软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, 2022,33(11):3903−3916 [doi: 10.13328/j.cnki.jos.006327]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

用于特征选择的乌鸦搜索算法的研究与改进*

廉 杰[1,2], 姚 鑫[1,2], 李占山[1,2]

[1](吉林大学 计算机科学与技术学院, 吉林 长春 130012)

[2](符号计算与知识工程教育部重点实验室(吉林大学), 吉林 长春 130012)

通信作者: 李占山, E-mail: lizs@jlu.edu.cn

摘 要: 特征选择是机器学习领域的热点问题. 元启发式算法作为特征选择的重要方法之一, 其性能会对问题求解产生直接影响. 乌鸦搜索算法(CSA)是受乌鸦智能群体行为启发提出的一种元启发式算法, 由于其具有简单、高效的特点, 广大学者将其用来解决特征选择问题. 然而, CSA 易陷入局部最优解且收敛速度较慢, 严重限制了算法求解能力. 针对这一问题, 采用 logistic 混沌映射、反向学习方法和差分进化这 3 种算子, 结合乌鸦搜索算法, 提出一种特征选择算法 BICSA 来选取最优特征子集. 实验阶段, 使用 UCI 数据库中的 16 个数据集来测试 BICSA 的性能. 实验结果表明, 与其他特征选择算法相比, BICSA 求得的特征子集具有更高的分类准确率和较高的维度压缩能力, 这说明 BICSA 在处理特征选择问题上具有很强的竞争力与足够的优越性.

关键词: 乌鸦搜索算法; 混沌映射; 反向学习; 差分进化; 特征选择

中图法分类号: TP301

中文引用格式: 廉杰, 姚鑫, 李占山. 用于特征选择的乌鸦搜索算法的研究与改进. 软件学报, 2022, 33(11): 3903−3916. http://www.jos.org.cn/1000-9825/6327.htm

英文引用格式: Lian J, Yao X, Li ZS. Research and Improvements on Crow Search Algorithm for Feature Selection. Ruan Jian Xue Bao/Journal of Software, 2022, 33(11): 3903−3916 (in Chinese). http://www.jos.org.cn/1000-9825/6327.htm

Research and Improvements on Crow Search Algorithm for Feature Selection

LIAN Jie[1,2], YAO Xin[1,2], LI Zhan-Shan[1,2]

[1](College of Computer Science and Technology, Jilin University, Changchun 130012, China)

[2](Key Laboratory of Symbolic Computation and Knowledge Engineering (Jilin University), Ministry of Education, Changchun 130012, China)

Abstract: Feature selection is a hot issue in the field of machine learning. Meta-heuristic algorithm is one of the important methods of feature selection, and its performance will have a direct impact on problem solving. Crow search algorithm (CSA) is a kind of meta-heuristic algorithm inspired by the behavior of crow intelligent group. Because of its simple and efficient characteristics, it is used by many scholars to solve the feature selection problem. However, CSA is easy to fall into a local optimal solution and the convergence speed is slow, which severely limits the algorithm's solving ability. In response to this problem, this study uses three operators, namely, logistic chaotic mapping, opposition-based learning method, and differential evolution, combined with crow search algorithm, proposes a feature selection algorithm BICSA to select the optimal feature subset. In the experimental phase, the performance of BICSA was demonstrated by using 16 data sets in the UCI database. Experimental results show that compared with other feature selection algorithms, the feature subset obtained by BICSA has higher classification accuracy and higher dimensional compression capabilities, indicating that BICSA has the ability to deal with feature selection problems with strong competitiveness and sufficient superiority.

Key words: crow search algorithm; chaotic map; opposition-based learning; differential evolution; feature selection

* 基金项目: 国家自然科学基金(61802056); 吉林省自然科学基金(20180101043JC); 吉林省发展和改革委员会产业技术研究与开发项目(2019C053-9)

收稿时间: 2020-12-09; 修改时间: 2021-01-11; 采用时间: 2021-02-08

在机器学习和数据挖掘等领域中, 获得可用数据集往往是算法的第 1 步. 当数据集维度变高且存在大量冗余无用特征时, 就会使得数据分析变得极其困难, 后续的算法性能也会受到极大影响[1]. 所以, 进行数据预处理是十分必要的. 特征选择是数据预处理中一个重要的方法, 该方法在大量特征中选取少量数量的相关和非冗余特征, 并且使处理后的数据集在最大程度上不影响分类准确率, 以此来达到降低维度、提升后续分类算法准确性的目的[2].

