计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2023,32(6):121−129 [doi: 10.15888/j.cnki.csa.009108]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于粒度粗糙熵与改进蜂群算法的特征选择①

孙雅芝, 江　峰, 杨志勇

(青岛科技大学 信息科学与技术学院, 青岛 266061)
通信作者: 江　峰, E-mail: jiangfeng@qust.edu.cn

摘　要: 经典的人工蜂群 (artificial bee colony, ABC) 算法面临着收敛速度慢、易陷入局部最优等不足, 因此基于该算法来进行特征选择还存在很多问题. 对此, 提出了一种基于粒度粗糙熵与改进蜂群算法的特征选择方法 FS_GREIABC. 首先, 将粗糙集中的知识粒度与粗糙熵有机地结合起来, 提出一种新的信息熵模型——粒度粗糙熵; 其次, 将粒度粗糙熵应用于 ABC 算法中, 提出一种基于粒度粗糙熵的适应度函数, 从而获得了一种新的适应度计算策略; 第三, 为了提高 ABC 算法的局部搜索能力, 将云模型引入到跟随蜂阶段. 在多个 UCI 数据集以及软件缺陷预测数据集上的实验表明, 相对于现有的特征选择算法, FS_GREIABC 不仅能够选择较少的特征, 而且具有更好的分类性能.

关键词: 知识粒度; 粒度粗糙熵; 云模型; 人工蜂群算法; 特征选择

引用格式: 孙雅芝,江峰,杨志勇.基于粒度粗糙熵与改进蜂群算法的特征选择.计算机系统应用,2023,32(6):121–129. http://www.c-s-a.org.cn/1003-3254/9108.html

Feature Selection Based on Granularity of Knowledge Rough Entropy and Improved Artificial Bee Colony Algorithm

SUN Ya-Zhi, JIANG Feng, YANG Zhi-Yong

(College of Information Science & Technology, Qingdao University of Science and Technology, Qingdao 266061, China)

Abstract: The classical artificial bee colony (ABC) algorithm is also faced with slow convergence speed, and it is easy to fall into local optimality, so there are still many problems in feature selection based on this algorithm. Therefore, a feature selection method based on the rough entropy of granularity and an improved bee colony algorithm, namely FS_GREIABC, is proposed. Firstly, a new information entropy model, namely the rough entropy of granularity, is proposed by combining the knowledge granularity and the rough entropy in the rough set. Secondly, the rough entropy of granularity is applied to the ABC algorithm, and a fitness function based on the rough entropy of granularity is proposed, so as to obtain a new fitness calculation strategy. Thirdly, in order to improve the local search ability of the ABC algorithm, a cloud model is introduced into the following bee stage. Experiments on multiple UCI datasets and software defect prediction datasets show that FS_GREIABC not only selects fewer features but also has better classification performance than the existing feature selection algorithms.

Key words: granularity of knowledge; granularity of knowledge rough entropy; cloud model; artificial bee colony (ABC) algorithm; feature selection

近年来, 随着信息技术的飞速发展, 现实生活中出现了非常多具有大量特征的数据集, 从而引发了高维数据的处理问题. 高维数据的处理方法可以分为特征提取和特征选择[1], 其中, 特征选择的主要目的是从原

① 基金项目: 国家自然科学基金 (61973180, 61671261); 山东省自然科学基金 (ZR2022MF326)
收稿时间: 2022-11-07; 修改时间: 2022-12-10; 采用时间: 2023-01-06; csa 在线出版时间: 2023-04-17
CNKI 网络首发时间: 2023-04-18

始特征集中选择出使得评估指标最佳的特征子集. 特征选择保留了数据的原始特征, 具有良好的可解释性, 因此得到了广泛应用.

特征选择本质上是一个组合优化问题, 因此, 很多学者利用各类智能优化算法来解决特征选择问题. Hancer 等[2] 提出了一种基于改进蜂群算法的特征选择方法, 该方法将遗传算子和非支配排序过程引入到蜂群算法并应用到特征选择中. Mashhour 等[3] 提出了一种基于萤火虫算法和卡方的特征选择方法, 该方法通过用每次移动后改变的细胞卡方值模拟萤火虫位置, 并通过计算一组不同的适应度函数作为特征权重来模拟萤火虫强度进而应用到特征选择中. 孙林等[4] 提出了一种基于改进帝王蝶优化算法的特征选择方法, 该方法使用差分进化算法中的变异操作替换帝王蝶算法的迁移算子, 其次融入了自适应调整策略, 对每次种群进行柯西变异, 然后引入 KNN 分类器构造适应度函数, 最终获得最优特征子集.

作为一种处理不确定性的有效工具, 粗糙集理论[5,6] 在特征选择方面也可以发挥重要的作用. 很多学者利用粗糙集理论来进行特征选择. 王锋等[7] 提出了一种组增量式的粗糙特征选择算法. 江峰等[8] 利用粗糙集的近似精度来定义近似决策熵, 并提出了一种基于近似决策熵的特征选择算法. 董红斌等[9] 提出了一种基于关联信息熵的特征选择算法. 王鹏等[10] 提出了一种基于改进模糊决策粗糙集的最小化决策代价特征选择算法. 然而, 基于粗糙集的特征选择方法还存在计算效率低、性能难以满足要求等问题. 针对这些问题, 有不少学者将粗糙集理论与智能优化算法结合在一起来进行特征选择的相关研究, 从而可以有效发挥这两类方法各自的优势. Tawhid 等[11] 提出了基于粗糙集与二进制鲸鱼优化的特征选择算法. Azar 等[12] 提出了一种基于粗糙集和灰狼算法的特征选择方法. Chen 等[13] 将鱼群算法与粗糙集结合在一起来进行特征选择. 高薇等[14] 提出了一种基于粗糙集与人工蜂群算法的动态特征选择方法.

人工蜂群 (artificial bee colony, ABC) 算法作为一种经典的群智能优化算法已被广泛应用于特征选择问题. 人工蜂群算法是由土耳其教授 Karaboga 等[15,16] 受蜜蜂蜂群采蜜机制的启发而提出的. 该算法运算简单、具有较好的探索能力且参数较少, 因此引起了大量的关注. 但是, 收敛速度较慢、容易陷入局部最优等仍然是该算法在实际应用中面临的问题.

