.575	0.617	0.854	**0.882**	0.769	0.807	0.879
WBC	0.966	**0.975**	0.969	0.910	0.959	0.959	0.917	0.966
Wine2	0.969	0.960	0.917	0.950	0.956	0.944	0.961	**1.000**
Zoo	0.800	0.769	0.625			0.850	0.775	0.721
Credit	0.830	**0.849**	0.814	0.637	0.741	0.797	0.331	0.797
WDBC	0.968	0.981	0.933	0.897	0.957	0.899	0.942	**0.974**
平均值	**0.914**	0.818	0.810	0.864	0.903	0.816	0.840	0.902

综上所述,FGHC 算法在分类精度和 $F1$ 分数两方面都与现有基于粒球的分类器相比具有明显优势,因此适合作为高效、准确的分类器应用于多种实际问题场景。

4　结论

本文提出基于 n 维超长方体的信息粒化算法。该算法依赖于数据集的原始结构,在中心点选择过程中不再采用随机算法,而是在更为合理的基础上选择中心点。这一改进不仅提升超粒方生成的有效性,也增强整个算法的稳定性和可靠性。本文还提出快速超粒方分类器。与传统分类器不同,FGHC 不须要计算每个数据点到中心点的距离,而是只须判断数据点是否位于超粒方内部。这一设计大幅度简化分类的过程,提高计算速度,使得 FGHC 在实际应用中能够更快地响应。实验结果表明,FGHC 在分类精度和时间效率方面普遍优于基于粒球的分类器。尽管 FGHC 在分类精度和效率方面表现出色,但仍然存在一些不足之处。例如,在部分数据集上纯度设置过大会增加一些不必要的计算。因此,未来的研究可以着重于如何优化超粒方生成的过程,以降低纯度对其生成的影响。本文仅提供超粒方生成的基本框架,未来的研究有望将超粒方扩展到更多现有算法中,例如图像处理、属性约简、聚类分析等领域。这将为粒计算的应用提供更广泛的可能性,同时也为相关算法的进一步优化和改进提供新的研究方向。

参考文献:

[1] YAO Jingtao, ATHANASIOS V, WITOLD P. Granular computing: perspectives and challenges[J]. IEEE Transactions on Cybernetics, 2013, 43(6):1977-1989.

[2] 王国胤,张清华,胡军. 粒计算研究综述[J]. 智能系统学报,2007,2(6):8-26.
WANG Guoyin, ZHANG Qinghua, HU Jun. An overview of granular computing[J]. CAAI Transactions on Intelligent Systems, 2007, 2(6):8-26.

[3] 张清华,王宇泰,赵凡. 复杂问题求解的多粒度计算框架[J]. 中国科学:信息科学,2025,55(5):1122-1139.
ZHANG Qinghua, WANG Yutai, ZHAO Fan. Multi-granularity computing framework for complex problem solving[J].