当搜索空间变得很大时, 特征选择就可视为一种最优化问题. 事实上, 在具有 N 个特征的数据集中, 选择最优特征子集的方案就存在 2^N 种, 所以当特征数 N 过大时, 穷举搜索的方法是不可行的[3]. 为了解决这个问题, 人们引入了元启发式算法来解决特征选择问题, 取得了良好的效果. 乌鸦搜索算法(crow search algorithm, CSA)[4]是近年来提出的一种受乌鸦群体智能行为启发的元启发式算法, 因为它简单和易实现的特质, 引起了广大学者的关注. 但跟其他元启发式算法相同, 具有收敛速度慢、容易陷入局部最优的缺点, 严重影响了算法的分类准确率以及收敛速度. 所以本文对 CSA 进行了改进, 提出一种新的特征选择算法 BICSA(binary improved crow search algorithm), 且通过实验来展示 BICSA 在解决特征选择问题上的卓越性能.

CSA 可以分为初始化和更新两个阶段, 本文针对这两个阶段应用了 3 种算子来改进算法, 提升算法的准确率, 加快收敛速度.

- 在初始阶段, 使用混沌方法(chaotic map)[5]代替了原始算法中的随机生成种群. 之所以选择这种方法, 是因为混沌映射具有随机性、遍历性和动态行为等特性, 这些特性可以生成一个具有良好分布的可行解空间, 从而避免随机生成种群的缺点. 随后, 使用反向学习方法(opposition-based learning, OBL)[6]来计算混沌种群的相反种群. 一些实验表明, 相反的候选解决方案比随机解能够到达全局最优的概率更高[7]. 所以, OBL 的添加使得初始种群可以从多个方向来接近最优解, 加大了算法找到最优解的可能性. 最后, 根据适应度函数值, 在混沌种群和相反种群中选取适应度值高的解. 这两个方法的添加, 使得算法能够在初始化阶段得到质量更好, 更有可能到达最优解的初始种群, 从而加快了算法的收敛速度.
- 在更新阶段引入了差分进化算法(differential evolution, DE)[8], 通过使用 CSA 与 DE 的混合算法来提升算法性能. 首先, 使用 CSA 对特征空间进行搜索; 然后, 用 DE 对生成的解进行变异、交叉、选择来得到更加优秀的全局解. DE 算子的添加, 增加了种群的多样性, 增强了算法的全局搜索能力, 在某种程度上降低了算法陷入局部最优解的概率, 使得算法更加容易地找到最优解.

这 3 种算子使得改进后的算法具备更快的收敛速度, 且更有可能获取全局最优解.

通过以上改进, 形成 ICSA (improved crow search algorithm)算法. 但 ICSA 只适合解决具有连续值的问题, 而特征选择是一个二元问题, 所以需要对 ICSA 进行二元化, 从而提出了特征选择算法 BICSA. 为了展示 BICSA 特征选择算法的性能, 在 UCI 数据库中选择了 16 个数据集进行实验, 并将实验结果与近年来提出的 7 种特征选择算法进行比较与分析. 比较结果证明了改进算法 BICSA 拥有着更高的分类准确率和较高的维度压缩能力.

本文的 3 个主要贡献如下:

(1) 利用混沌映射、反向学习方法和差分进化算法这 3 种算子对 CSA 进行改进, 形成 ICSA 算法. 在初始化阶段, 结合混沌映射和 OBL 这两种方法提出一种新的初始化策略, 并在更新阶段使用了 CSA 与 DE 的混合算法进行种群的更新. 这些改进加快了算法的收敛速度, 并且增加了算法获取全局最优解的概率.

(2) 对 ICSA 算法进行二元化, 提出一种新的特征选择算法: BICSA.

(3) 实验方面, 使用了 UCI 数据库中的 16 个数据集来评估 BICSA 的性能, 并将实验结果与近年来提出的 7 个特征选择算法进行比较. 比较结果证明了 BICSA 在特征选择问题上具有很强的竞争力.

本文第 1 节介绍相关工作. 第 2 节是先导知识, 包括乌鸦搜索算法、混沌映射、反向学习方法和差分进化算法. 第 3 节提出改进的乌鸦特征选择算法 BICSA. 第 4 节给出了实验结果与分析. 第 5 节是总结与展望.