针对人工蜂群算法的不足, 本文将粗糙集理论[6] 引入到人工蜂群算法中, 并由此提出一种基于粒度粗糙熵与改进蜂群算法的特征选择算法 FS_GREIABC. FS_GREIABC 算法的主要思路如下: 首先, 针对人工蜂群算法中的适应度函数构建问题, 将粗糙集中的知识粒度与粗糙熵有机地结合在一起, 从而得到一种新的信息熵模型——粒度粗糙熵. 粒度粗糙熵中采用到的粗糙熵能够有效度量知识的完备性, 但却不能有效度量知识的粒度大小, 而知识粒度可以度量知识的粒度大小, 两者有机地结合起来能够更加全面地对信息表中的属性进行度量, 为后续的特征选择提供了一种更加全面的度量机制; 其次, 人工蜂群算法的关键步骤之一就是如何为算法选择一个合适的适应度函数, 我们提出了一种基于粒度粗糙熵的适应度函数, 针对特征选择中两个相互冲突的目标是必须完成所选特征的最小数目和最大的分类精度, 我们利用粒度粗糙熵来构造适应度函数能够更好地对特征子集进行评估; 第三, 为了提高人工蜂群算法的局部搜索能力, 考虑到云模型具有稳定性和随机性等优势[17,18], 因此, 将云模型引入到跟随蜂阶段, 并采用一维正态云模型来调整局部搜索范围, 以增强算法的局部开采能力, 提高蜂群算法的鲁棒性和收敛速度.

本文的主要贡献如下: (1) 提出了一种新的信息熵模型——粒度粗糙熵. 粒度粗糙熵将粗糙集中的知识粒度与粗糙熵这两个基本概念结合在一起, 既考虑了知识的不完备性, 又刻画了知识的粒度大小, 是一种更加全面的不确定性度量机制. (2) 提出了一种基于粒度粗糙熵的适应度函数, 通过粒度粗糙熵来计算每只蜜蜂的适应度值, 从而为人工蜂群算法提供一种新的适应度计算策略. (3) 为了提高 ABC 算法的局部搜索能力, 将云模型引入到人工蜂群算法的跟随蜂阶段, 采用一维正态云模型来改进跟随蜂阶段的搜索策略.

1 相关技术介绍

1.1 粗糙集的相关概念

在粗糙集中, 信息表[6] 是一个四元组$S=(U,A,V,f)$, 其中 U 表示非空有限的对象集; A 表示非空有限的属性集; V 表示所有属性值域的并集, 即$V=\bigcup_{a\in A}V_a$ (V_a 是属性a的值域); $f:U\times A\to V$ 表示一个函数, 使得对

任意$a \in A$以及$x \in U$, 都有$f(x,a) \in V_a$.

给定一个信息表$S=(U,A,V,f)$, 对于任意属性子集$B \subseteq A$, 由属性子集 B 所确定的一个不可分辨关系如下:

$$IND(B)=\{(x,y) \in U \times U : f(x,a)=f(y,a), \forall a \in B\}$$

其中, $IND(B)$是论域U上的一个等价关系, 它把U划分成不同的等价类, 所有等价类的集合就构成了U上的一个划分, 记为$U/IND(B)$.

定义 1. 知识粒度[19]. 给定一个信息表$S=(U,A,V,f)$, 对于任意属性子集$B \subseteq A$, 令$U/IND(B)=\{X_1,X_2,\cdots,X_t\}$表示关系$IND(B)$对论域$U$的划分. B的知识粒度$GK(B)$被定义为:

$$GK(B)=\frac{\sum_{i=1}^{t}|X_i|^2}{|U|^2}$$

其中, 对任意$1 \leqslant i \leqslant t$, $|X_i|$表示集合X_i的势, $|U|$表示集合U的势.

可以证明, 对任意$B \subseteq A$, $\frac{1}{|U|} \leqslant GK(B) \leqslant 1$.

定义 2. 粗糙熵[20]. 给定一个信息表$S=(U,A,V,f)$, 对于任意属性子集$B \subseteq A$, 令$U/IND(B)=\{X_1,X_2,\cdots,X_t\}$表示关系$IND(B)$对论域$U$的划分. B的粗糙熵$E_r(B)$被定义为:

$$E_r(B)=-\sum_{i=1}^{t}\frac{|X_i|}{|U|}\log_2\frac{1}{|X_i|} \tag{1}$$

可以证明, 对任意$B \subseteq A$, $0 \leqslant E_r(B) \leqslant \log_2|U|$.

1.2 人工蜂群算法介绍

传统的人工蜂群算法的具体步骤如下.

① 初始化种群. 假设随机产生 N 个初始解, 并且X_i $(1 \leqslant i \leqslant N)$为种群的第 i 个解. X_i 可以用一个 D 维向量来表示, 其中, D 是待优化目标问题的参数个数. 随机获得初始解的过程如下:

$$X_{i,j}=X_{\min,j}+Rand(0,1)\times(X_{\max,j}-X_{\min,j}) \tag{2}$$

其中, $1 \leqslant i \leqslant N$, $1 \leqslant j \leqslant D$; $X_{\min,j}$为搜索解空间的下限; $X_{\max,j}$为搜索解空间的上限.

② 计算适应度. 蜂群将位置替换为适应度最优的位置. 雇佣蜂使用目标函数来评估每个解X_i的适应度, 适应度由式 (3) 计算:

$$fit_i=\begin{cases}\dfrac{1}{1+f_i}, f_i \geqslant 0\\ 1+\mathrm{abs}(f_i), f_i<0\end{cases} \tag{3}$$

其中, f_i是X_i 所对应的目标函数值, 而fit_i则是X_i的适应度.

③ 雇佣蜂 (引领蜂) 搜索. 雇佣蜂对食物源进行领域搜索, 产生候选解, 并计算候选解的适应度值. 具体而言, 雇佣蜂基于贪婪策略来选择食物源, 保留适应度最高的食物源. 根据当前的解X_i, 可以利用式 (4) 来计算新的候选解V_i:

$$V_{i,j}=X_{i,j}+r_{i,j}(X_{i,j}-X_{k,j}) \tag{4}$$

其中, $1 \leqslant i \leqslant N$, $1 \leqslant j \leqslant D$; k 与 j 为随机选取的两个下标, 并且$1 \leqslant k \leqslant N$, $k \neq i$; $r_{i,j}$是 [−1, 1] 区间内的一个随机数.

④ 跟随蜂阶段. 在传统的人工蜂群算法中, 跟随蜂采用轮盘赌的方式 (即按照一定的概率p_i) 来选择已寻找到较优食物源的雇佣蜂进行跟随, 然后根据雇佣蜂搜索公式在其邻域内搜索新的食物源, 并择优保留食物源. 选择概率p_i由式 (5) 来计算:

$$p_i=\frac{fit_i}{\sum_{i=1}^{N/2}fit_i} \tag{5}$$

其中, $1 \leqslant i \leqslant N$, p_i为第 i 个食物源 (解) X_i所对应的概率, fit_i则是X_i的适应度.