Scientia Sinica Informations, 2025, 55(5):1122-1139.
[4] ZADEH L A. Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic[J]. Fuzzy Sets and Systems, 1997, 90(2):111-127.
[5] SARKAR M. Fuzzy-rough nearest neighbor algorithms in classification[J]. Fuzzy Sets and Systems, 2007, 158(19):2134-2152.
[6] XIA Shuyin, ZHENG Shaoyuan, WANG Guoyin, et al. Granular ball sampling for noisy label classification or imbalanced classification[J]. IEEE Transactions on Neural Networks and Learning Systems, 2021, 34(4):2144-2155.
[7] QUADIR A, TANVEER M. Granular ball twin support vector machine with pinball loss function[J]. IEEE Transactions on Computational Social Systems, 2024: 1-10.
[8] SAJID M, QUADIR A, TANVEER M, et al. GB-RVFL: fusion of randomized neural network and granular ball computing [J]. Pattern Recognition, 2025, 159:111142.
[9] 华有霖,邵亚斌,朱学勤. 基于粒球计算的多粒度支持向量回归算法[J]. 山东大学学报(理学版),2025,60(7):1-12.
HUA Youlin, SHAO Yabin, ZHU Xueqin. Multi-granularity support vector regression algorithm based on granular ball computing[J]. Journal of Shandong University(Natural Science), 2025, 60(7):1-12.
[10] 薛任煊,伊士超,王平心. GBDEN:一种基于粒球的大规模数据快速聚类方法[J]. 计算机科学,2024,51(12):166-173.
XUE Renxuan, YI Shichao, WANG Pingxin. GBDEN: a fast clustering algorithm for large-scale data based on granular ball [J]. Computer Science, 2024, 51(12):166-173.
[11] PENG Xiaoli, WANG Ping, XIA Shuyin, et al. VPGB: a granular-ball based model for attribute reduction and classification with label noise[J]. Information Sciences, 2022, 611:504-521.
[12] CHENG Dongdong, LI Ya, XIA Shuyin, et al. a fast granular-ball-based density peaks clustering algorithm for large-scale data[J]. IEEE Transactions on Neural Networks and Learning Systems, 2024, 35(12):17202-17215.
[13] GILET C, BARBOSA S, FILLATRE L. Discrete box-constrained minimax classifier for uncertain and imbalanced class proportions[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2022, 44(6):2923-2937.
[14] WU Chengying, ZHANG Qinghua, YIN Longjun, et al. Data-driven interval granulation approach based on uncertainty principle for efficient classification[J]. IEEE Transactions on Fuzzy Systems, 2023, 32(1):12-26.
[15] XIA Shuyin, LIU Yunsheng, DING Xin, et al. Granular ball computing classifiers for efficient, scalable and robust learning [J]. Information Sciences, 2019, 483:136-152.
[16] XIA Shuyin, DAI Xiaochuan, WANG Guoyin, et al. An efficient and adaptive granular-ball generation method in classification problem[J]. IEEE Transactions on Neural Networks and Learning Systems, 2022, 35(4):5319-5331.
[17] CHENG Dongdong, ZHANG Cheng, LI Ya, et al. GB-DBSCAN: a fast granular-ball based dbscan clustering algorithm[J]. Information Sciences, 2024, 674:120731.
[18] SHAO Yabin, HUA Youlin, GONG Zengtai, et al. CON-MGSVM: controllable multi-granularity support vector algorithm for classification and regression[J]. Information Fusion, 2025,117:102867.
[19] XIE Qin, ZHANG Qinghua, XIA Shuyin, et al. GBG++: a fast and stable granular ball generation method for classification [J]. IEEE Transactions on Emerging Topics in Computational Intelligence, 2024, 8(2):2022-2036.
[20] JODAS D, PASSOS L, ADEEL A, et al. PL-KNN: a python-based implementation of a parameterless *K*-nearest neighbors classifier[J]. Software Impacts, 2023, 15:100459.
[21] LI Chen, SHAO Yabin, XIA Shuyin, et al. An adaptive granular ball classifier based on natural neighbor[C]//Proceedings of the 2023 8th International Conference on Mathematics and Artificial Intelligence. New York: ACM, 2023:47-52.
[22] XIA Shuyin, LIAN Xiaoyu, WANG Guoyin, et al. GBSVM: an efficient and robust support vector machine framework via granular-ball computing[J]. IEEE Transactions on Neural Networks and Learning Systems, 2024, 36(5):9253-9267.
[23] XIE Jiang, XIANG Xuexin, XIA Shuyin, et al. MGNR: a multi-granularity neighbor relationship and its application in KNN classification and clustering methods[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2024, 46(12): 7956-7972.
[24] GANAIE M, VRUSHANK A, ANOUCK G. Granular ball K-class twin support vector classifier[J]. Pattern Recognition, 2025, 116:111636.
[25] 邓波军,吴南海,陈玉明,等. 旋转粒支持向量机分类器算法[J]. 山东大学学报(理学版),2026,61(5):102-113.

(编辑:陈丽萍)