1 相关工作

特征可以看作是数据集的一种属性, 近些年来数据集中的特征数量呈指数级增长, 对机器学习和数据挖掘算法带来了极大的困难, 所以人们提出特征选择方法来对数据进行预处理, 减轻维数灾难, 提升后续分类算法的分类准确率[9]. 特征选择是从所有特征中选取最优特征子集, 去除无用、噪声特征的一种过程, 主要分为两个步骤: 搜索策略和子集质量评估[10]. 搜索策略利用算法来选取特征子集, 然后通过分类器对选取的子集进行训练测试来评估子集的分类准确率等性能指标(即特征子集的质量), 这一过程被称作子集质量评估. 算法会进行一定次数的迭代, 在这个迭代过程中保留下来的质量最好的特征子集, 就是特征选择算法所得到的最优解.

在文献中, 特征选择有 3 种方法: 过滤式[3]、包裹式[11]和嵌入式[12]. 过滤式方法是早期经常使用的一种方法, 它通过数据集本身的一些特性去选取特征子集, 删除无用特征, 然后送进分类器中学习. 也就是说, 选取特征的过程与分类器是无关的. 这种方法速度较快、开销小, 但是最终算法得到的结果一般. 包裹式方法是先通过分类器的性能来得到适应度函数, 然后把它当作选取特征子集的评价指标, 再利用分类器去反复学习, 最终选取最优特征子集. 这种方法效果较好, 但缺点是开销较大, 比较耗费时间. 嵌入式方法的选取过程主要是将分类器学习和特征选择集成到一个过程中, 以选取相关特征子集[12].

元启发式算法是一种通过模仿自然界的生物和物理现象形成的算法[13]. 近年来, 由于元启发式算法在解决最优化等复杂问题上表现出的强大的效率和表现, 使得许多学者们选择元启发式算法来作为一种包裹式方法去解决特征选择问题. 比如: Aziz 等人[14]通过 rough set 理论改进布谷鸟搜索算法, 提出了 MCSRS 特征选择算法; Hou 等人[15]将果蝇算法作为一种包裹式方法来解决特征选择问题, 并且根据动态种群进化方法结合变异策略, 提出了 4 种算法变体, 提升了算法性能; Kashef 等人[16]提出了一种新的基于蚁群算法的特征选择算法, 算法将特征表示为图的节点, 通过蚂蚁在节点上的移动来选择最佳特征子集; Mafarja 等人[17]介绍了一种二元化的鲸鱼优化算法来解决特征选择问题; Al-Tashi 等人[18]将灰狼优化算法和粒子群算法的混合算法二元化, 提出了 BGWOPSO 算法, 并选取了 UCI 数据库中的 18 个数据集进行了测试, 证明了算法性能.

本文选择的 3 种改进方法已经在特征选择领域上证实了其可以提升算法的效率, 比如: Aziz 等人[19]提出了一种混合差分进化正余弦特征选择算法, 并在 8 个数据集上进行了实验, 结果证明这种混合算法很大程度上改进了正余弦算法容易陷入局部最优的缺点, 提升了分类准确率; Sayed 等人[20]选择了 10 种混沌映射方法来提升 CSA 的性能, 提出了 CCSA 特征选择算法; Ibrahim 等人[21]利用 OBL 方法产生初始种群来改进群居蜘蛛优化算法, 提出了 OBSSA 特征选择算法. 所以, 本文结合这 3 种方法对乌鸦搜索算法做出了改进, 提出了特征选择算法 BICSA, 以此提升算法的分类性能.

2 先导知识

2.1 乌鸦搜索算法

乌鸦搜索算法[4]是 2016 年提出的一种基于乌鸦种群的元启发式算法. 乌鸦会隐藏食物, 并且会跟踪其他乌鸦以便偷取食物, 而被跟踪的乌鸦会有一定的概率发现跟踪者. 根据乌鸦的这些智能行为产生了乌鸦搜索算法. 在搜索空间中, 每只乌鸦 i 在第 gen 次迭代的位置可以表达为一个向量如公式(1). 它代表了当前空间中的一个解.

$$x_i^{gen}=[x_{i,1}^{gen},x_{i,2}^{gen},...,x_{i,n}^{gen}],\ i=1,2,...,NP,\ gen=1,...,max\_gen \tag{1}$$

其中, NP 代表了种群大小, max_gen 代表了最大的迭代次数, n 代表数据维度.

每只乌鸦在每次迭代中都会保存自己隐藏食物的位置, 用 m 来表示, 这个位置代表了乌鸦当前所拥有的最佳位置. 算法进行迭代时, 乌鸦会在搜索空间中通过跟踪其他乌鸦来得到更加优秀的位置. 假如在第 gen 次迭代时, 乌鸦 j 准备去查看自己的食物, 此时乌鸦 i 决定跟踪它以获取它隐藏食物的位置. 在这种情况下, 会有以下两种可能性发生[4].