由式 (6) 可知, 选择概率p_i与适应度成正比关系. 因此, 在进行特征选择时, 人工蜂群算法容易出现过早收敛、收敛到局部最优等问题. 事实上, 经过多次迭代之后, 当前较差的解也可能包含一些非常有用的信息. 为了避免过早收敛并保证物种多样性, 本文将采用一种改进的基于轮盘赌的选择策略, 使得较差的解也具有较大的选择概率. 在这种改进的选择策略中, 选择概率p_i由式 (6) 来计算:

$$p_i=\frac{1/fit_i}{\sum_{i=1}^{N}(1/fit_i)} \tag{6}$$

⑤ 侦查蜂阶段. 侦查蜂基于概率$prob_i$ $(1 \leqslant i \leqslant N)$来选择一个食物源, 并且更新其位置信息. 每个食物源X_i的选择概率$prob_i$由式 (7) 计算得到:

$$prob_i=\frac{0.9\times fit_i}{max\_fit}+0.1 \tag{7}$$

其中, $1 \leqslant i \leqslant N$, fit_i是X_i的适应度, max_fit为最大适应度.

2 基于云模型的跟随蜂搜索策略

云模型是李德毅院士等所提出的一种表示某个定

性概念$\widetilde{A}$与其定量数值之间不确定性的转换模型[17,18]. 云模型的数字特征有数值期望Ex、熵En和超熵He. 期望Ex是云的重心位置, 是最能表现定性概念的点, 表示在论域空间上分布的期望; 熵En是云的期望曲线宽度, 它的大小反映了在论域中可被模糊概念接受的元素数, 是对定性概念模糊度的度量; 超熵He是云滴的厚度, 反映了云的熵的离散程度, 即熵的熵, 是对熵的不确定度的度量. 云模型综合考虑了随机性和模糊性, 由定量到定性, 用数字特征表示语言值. 再从定性到定量, 通过云发生器来模拟随机性、模糊性以及二者之间的关联性[19]. 假设 U 是用数值集合表示的定量论域, C 是 U 上的定性概念, 若定量值$x \in U$是定性概念 C 的一次随机体现, x对 C 的确定度$\mu(x) \in [0,1]$是具有稳定倾向性的随机数, 即$U \to [0,1]$, $\forall x \in U$, $x \to \mu(x)$, 则称x在论域 U 上的分布为云, 每一个x 称为云滴.

一维正态云算子$C(Ex,En,He)$的计算步骤如下[21].

① 生成以En为期望、He为标准差的正态随机数 $Ei\ (1 \leqslant i \leqslant N)$.

② 生成以Ex为期望、Ei为标准差的正态随机数 $x_i\ (1 \leqslant i \leqslant N)$, x_i对应的是论域空间中的云滴, N 是云滴的总个数.

③ 计算$y_i = \exp(-(x_i - Ex)^2/2Ei^2)$, 令$y_i$为$x_i$属于定性概念$\widetilde{A}$的确定度.

人工蜂群算法的跟随蜂在搜索新的食物源时, 不够稳定且随机性较大, 当搜索次数迭代到一定的数目后, 便由于很难提高蜜源寻优的能力而陷入局部最优, 降低了算法的求解精度. 由于云模型具有稳定倾向性和随机性等优势, 本文运用正态云模型对跟随峰的搜索方式进行改进, 以提高算法的局部开采能力, 从而改善 ABC 算法求解精度低、收敛速度慢的缺点, 提高 ABC 算法的鲁棒性和收敛速度. 我们利用一维正态云模型$C(Ex,En,He)$在初始食物源附近产生新的食物源, 即在跟随蜂阶段, 假设食物源位置为x_{ij}, 令:

$$\begin{cases} Ex = x_{ij} \\ En = -(X_{\max,j} - X_{\min,j})(iter/iter\_max)^2 + X_{\max,j} \\ He = En/10 \end{cases} \tag{8}$$

其中, $X_{\min,j}$为搜索解空间的下限, $X_{\max,j}$为搜索解空间的上限, $iter$ 为当前迭代次数, $iter\_max$ 为最大迭代次数.

3 基于粒度粗糙熵的人工蜂群算法适应度计算

人工蜂群算法的一个关键环节就是适应度的计算, 我们有必要为该算法选择一个合适的适应度函数. 本节将在文献 [18] 中所提出的粗糙熵模型基础上, 进一步提出一种新的信息熵模型——粒度粗糙熵, 并利用粒度粗糙熵来构建人工蜂群算法的适应度函数. 粒度粗糙熵不仅能够刻画出知识的粒度大小, 而且还可以对知识的不完备性进行度量, 因此, 它非常适合于为人工蜂群算法构建适应度函数.

下面, 我们首先给出粒度粗糙熵的具体定义, 并对粒度粗糙熵的基本性质进行分析, 最后将粒度粗糙熵应用于人工蜂群算法的适应度计算中.

定义 3. 粒度粗糙熵. 给定一个信息表$S = (U,A,V,f)$, 对于任意属性子集$B \subseteq A$, 令$U/IND(B) = \{X_1, X_2, \cdots, X_t\}$表示关系$IND(B)$对论域 U 的划分. 属性子集 B 的粒度粗糙熵$GKE_r(B)$被定义为:

$$GKE_r(B) = GK(B) \times E_r(B) \tag{9}$$

其中, $GK(B)$为B的知识粒度, $E_r(B)$为B的粗糙熵.

从定义 3 可以看出, 粒度粗糙熵将知识粒度和粗糙熵有机地结合在一起, 既考虑了知识的不完备性, 又刻画了知识的粒度大小, 从而得到了一种更加全面的不确定性度量机制.

下面, 给出粒度粗糙熵的一些基本性质.

定理 1. 给定一个信息表$S = (U,A,V,f)$, 其中$U = \{x_1, x_2, \cdots, x_n\}$. 对于任意属性子集$B \subseteq A$, B的粒度粗糙熵$GKE_r(B)$满足以下性质.

1) $0 \leqslant GKE_r(B) \leqslant \log_2|U|$.

2) 当$U/IND(B) = \{\{x_1\},\{x_2\},\cdots,\{x_n\}\}$时, $GKE_r(B)$达到最小值.

3) 当$U/IND(B) = \{U\}$时, $GKE_r(B)$达到最大值.