- 情况 1: 乌鸦 j 没有发现乌鸦 i 在跟踪它. 那么, 乌鸦 i 就会一直跟随乌鸦 j 直到到达它隐藏食物的位置, 由公式(2)可以得到乌鸦 i 的新位置.

$$x_i^{gen+1}=x_i^{gen}+r_i\times FL_i^{gen}(m_j^{gen}-x_i^{gen}) \tag{2}$$

其中, r_i 为[0,1]之间均匀分布的随机数, FL_i^{gen} 是乌鸦 i 在第 gen 次迭代时的飞行长度.

- 情况 2: 乌鸦 j 发现了乌鸦 i 在跟踪它. 此时, 乌鸦 j 会随机飞到一个位置, 以保护自己的食物.

综合以上两种情况, 乌鸦 i 在跟踪乌鸦 j 时得到的新位置如公式(3)所示.

$$x_i^{gen+1}=\begin{cases}x_i^{gen}+r_i\times FL_i^{gen}(m_j^{gen}-x_i^{gen}), & \text{if } a_j\geqslant AP_j^{gen}\\ \text{a random position}, & \text{otherwise}\end{cases} \tag{3}$$

其中, a_j 和 r_i 为[0,1]之间均匀分布的随机数, AP 是被跟随乌鸦发现其他乌鸦的跟踪概率.

算法初始会随机生成每只乌鸦的初始位置, 并将初始位置设定为它们的初始记忆, 然后在每次迭代中, 每只乌鸦会随机挑选一只乌鸦进行跟踪, 且根据公式(3)来更新乌鸦的位置, 在位置改变后, 利用适应度函数 fit 来评估乌鸦的新位置: 如果新位置更加优秀, 那么就将乌鸦隐藏食物的位置改变为新位置, 即改变乌鸦的记忆; 反之则不变. 如公式(4)所示.

$$m_i^{gen+1}=\begin{cases}x_i^{gen+1}, & \text{if } fit(x_i^{gen}) \text{ is better than } fit(m_i^{gen})\\ m_i^{gen}, & \text{otherwise}\end{cases} \tag{4}$$

CSA 的伪代码如算法 1 所示.

算法 1. Crow Search Algorithm.

Set the initial values of m, AP, FL, and gen_max. Initialize randomly positions of NP crows.

Initialize the memory of NP crows with the initialize positions.

Evaluate the position of each crow.

While $gen<gen\_max$

  For i=1:NP

    Update every crow's position using Eq.(3).

  End for

  Check new positions according to the constrains of the problems.

  Evaluate the new position of each crow

  Using Eq.(4) to update the memory of crows.

  $gen=gen+1$.

End while

Return the best solution;

2.2 混沌映射

混沌系统[5]是数学研究的一个领域, 它也被应用在物理、工程和经济学等多个方面. 近些年来, 越来越多的学者将混沌系统和元启发式算法进行结合, 以提升算法性能. 混沌系统之所以能对元启发式算法性能进行改进, 是因为混沌系统具有随机性、周期性、遍历性和对初始条件敏感等特性[22], 这些特性使得算法的搜索能力和收敛速度提高, 从而提升算法性能.

研究表明, 混沌系统在算法中代替初始种群、交叉算子和变异算子这 3 部分中的随机数值, 可以提升算法性能[22]. 本文选择将混沌系统应用在初始种群部分. 对于元启发式算法来说, 为了尽可能地确保初始种群的个体均匀地分布在搜索空间中, 初始种群的多样性和遍历性是十分重要的. 所以, 利用混沌系统的遍历性和随机性的特点, 可以生成一个具有良好分布的可行解空间. 用混沌系统来代替 CSA 中的随机生成初始种群, 保证了种群的多样性, 为元启发式算法提供了一种更有效的搜索方案, 加快了算法的收敛速度, 提升了算法的全局搜索能力. 在混沌系统中, 用混沌映射来生成混沌序列. 本文采用经典的 logistic 映射[23]来生成初始解

序列.

2.3 OBL方法

反向学习(OBL)[6]方法是一种基于相反概念的机器智能算法, 当我们在种群中随机生成一个解时, 往往会重新访问搜索空间中没有希望的区域, 而 OBL 的思想就是要同时考虑候选方案及其相反的解决方案, 然后通过适应度函数比较结果来选取更加优秀的解. 实验表明, 如果没有先验知识来优化问题, 相反的候选解往往比随机解能够到达全局最优的概率更高[24]. 因此, 引入一个随机解及其对应的相反解, 比引入两个独立的随机解更有希望到达全局最优解.

在生成初始种群时, 由于没有算法的反馈信息, 往往是随机生成初始种群, 然后在算法迭代过程中, 利用适应度函数等结果来引导解决方案向着最优方案的方向前进, 但如果在最坏的情况下, 即生成解是最优解的相反解时, 那么在算法搜索和优化过程中将会消耗很长的时间, 有时问题还会变得非常棘手, 使得算法难以找到最优解. 所以, 本文在生成初始种群时加入 OBL 方法, 使得在迭代初期可以向多个方向进行搜索, 从而加快收敛速度, 增加了寻求最优解的机会.

Tizhoosh[6]在 2005 年提出了 OBL 方法的基础定义, 考虑一个实数 $x\in[l,u]$, u 和 l 分别是 x 的上界和下界, 那么 x 的相反数如公式(5)所示.

$$\overline{x}=u+l-x \tag{5}$$

最后, 在优化算法中, 如果现有解的适应度函数值小于相反解的适应度函数值, 那么就选择当前解; 否则就选择相反解. 通过计算和评估现有解和相反解来选择更加优秀的解, 从而提升了算法找到最优解的概率.

2.4 差分进化算法

差分进化算法(DE)[8]是一种求解优化问题的基于种群的元启发式算法. 和其他元启发式算法相似, DE 先在解空间中随机生成初始种群, 种群个体在每次迭代中进行变异、交叉、选择等操作, 直到迭代终止或者得到最优解.

对于最原始的 DE 来说, 交叉、变异、选择操作定义如下.

(1) 变异. 经典的 DE 算法的变异算子如公式(6)所示.

$$v_i^t=x_{r1}^t+F\times(x_{r2}^t-x_{r3}^t) \tag{6}$$

其中, v 代表变异得到的解, x 代表种群中的解, $r1$–$r3$ 是 1–NP 中的 3 个互不相等的随机整数, t 是迭代次数, F 是变异概率.

(2) 交叉. 在变异操作后, 将变异得到的解与原有解进行交叉, 以提高种群的潜在多样性. 具体如公式(7)所示.

$$z_{i,j}^t=\begin{cases}v_{i,j}^t, \text{ if } rand\leqslant CR\\ x_{i,j}^t, \text{ otherwise}\end{cases} \tag{7}$$

其中, $rand$ 是 0–1 的随机数, i 代表乌鸦, j 代表数据维度, CR 为交叉概率.

(3) 选择. DE 按照贪心思想, 根据适应度函数值来选择下一代的个体, 如公式(8)所示.

$$x_i^{t+1}=\begin{cases}z_i^t, \text{ if } f(z_i^t)\leqslant f(x_i^t)\\ x_i^t, \text{ otherwise}\end{cases} \tag{8}$$

其中, f(z)是种群个体解 z 的适应度函数值.

3 BICSA 算法

与其他元启发式算法类似, CSA 算法收敛速度慢, 容易陷入局部最优. 针对这些缺点, 本文从初始化和更新两个阶段对 CSA 进行改进形成 ICSA 算法; 并且针对特征选择问题, 将改进算法 ICSA 二元化, 提出了 BICSA 特征选择算法.

3.1 初始化阶段

在初始化阶段, 本文使用了混沌映射和 OBL 方法来生成初始种群. 初始种群是元启发式算法中影响算法的一个重要因素, 因此本文使用了混沌映射的方式代替了 CSA 中的随机方式来生成种群, 公式如(9)所示.

$$x_{ij}=1_{ij}+ch_{ij}\times(u_{ij}-l_{ij}),\ x_i\in X,\ i=1,2,\ldots,N,\ j=1,\ldots,dim \tag{9}$$

其中, l 和 u 代表了下界和上界, ch_{ij} 是根据混沌 logistic 映射[24]产生的混沌参数, 如公式(10)所示.

$$G^{t+1}=\mu\times G^t(1-G^t),\ t=1,2,\ldots,t_{\max} \tag{10}$$

其中, μ为控制参数, $\mu\in(2,4]$, t 代表迭代次数.