证明:

1) 当$IND(B)$为U上的相等关系时, 即$U/IND(B) = \{\{x_1\},\{x_2\},\cdots,\{x_n\}\}$时, 我们可以得到:

$$\begin{cases} GK(B) = \dfrac{\sum\limits_{i=1}^{t}|X_i|^2}{|U|^2} = \dfrac{\sum\limits_{i=1}^{|U|}1^2}{|U|^2} = \dfrac{|U|}{|U|^2} = \dfrac{1}{|U|} \\ E_r(B) = -\sum\limits_{i=1}^{t}\dfrac{|X_i|}{|U|}\log_2\dfrac{1}{|X_i|} = -\sum\limits_{i=1}^{|U|}\dfrac{1}{|U|}\log_2\dfrac{1}{1} = 0 \end{cases}$$

这时, 我们有: $GKE_r(B) = GK(B) \times E_r(B) = 0$.

在上述情形下, 粒度粗糙熵$GKE_r(B)$取得最小值 0.

2) 当$IND(B)$为U上的论域关系时, 即$U/IND(B)=\{U\}$时, 我们可以得到:

$$\begin{cases} GK(B)=\dfrac{\sum\limits_{i=1}^{t}|X_i|^2}{|U|^2}=\dfrac{|U|^2}{|U|^2}=1 \\ E_r(B)=-\sum\limits_{i=1}^{t}\dfrac{|X_i|}{|U|}\log_2\dfrac{1}{|X_i|}=-\dfrac{|U|}{|U|}\log_2\dfrac{1}{|U|}=\log_2|U| \end{cases}$$

这时, 我们有: $GKE_r(B)=GK(B)\times E_r(B)=\log_2|U|$.

在上述情形下, 粒度粗糙熵$GKE_r(B)$取得最大值$\log_2|U|$.

根据上述证明, 可知$0\leqslant GKE_r(B)\leqslant\log_2|U|$.

由于 2) 和 3) 的证明可以由 1) 的证明直接得到, 因此这里忽略具体过程.

定理 2. 给定一个信息表$S=(U,A,V,f)$, 对于任意两个属性子集$B_1\subseteq A$, $B_2\subseteq A$, 如果$U/IND(B_1)\preceq U/IND(B_2)$, 则$GKE_r(B_1)\leqslant GKE_r(B_2)$.

证明: 假设$U/IND(B_1)=\{X_1,X_2,\cdots,X_m\}$, 并且$U/IND(B_2)=\{Y_1,Y_2,\cdots,Y_n\}$. 由于$U/IND(B_1)\preceq U/IND(B_2)$, 我们可以得到: $m\geqslant n$并且存在一个集合$\{1,2,\cdots,m\}$的划分$C=\{C_1,C_2,\cdots,C_n\}$使得下列公式成立:

$$Y_j=\bigcup_{i\in C_j}X_i,1\leqslant j\leqslant n$$

由定义 2, 我们可以得出:

$$\begin{aligned} E_r(B_2)&=-\sum_{j=1}^{n}\frac{|Y_j|}{|U|}\log_2\frac{1}{|Y_j|} \\ &=-\frac{1}{|U|}\sum_{j=1}^{n}\left|\bigcup_{i\in C_j}X_i\right|\log_2\frac{1}{\left|\bigcup\limits_{i\in C_j}X_i\right|} \\ &=-\frac{1}{|U|}\sum_{j=1}^{n}\left(\sum_{i\in C_j}|X_i|\log_2\left(\frac{1}{\sum\limits_{i\in C_j}|X_i|}\right)\right) \\ &=\frac{1}{|U|}\sum_{j=1}^{n}\left(\sum_{i\in C_j}|X_i|\log_2\left(\sum_{i\in C_j}|X_i|\right)\right) \end{aligned}$$

由于$m\geqslant n$, 并且对任意$1\leqslant i\leqslant m$, $|X_i|\geqslant 1$, 因此, 我们可以得出如下结论: 对于每一个$1\leqslant j\leqslant n$, 都有$\sum\limits_{i\in C_j}|X_i|\log_2\left(\sum\limits_{i\in C_j}|X_i|\right)\geqslant\sum\limits_{i\in C_j}|X_i|\log_2|X_i|$.

根据上述结果, 我们可以进一步得出:

$$\begin{aligned} E_r(B_2)&=\frac{1}{|U|}\sum_{j=1}^{n}\left(\sum_{i\in C_j}|X_i|\log_2\left(\sum_{i\in C_j}|X_i|\right)\right) \\ &\geqslant\frac{1}{|U|}\sum_{j=1}^{n}\left(\sum_{i\in C_j}|X_i|\log_2|X_i|\right) \\ &=\frac{1}{|U|}\sum_{i=1}^{m}|X_i|\log_2|X_i| \\ &=-\sum_{i=1}^{m}\frac{|X_i|}{|U|}\log_2\frac{1}{|X_i|} \\ &=E_r(B_1) \end{aligned}$$

另一方面, 根据文献 [17], 我们可以得出: 如果$U/IND(B_1)\preceq U/IND(B_2)$, 则$GK(B_1)\leqslant GK(B_2)$. 这样, 根据定义 3, 可以得出: $GKE_r(B_1)=GK(B_1)\times E_r(B_1)\leqslant GK(B_2)\times E_r(B_2)=GKE_r(B_2)$.

传统的人工蜂群算法采用式 (4) 计算适应度, 本文则为人工蜂群算法提供一种基于粒度粗糙熵的适应度函数, 具体定义如下:

定义 4. 基于粒度粗糙熵的适应度函数. 假设当前有N个解 (食物源), 对任意$1\leqslant i\leqslant N$, 令X_i表示第i个解, 并且令B_i为X_i所对应的属性子集. 解X_i的适应度由式 (10) 计算:

$$fit_i=\frac{1}{1+GKE_r(B_i)} \tag{10}$$

其中, $1\leqslant i\leqslant N$, $GKE_r(B_i)$是属性子集B_i的粒度粗糙熵, fit_i则是X_i的适应度.

由定理 1 可知, 对任意$1\leqslant i\leqslant N$, fit_i的取值范围为$[(1+\log_2|U|)^{-1},1]$.

4 特征选择算法

本节首先给出一种计算粒度粗糙熵的算法 (即算法 1), 然后给出特征选择算法 FS_GREIABC (即算法 2).

算法 1. 粒度粗糙熵的计算

输入: 给定的信息表$S=(U,A,V,f)$, 任意属性子集$B\subseteq A$.