经过混沌映射后, 再利用 OBL 方法来产生相反种群. OBL 方法在多维情况下的相反向量[6]定义如公式(11)所示.

$$\bar{x}_j=u_j+l_j-x_j,\ j=1,2,...,dim \tag{11}$$

其中, l_j 和 u_j 是 x_j 的下界和上界, dim 是数据集的维度

得到相反种群后, 对混沌种群和相反种群中每一个解计算适应度函数值并进行有序排列, 最后利用贪心思想选择前 NP 个优秀解作为初始种群, 以此达到了提升初始种群质量的目的.

3.2 更新阶段

在更新阶段, 使用了 CSA 与 DE 的混合算法进行求解. 原始 CSA 的更新阶段具有一定的缺陷, 比如容易陷入局部最优解等, 所以我们在更新阶段中加入了 DE 算法来进行改进. 算法先通过 CSA 的全局搜索和局部搜索来对解进行更新, 然后用 DE 算法对得到的种群进行变异、交叉, 最后选择解空间中最优秀的解. DE 的添加, 使得种群有了进一步更新的机会, 增加了种群的多样性, 进而增强了算法的搜索能力, 减少算法陷入局部最优解的概率.

3.3 BICSA

为解决特征选择问题, 对改进算法 ICSA 进行了二元化, 提出了一种新型的乌鸦特征选择算法 BICSA. 在 BICSA 中, 解空间的所有解都是在[0,1]范围内的. 解从连续值转换到二进制形式的转换函数如公式(12)所示.

$$y^{j,t+1}=\begin{cases}1, & \text{if } (s(y^{j,t+1}))\geqslant rand\\ 0, & \text{otherwise}\end{cases} \tag{12}$$

其中, $y^{j,t+1}$ 是在第 t 次迭代在第 j 维的连续值的输入, $rand$ 为[0,1]之间均匀分布的随机数, s 型函数如公式(13)所示.

$$s(y^{j,t+1})=\frac{1}{1+\mathrm{e}^{-y^{j,t+1}}} \tag{13}$$

BICSA 算法所选择的特征子集都会由 KNN 分类器进行评估, 因为特征选择问题的目标是想找到有着最大分类准确率的最小特征子集, 所以我们的适应度函数设置如公式(14)所示.

$$fit=\alpha\times Err+(1-\alpha)\times\frac{|R|}{|C|} \tag{14}$$

其中, Err 是分类错误率, R 是选取的特征数, C 是数据集中的整体特征数, α对应分类错误率在整个公式所占的比例权重, α的取值在[0,1]范围内.

BICSA 算法的参数设定由表 1 给出, BICSA 的伪代码如算法 2 所示.

表 1 改进算法的参数值

参数	*AP*	*FL*	*F*	*CR*	*u*	Lower bounds	Upper bounds
值	0.1	2	0.9	0.5	4	0	1

算法 2. BICSA.

Set the initial values of *m*, *AP*, *FL*, and *gen_max*.

Use Eq.(11) to generate chaotic population.

Use Eq.(7) to find the opposite solution of chaotic population.
Evaluate the position of two populations
Choose the best *NP* solutions as the initial population.
Initialize the memory of *NP* crows with the initialize positions.
While *gen*<*gen_max*
  For *i*=1:*NP*
    Randomly select one crow to follow (for example *j*).
    Update crow's position using Eq.(3).
  End for
  Mutate each solution in the population according to the Eq.(8).
  According to the Eq.(9), cross the variation solution with the solution in the population.
  Choose a better next-generation solution according to the Eq.(10).
  Check new positions according to the constrains of the problems.
  Using Eq.(4) to update the memory of crows.
  gen=*gen*+1.
End while
Return the best solution;

4 实验部分与讨论

在实验中采用了 MATLAB 作为编程语言, 所有实验都是在带有 Inter Core i7 CPU(3.20GHz), 8G 内存的 DELL 机上执行, 所有的算法都使用了 MATLAB R2020a 进行测试.

4.1 数据集

实验中选取 16 个 UCI 数据库中的数据集来验证 BICSA 的性能, 表 2 描述了数据集的详细信息. 每个数据集都使用 10 折方法来生成训练集和测试集并进行交叉验证. 此外, 实验利用 KNN 分类器对得到的每个特征子集进行评价.

表 2 数据集详细信息

No.	Dataset	Features	Samples
1	Breastcancer	9	699
2	BreastEW	30	569
3	Clean1	166	476
4	Clean2	166	6 598
5	CongressEW	16	435
6	HeartEW	13	270
7	IonosphereEW	34	351
8	KrvskpEW	36	3 196
9	Lymphography	18	148
10	PenglungEW	325	73
11	SonarEW	60	208
12	SpectEW	22	267
13	Tic-tac-toe	9	958
14	Vote	16	300
15	WaveformEW	40	5 000
16	WineEW	13	178

4.2 评价指标和共同参数设置

为了直观地说明 BICSA 算法相较于其他特征选择算法的优势, 实验将使用以下指标进行衡量.

(1) 最优适应度. 算法在 *M* 次运行后的最优秀的适应度值.

(2) 平均准确率. 算法在 *M* 次运行后的准确率的平均值, 如公式(15)所示.

$$AvgAcc=\frac{1}{M}\sum_{i=1}^{M}Acc^{i} \tag{15}$$

其中, Acc^{i} 是第 i 次运行后获得的最优特征子集的准确率值.

(3) 平均适应度. 算法在 M 次运行后的适应度的平均值, 如公式(16)所示.

$$AvgFit=\frac{1}{M}\sum_{i=1}^{M}fit_{Best}^{i} \tag{16}$$

其中, fit_{Best}^{i} 是第 i 次运行后获得的最优特征子集的适应度值.

(4) 平均特征选择率. 算法运行 M 次后, 选择的特征数与原始特征数的比值的平均值, 公式如(17)所示.

$$AvgFR=\frac{1}{M}\sum_{i=1}^{M}\frac{FSN^{i}}{D} \tag{17}$$

其中, FSN^{i} 是第 i 次运行后为获得最佳方案而选择的特征数量, D 是数据集的维度.

在所有实验中, 最大迭代次数设置为 200, 种群大小设置为 50. 此外, 实验结果是通过 20 次(M=20)独立运行所得到的最终统计测量数据, 解的维数等于每个数据集的特征数. 在适应度函数中, 决定了分类准确率的权重α设置为 0.99.

4.3 实验结果和分析

本节将 BICSA 与其他 7 个近年来提出的特征选择算法进行比较, 分别为 CSA, SCA, CASO, GWOPSO, BMNABC, CCSA, WOA. 比较算法的详细信息在表 3 中列出.

表 3 对比算法

算法名称	描述/发表年份	参数	值
CSA	乌鸦搜索算法[4]/2016	AP/FL	0.1/2
SCA	正余弦优化算法[25]/2016	α	2
WOA	鲸鱼优化算法[17]/2017	r	[0,0.9]
BMNABC	二元多邻域人工蜂群优化算法[26]/2018	$r_{\max}/r_{\min}$	1/0
GWOPSO	混合灰狼优化算法[18]/2019	$c1, c2, c3$	0.5
CCSA	混沌乌鸦搜索算法[20]/2019	AP/FL	0.1/2
CASO	混沌原子搜索优化算法[27]/2020	α/β	50/0.2

表 4 中列出了所有算法对于每个数据集的最优适应度值.

表 4 BICSA 及其比较算法的最优适应度值

Dataset	CSA	SCA	CASO	GWOPSO	BMNABC	CCSA	WOA	BICSA
Breastcancer	0.026 5	0.028 8	0.029 3	0.029 0	0.027 9	0.026 5	0.027 9	**0.025 1**
BreastEW	0.047 5	0.045 6	0.044 5	0.048 7	0.044 5	0.049 2	0.044 8	**0.042 4**
Clean1	0.080 8	0.084 7	0.071 4	0.076 1	0.069 6	0.085 2	0.099 3	**0.056 6**
Clean2	0.031 8	0.034 2	0.025 8	0.026 7	0.024 1	0.032 0	0.030 2	**0.020 2**
CongressEW	0.038 0	0.041 3	0.035 1	0.032 2	0.035 1	0.036 3	0.035 7	**0.031 7**
HeartEW	0.134 5	0.147 6	0.137 4	0.130 8	0.140 3	0.136 6	0.130 8	**0.130 1**
IonosphereEW	0.117 5	0.062 9	0.061 3	0.083 9	0.062 9	0.108 5	0.079 9	**0.060 7**
KrvskpEW	0.048 8	0.055 7	0.031 5	0.029 7	0.031 6	0.056 7	0.054 4	**0.027 5**
Lymphography	0.113 1	0.130 4	0.105 3	0.092 0	0.105 3	0.113 1	0.124 3	**0.091 6**
PenglungEW	0.087 2	0.068 7	0.058 8	0.073 0	**0.028 9**	0.074 0	0.056 7	0.030 0
SonarEW	0.116 1	0.111 1	0.085 9	0.091 0	0.097 2	0.119 9	0.128 1	**0.065 9**
SpectEW	0.160 3	0.146 9	0.149 6	0.154 2	0.143 6	0.153 8	0.157 5	**0.138 0**
Tic-tac-toe	**0.161 8**	0.173 1	0.162 8	0.162 8	0.161 9	0.164 9	0.163 9	0.163 8
Vote	0.046 5	**0.035 5**	0.038 8	0.043 4	0.041 5	0.045 2	0.046 0	0.038 2
WaveformEW	0.160 9	0.174 2	0.154 1	0.158 4	0.155 2	0.163 8	0.170 4	**0.153 8**
WineEW	0.038 8	0.032 1	0.037 2	0.038 8	0.032 4	0.040 3	0.041 8	**0.031 7**