输出: 属性子集B的粒度粗糙熵$GKE_r(B)$.

1) 初始化: 令$E_{(r)}(B)=0$, 并且令$GK(B)=0$.

2) 通过计数排序的方法[8] 计算划分$U/IND(B)$ (假设$U/IND(B)=\{X_1,X_2,\cdots,X_t\}$).

3) 根据划分$U/IND(B)$计算每个等价类X_i的势$|X_i|$, 其中, $1\leqslant i\leqslant t$.

4) 对任意$1\leqslant i\leqslant t$, 循环执行:

令$E_r(B)=E_r(B)-\frac{|X_i|}{|U|}\log_2\frac{1}{|X_i|}$.

5) 对任意$1\leqslant i\leqslant t$, 循环执行:

令$GK(B)=GK(B)+\frac{|X_i|^2}{|U|^2}$.

6) 计算粒度粗糙熵$GKE_r(B)$, 即令$GKE_r(B)=GK(B)\times E_r(B)$.

7) 返回属性子集B的粒度粗糙熵$GKE_r(B)$.

算法 2. FS_GREIABC

输入: 给定的信息表$S=(U,A,V,f)$.
输出: 最终选择的特征子集.

1) 初始化: 随机产生初始种群, 得到 N 个初始解$X_1,X_2,\cdots,X_N$, 设置蜜源个数 (雇佣蜂个数) *FoodCount*、观察蜂个数 *OnlookerCount*、最大迭代次数 *MaxIterCount* 以及云滴数 *nc*.
2) 对信息表 S 进行离散化处理.
3) 对任意$1\leqslant j\leqslant FoodCount$, 循环执行:
4) 根据算法 1 计算种群中各只蜜蜂的粒度粗糙熵$GKE_r(B_j)$;
5) 根据式 (11) 计算种群中各只蜜蜂的适应度值fit_j.
6) 对任意$1\leqslant t\leqslant MaxIterCount$, 循环执行:
7) 雇佣蜂阶段: 对任意$1\leqslant j\leqslant FoodCount$, 各蜜源根据式 (5) 依次更新.
8) 跟随蜂阶段:
9) 对任意$1\leqslant m\leqslant nc$, 循环执行:
10) 根据式 (9) 计算各食物源一维正态云模型的参数 *Ex*、*En*、*He* 并构造一维正态云模型;
11) 根据式 (7) 计算选择蜜源X_i的概率p_i;
12) 对任意 $1\leqslant k\leqslant OnlookerCount$, 循环执行: 选择较好的蜜源$X_i$进行跟随.
13) 侦查蜂阶段: 对任意$1\leqslant j\leqslant FoodCount$, 循环执行: 选择放弃蜜源并根据式 (8) 产生新蜜源来替代旧蜜源.
14) 获得全局最佳蜜源 *bestfood*.
15) 对最佳蜜源进行 0-1 编码.
16) 得到当前最优的特征子集并返回该特征子集.

5 实验

5.1 实验数据与实验设置

为了验证 FS_GREIABC 算法的性能, 我们首先选取了 10 个 UCI 数据集[22] 进行特征选择实验, 具体包括: 1) Heart; 2) German; 3) Diabetes; 4) Ionosphere; 5) Smart_grid; 6) Sonar; 7) Musk2; 8) Spmbase_disc; 9) WDBC; 10) Dress. 这 10 个数据集的详细信息如表 1 所示.

表 1 10 个 UCI 数据集

数据集	属性个数	样本个数	类别个数
Heart	13	270	2
German	20	1 000	2
Diabetes	16	208	2
Ionosphere	34	351	2
Smart_grid	13	600	2
Sonar	60	208	2
Musk2	168	98	2
Spmbase_disc	57	4 601	2
WDBC	30	569	2
Dress	12	247	2

除上述 10 个 UCI 数据集之外, 为了将 FS_GREIABC 算法与实际应用更好地结合在一起, 我们还考虑将该算法应用于软件缺陷预测领域. 当前软件的规模和复杂度日益增长, 因此, 软件的可靠性备受关注. 在软件工程中, 如果可以找出软件系统中有可能存在缺陷的模块及其分布, 对软件开发者合理配置资源以及提高软件质量将起到至关重要的作用. 软件缺陷预测便是这样一种技术, 它对软件模块中是否存在缺陷进行预测, 能够根据历史数据以及已经发现的缺陷等软件度量数据, 提前预测哪些模块有出错的倾向. 合理地预测软件缺陷可以有效帮助测试者快速定位并弥补软件缺陷, 从而达到显著减少软件开发成本和提高软件可信度的效果.

当前的研究通常将缺陷预测的实现形式化为一个机器学习问题, 很多机器学习技术被用于软件缺陷预测. 然而, 现有的缺陷预测方法在实际应用中还存在许多问题. 例如, 这些方法在高维数据情况下 (如存在大量冗余、不相关的度量元), 预测精度不高, 而在实际应用中高维的缺陷预测数据是非常普遍的[23].

针对软件缺陷预测中所存在的高维数据问题, 本文尝试采用 FS_GREIABC 算法来进行特征选择, 通过对度量元进行有效降维, 从而提高软件缺陷预测模型的性能. 我们选取了 5 个来自于 Jureczko 和 Madeyski[24] 所整理的 Promise 数据集进行特征选择实验, 具体包括: 1) Ant-1.3; 2) Camel-1.0; 3) Jedit-4.3; 4) Synapse-1.0; 5) Log4j-1.0. 这 5 个数据集的详情信息如表 2 所示.

表 2 5 个 Promise 数据集

数据集	样本个数	属性个数	类别个数
Ant-1.3	125	21	2
Camel-1.0	872	21	2
Jedit-4.3	492	21	2
Synapse-1.0	157	21	2
Log4j-1.0	135	21	2

由于粗糙集理论更适合于处理离散型属性, 因此, 对于上述数据集中的连续型属性, 我们统一采用 Weka 平台中所提供的等宽离散化算法进行了离散化处理, 其中, bins=5.

我们基于 Python 语言实现了 FS_GREIABC 算法, 实验所采用的硬件环境具体如下: 1.8 GHz Intel 处理器、8.0 GB 内存. 我们将 FS_GREIABC 算法与以下 5 种具有代表性的基于智能优化技术的特征选择算法进行了对比: 1) 哈里斯鹰算法 (HHO)[25]; 2) 樽海鞘算法 (SSA)[26]; 3) 鲸鱼算法 (WOA)[27]; 4) Jaya 算法[28]; 5) 正余弦算法 (SCA)[29].

为了有效评价 FS_GREIABC 算法及其他对比算法的性能, 本文分别采用了如下 3 种性能评价指标: Accuracy、*F*1-score 和 AUC. 其中, Accuracy 代表分类正确率, 正确率越高, 分类效果越好; *F*1-score 代表由

Accuracy 和 Recall 的调和平均值, 取值范围为 [0, 1], 当 *F*1-score 取得 1 时, 代表模型的输出结果最好, 当 *F*1-score 取得 0 时, 代表模型的输出结果最差; AUC 代表 ROC 曲线下方的面积, 用来判断二分类预测模型的优劣, 其考虑了因数据不均衡而导致的预测偏差问题, 当 AUC 越接近 1 时, 代表模型的分类效果越好, 当 AUC 低于 0.5 时, 代表模型的分类效果较差.

对于每一个实验数据集, 我们通过随机抽样的方式将其划分成训练集和测试集, 其中, 70% 的样本作为训练集, 其余 30% 的样本作为测试集. 我们采用机器学习中常用的一种分类算法——CART 在特征选择之后的训练集上训练分类器, 并利用测试集来测试分类器的分类性能 (即计算该分类器的 Accuracy 值、*F*1-score 值以及 AUC 值). 此外, 我们还比较了不同特征选择算法所选择的特征子集的大小.

在实验过程中, 我们发现, 对于 FS_GREIABC 算法, 不同的参数会达到不同的效果. 通过多次实验, 我们逐步调节 *FoodCount*、*OnlookerCount*、*MaxIterCount*、*nc* 的值, 并选择能够获得最优实验结果的参数值. 最终, 将 *FoodCount* 设置 40、*OnlookerCount* 设置为 40、*MaxIterCount* 设置为 100、*nc* 设置为 500. 对于 5 个对比算法, 它们的参数均采用相关文献中所提供的参数值进行设置.

5.2 实验结果

图 1–图 3 分别表示实验过程中在 Heart、Diabetes 以及 Ionosphere 这 3 个数据集上采用 500 个云滴所构造出的云模型示图.

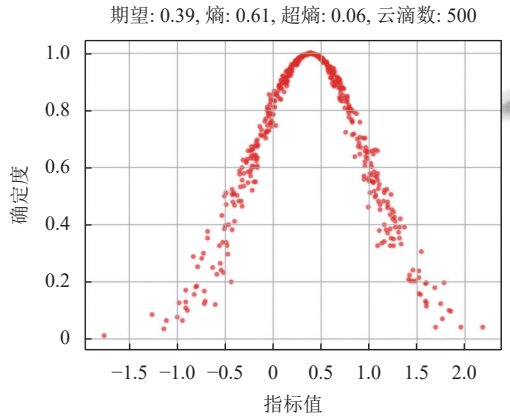

图 1 Heart 数据集上的云模型示意图

从图 1–图 3 可以看出, 熵*En*的取值越大, 则云滴的取值范围越广, 相应地跟随蜂的搜索范围就越广; 反之, 跟随蜂的搜索范围就越窄. 此外, 超熵的取值越大, 则云滴越厚, 相应地食物源的分布就越离散.

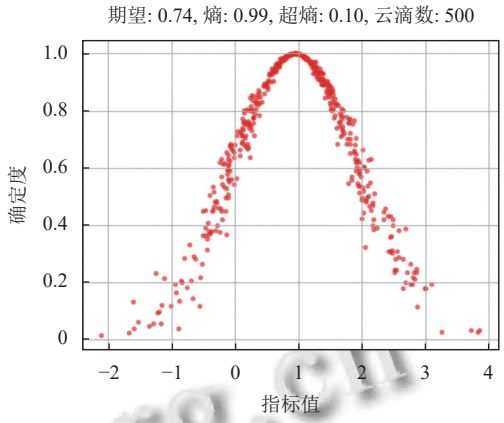

图 2 Diabetes 数据集上的云模型示意图

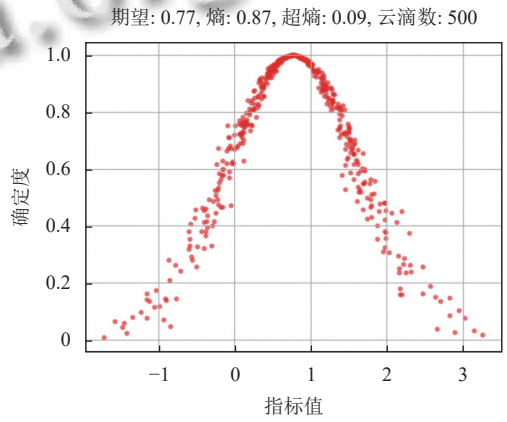

图 3 Ionosphere 数据集上的云模型示意图

表 3 给出了在不同数据集上 6 种特征选择算法所选择的特征子集的大小.

表 3 不同算法所选择的特征子集的大小

数据集	FS_GREIABC	HHO	SSA	WOA	Jaya	SCA
Heart	**5**	9	7	6	7	**5**
German	**5**	10	7	15	11	6
Diabetes	**7**	10	8	10	9	8
Ionosphere	**7**	14	15	15	18	8
Smart_grid	6	**4**	7	11	5	6
Sonar	22	**20**	24	26	31	23
Musk2	**75**	76	77	80	78	76
Spmbase_disc	**17**	27	26	30	34	18
WDBC	**15**	19	19	16	16	16
Dress	**6**	8	7	7	9	7
Ant-1.3	**5**	9	10	15	8	7
Camel-1.0	**7**	11	8	10	12	8
Jedit-4.3	**4**	8	7	9	6	6
Synapse-1.0	**5**	7	7	9	**5**	6
Log4j-1.0	**5**	6	10	6	7	8

从表 3 可以看出 FS_GREIABC 算法所选择的特征数量在大部分数据集上都要少于或等于 5 种对比算法. 除了在 Smart_grid 和 Sonar 这两个数据集上要差于 HHO 之外, FS_GREIABC 在其他数据集上都表现

最好 (即其所选择的特征数量最少). 事实上, 在 Smart_grid 和 Sonar 这两个数据集上, FS_GREIABC 的表现也非常接近于表现最好的 HHO 算法. 因此, 从整体上看, 相对于 HHO、SSA、WOA、Jaya、SCA 这 5 种对比算法, FS_GREIABC 算法能够获得更小的特征子集.

表 4–表 6 分别给出了 6 种特征选择算法在 Accuracy 值、*F*1-score 值以及 AUC 值上的对比结果.

表 4 不同算法的 Accuracy 值对比

数据集	FS_GREIABC	HHO	SSA	WOA	Jaya	SCA
Heart	**0.8636**	**0.8386**	0.85	0.8346	0.7869	0.8259
German	**0.8074**	0.6467	0.6621	0.5073	0.7079	0.5452
Diabetes	**0.9584**	0.9337	0.9352	0.9036	0.9512	0.9186
Ionosphere	**0.9856**	0.9337	0.9478	0.9691	0.9248	0.9687
Smart_grid	0.7424	0.6444	0.6717	0.7222	**0.7488**	0.7378
Sonar	**0.871**	0.7928	0.7918	0.7778	0.8412	0.8163
Musk2	**0.7667**	0.6296	0.4444	0.6296	0.6296	0.7574
Spmbase_disc	**0.9047**	0.9087	0.8928	0.8337	0.8911	0.8942
WDBC	**0.9106**	0.8942	0.9087	0.9087	0.8829	0.903
Dress	0.6571	0.5784	0.5949	0.6238	0.6346	**0.6667**
Ant-1.3	**0.9474**	0.8684	0.8947	0.8421	0.8421	0.8947
Camel-1.0	**0.8969**	0.8435	0.8473	0.8664	0.8664	0.8511
Jedit-4.3	**0.9865**	0.9797	0.973	0.9797	0.9797	0.973
Synapse-1.0	**0.9574**	0.8936	0.9149	0.8936	0.9148	0.9362
Log4j-1.0	**0.9024**	0.8293	0.8293	0.878	0.8537	0.878

表 5 不同算法的 *F*1-score 值对比

数据集	FS_GREIABC	HHO	SSA	WOA	Jaya	SCA
Heart	**0.8765**	0.8024	0.8395	0.8148	0.7901	0.7901
German	**0.73**	0.69	0.7067	0.7033	**0.73**	0.7167
Diabetes	**0.9683**	0.9048	0.9524	0.9206	0.9365	0.9048
Ionosphere	**0.9714**	0.9434	0.934	0.9434	0.915	0.9245
Smart_grid	0.8167	0.7333	0.7333	**0.8333**	0.7667	0.8
Sonar	**0.8888**	0.7302	0.7937	0.8413	0.7143	0.7619
Musk2	**0.9333**	0.90	0.8333	0.90	0.90	0.9
Spmbase_disc	**0.9247**	0.913	0.8986	0.8406	0.913	0.913
WDBC	**0.9825**	0.9275	0.913	0.913	0.9565	0.9565
Dress	**0.68**	0.60	0.6667	0.6533	0.6533	0.6533
Ant-1.3	**0.9714**	0.6849	0.8021	0.7031	0.6545	0.7697
Camel-1.0	**0.9353**	0.5016	0.6533	0.6445	0.7005	0.561
Jedit-4.3	**0.7466**	0.4948	0.4932	0.4949	0.4949	0.4932
Synapse-1.0	**0.8881**	0.4719	0.7267	0.4719	0.6439	0.7685
Log4j-1.0	**0.8576**	0.7403	0.7403	0.796	0.8016	0.8287

从表 4 看出, FS_GREIABC 算法的 Accuracy 值在大部分数据集上都要高于或等于 5 种对比算法. 除了在 Smart_grid 和 Dress 这两个数据集上分别低于 Jaya 和 SCA 算法之外, 在其余 13 个数据集上, FS_GREIABC 算法都具有最高的 Accuracy 值. 事实上, 在 Smart_grid 和 Dress 数据集上, FS_GREIABC 算法的 Accuracy 值也只是略低于 Jaya 和 SCA 算法, 比其余 4 个算法都要高. 因此, 从 Accuracy 这个评价指标来看, FS_GREIABC 算法的性能要优于 5 种对比算法, 即 FS_GREIABC 在提升分类器的分类性能方面表现更好.

从表 5 可看出, FS_GREIABC 算法的 *F*1-score 值在大部分数据集上都要高于 5 种对比算法. 除了在 Smart_grid 数据集上, 在其余 14 个数据集上, FS_GREIABC 算法都具有最高的 *F*1-score 值. 事实上, 在 Smart_grid 数据集上, FS_GREIABC 算法的性能也是非常接近于最优的算法. 因此, 从 *F*1-score 这个评价指标来看, FS_GREIABC 算法的性能要优于 5 种对比算法.

表 6 不同算法的 AUC 值对比

数据集	FS_GREIABC	HHO	SSA	WOA	Jaya	SCA
Heart	**0.8588**	0.8417	0.85	0.8306	0.7861	0.8278
German	0.6548	0.6548	0.6515	0.5412	**0.6952**	0.5619
Diabetes	**0.9773**	0.9337	0.9474	0.9211	0.9605	0.9274
Ionosphere	**0.9973**	0.9365	0.94	0.9663	0.9079	0.9605
Smart_grid	**0.8265**	0.6495	0.6651	0.7105	0.811	0.7822
Sonar	**0.8758**	0.7936	0.7911	0.7814	0.8479	0.8129
Musk2	**0.896**	0.6	0.48	0.7	0.88	0.7
Spmbase_disc	**0.9196**	0.9087	0.8902	0.836	0.8836	0.9074
WDBC	**0.9254**	0.892	0.9042	0.9042	0.8968	0.8985
Dress	**0.6786**	0.5792	0.5985	0.6257	0.6337	0.665
Ant-1.3	**0.8348**	0.651	0.6718	0.7031	0.6354	0.7343
Camel-1.0	**0.6987**	0.5233	0.6283	0.6117	0.6679	0.5559
Jedit-4.3	**0.6667**	0.5	0.4966	0.5	0.5	0.4966
Synapse-1.0	**0.8881**	0.5	0.6881	0.5	0.6	0.7
Log4j-1.0	**0.8339**	0.7177	0.7177	0.75	0.8016	0.8177

从表 6 可以看出, FS_GREIABC 算法的 AUC 值在所有 15 个数据集上都要高于或等于 5 种对比算法. 除了在 German 数据集上, 在其余 14 个数据集上, FS_GREIABC 算法的 AUC 值都要高于其他算法. 因此, 从 AUC 这个评价指标来看, FS_GREIABC 算法的性能要优于 5 种对比算法.

综合表 3–表 6 中的实验结果, 我们可以得出: 本文所提出的基于粒度粗糙熵与改进蜂群算法的特征选择算法 FS_GREIABC 具有更好的特征选择性能.

6 结论

本文利用粗糙集理论和云模型来改进传统的人工蜂群算法, 并由此提出一种新的特征选择算法 FS_GREIABC. FS_GREIABC 通过粒度粗糙熵来计算每个解的适应度. 作为一种新的信息熵模型, 粒度粗糙熵不仅可以对知识的不完备性进行度量, 而且还能够刻画出知识的粒度大小. 此外, 为了提高人工蜂群算法的局部搜索能力, FS_GREIABC 采用一维正态云模型来改进跟随蜂阶段的搜索策略. 实验结果表明, FS_GREIABC 所选择的特征子集不仅更小, 而且基于这些特征子集

所训练的分类器也具有更好的分类性能.

FS_GREIABC 通过粒度粗糙熵来为人工蜂群算法构建适应度函数. 由于粒度粗糙熵是基于经典的粗糙集模型定义的, 该模型更适合于处理离散型属性, 因此, 对于连续型属性, 必须提前进行离散化处理. 然而, 离散化过程必然会带来信息的损失问题. 针对上述问题, 下一步将考虑如何把本文所提出的特征选择方法推广到邻域粗糙集或模糊粗糙集等扩展的粗糙集模型中, 这些模型不需要经过离散化阶段就可以直接处理连续型属性.

参考文献

1 Bennasar M, Hicks Y, Setchi R. Feature selection using joint mutual information maximisation. Expert Systems with Applications, 2015, 42(22): 8520–8532. [doi: 10.1016/j.eswa.2015.07.007]

2 Hancer E, Xue B, Zhang MJ, *et al*. Pareto front feature selection based on artificial bee colony optimization. Information Sciences, 2018, 422: 462–479. [doi: 10.1016/j.ins.2017.09.028]

3 Mashhour EM, El Houby EMF, Wassif KT, *et al*. Feature selection approach based on firefly algorithm and chi-square. International Journal of Electrical and Computer Engineering, 2018, 8(4): 2338–2350.

4 孙林, 赵婧, 徐久成, 等. 基于改进帝王蝶优化算法的特征选择方法. 模式识别与人工智能, 2020, 33(11): 981–994.

5 刘艳, 程璐, 孙林. 基于 K-S 检验和邻域粗糙集的特征选择方法. 河南师范大学学报 (自然科学版), 2019, 47(2): 21–28.

6 Pawlak Z. Rough sets. International Journal of Computer & Information Sciences, 1982, 11(5): 341–356.

7 王锋, 魏巍. 缺失数据数据集的组增量式特征选择. 计算机科学, 2015, 42(7): 285–290.

8 江峰, 王莎莎, 杜军威, 等. 基于近似决策熵的属性约简. 控制与决策, 2015, 30(1): 65–70.

9 董红斌, 滕旭阳, 杨雪. 一种基于关联信息熵度量的特征选择方法. 计算机研究与发展, 2016, 53(8): 1684–1695.

10 王鹏, 王玉红. 基于改进模糊决策粗糙集的最小化决策代价特征选择算法. 计算机应用与软件, 2021, 38(1): 284–292, 296.

11 Tawhid MA, Ibrahim AM. Feature selection based on rough set approach, wrapper approach, and binary whale optimization algorithm. International Journal of Machine Learning and Cybernetics, 2020, 11(3): 573–602. [doi: 10.1007/s13042-019-00996-5]

12 Azar AT, Anter AM, Fouad KM. Intelligent system for feature selection based on rough set and chaotic binary grey wolf optimisation. International Journal of Computer Applications in Technology, 2020, 63(1–2): 4–24.

13 Chen YM, Zhu QX, Xu HR. Finding rough set reducts with fish swarm algorithm. Knowledge-based Systems, 2015, 81: 22–29. [doi: 10.1016/j.knosys.2015.02.002]

14 高薇, 解辉. 基于粗糙集与人工蜂群算法的动态特征选择. 计算机工程与设计, 2019, 40(9): 2697–2703.

15 Karaboga D. An idea based on honey bee swarm for numerical optimization. Kayseri: Erciyes University, 2005. 1899–1902.

16 Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. Journal of Global Optimization, 2007, 39(3): 459–471. [doi: 10.1007/s10898-007-9149-x]

17 李德毅, 刘常昱, 杜鹢, 等. 不确定性人工智能. 软件学报, 2004, 15(11): 1583–1594.

18 张飞舟, 范跃祖, 沈程智, 等. 利用云模型实现智能控制倒立摆. 控制理论与应用, 2000, 17(4): 519–523.

19 苗夺谦, 范世栋. 知识的粒度计算及其应用. 系统工程理论与实践, 2002, 22(1): 48–56.

20 Liang JY, Shi ZZ. The information entropy, rough entropy and knowledge granulation in rough set theory. International Journal of Uncertainty, Fuzziness and Knowledge-based Systems, 2004, 12(1): 37–46. [doi: 10.1142/S0218488504002631]

21 张光卫, 何锐, 刘禹, 等. 基于云模型的进化算法. 计算机学报, 2008, 31(7): 1082–1091. [doi: 10.3321/j.issn:0254-4164.2008.07.003]

22 Dua D, Graff C. UCI machine learning repository. https://archive.ics.uci.edu/ml/index.php. (2021-06-05)[2022-12-29].

23 李瑞. 软件缺陷预测中高维数据处理研究 [硕士学位论文]. 青岛: 青岛科技大学, 2020.

24 Jureczko M, Madeyski L. Towards identifying software project clusters with regard to defect prediction. Proceedings of the 6th International Conference on Predictive Models in Software Engineering. Timişoara: ACM, 2010. 9.

25 Heidari AA, Mirjalili S, Faris H, *et al*. Harris hawks optimization: Algorithm and applications. Future Generation Computer Systems, 2019, 97: 849–872. [doi: 10.1016/j.future.2019.02.028]

26 Mirjalili S, Gandomi AH, Mirjalili SZ, *et al*. Salp swarm algorithm: A bio-inspired optimizer for engineering design problems. Advances in Engineering Software, 2017, 114: 163–191. [doi: 10.1016/j.advengsoft.2017.07.002]

27 Mirjalili S, Lewis A. The whale optimization algorithm. Advances in Engineering Software, 2016, 95: 51–67. [doi: 10.1016/j.advengsoft.2016.01.008]

28 Rao RV. Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. International Journal of Industrial Engineering Computations, 2016, 7(1): 19–34.

29 Mirjalili S. SCA: A sine cosine algorithm for solving optimization problems. Knowledge-based Systems, 2016, 96: 120–133. [doi: 10.1016/j.knosys.2015.12.022]

(校对责编: 孙君艳)