观察表 4 可以看出, 在 16 个数据集中, BICSA 取得的最优适应度值在 13 个数据集上取得领先位置; 仅在 PenglungEW 和 Vote 数据集上分别输给 BMNABC 和 SCA, 居于第二位; 在 Tic-tac-toe 数据集上表现不佳, 排在第五位.

表 5 列出了所有算法的平均适应度值, 观察表 5 可得: BICSA 在 16 个数据集中的 14 个上的平均适应度值

超过了其他所有的算法结果, 均居于首位, 仅有 IonosphereEW 和 PenglungEW 数据集的结果排在 BMNABC 算法的后面. 而在平均准确率方面, 观察表 6 可知, BICSA 的分类准确率同样在 14 个数据集上超过了全部比较算法, 仅在 PenglungEW 以 93.42%输给了 BMNABC 的 95.07%, 在 Tic-tac-toe 以 84.34%输给了 CSA 的 84.43%. 上述结果都表明了提出的 BICSA 算法在处理特征选择问题具有足够的优越性.

表 5 BICSA 及其比较算法的平均适应度值

Dataset	CSA	SCA	CASO	GWOPSO	BMNABC	CCSA	WOA	BICSA
Breastcancer	0.028 4	0.058 7	0.030 2	0.030 3	0.029 7	0.028 8	0.030 4	**0.026 4**
BreastEW	0.051 4	0.053 4	0.050 4	0.055 5	0.046 7	0.052 0	0.049 6	**0.044 8**
Clean1	0.094 2	0.104 7	0.074 7	0.086 1	0.079 7	0.093 7	0.105 4	**0.064 9**
Clean2	0.034 1	0.045 1	0.027 2	0.029 2	0.028 0	0.032 2	0.033 8	**0.023 7**
CongressEW	0.040 1	0.045 7	0.036 0	0.034 9	0.036 8	0.039 9	0.040 8	**0.034 1**
HeartEW	0.142 4	0.172 2	0.151 1	0.137 2	0.148 7	0.156 2	0.153 1	**0.135 0**
IonosphereEW	0.120 6	0.079 5	0.082 7	0.095 8	**0.066 5**	0.119 2	0.086 3	0.073 4
KrvskpEW	0.053 6	0.092 8	0.033 9	0.037 6	0.050 1	0.057 9	0.061 1	**0.029 7**
Lymphography	0.114 5	0.160 0	0.113 2	0.123 3	0.129 8	0.115 1	0.153 7	**0.101 0**
PenglungEW	0.091 2	0.071 1	0.073 6	0.079 2	**0.047 3**	0.089 8	0.088 8	0.058 2
SonarEW	0.127 0	0.124 0	0.103 2	0.115 7	0.103 3	0.127 8	0.137 1	**0.084 6**
SpectEW	0.168 2	0.163 7	0.159 2	0.168 8	0.155 6	0.165 9	0.169 0	**0.144 4**
Tic-tac-toe	0.165 2	0.226 6	0.170 8	0.166 4	0.165 5	0.166 1	0.165 3	**0.164 8**
Vote	0.048 7	0.044 4	0.041 4	0.045 6	0.040 6	0.049 1	0.047 2	**0.038 9**
WaveformEW	0.174 6	0.181 7	0.159 9	0.165 0	0.163 0	0.173 8	0.177 0	**0.157 7**
WineEW	0.040 1	0.045 1	0.040 7	0.039 8	0.041 7	0.042 3	0.044 0	**0.034 2**

表 6 BICSA 及其比较算法的平均分类准确率

Dataset	CSA	SCA	CASO	GWOPSO	BMNABC	CCSA	WOA	BICSA
Breastcancer	0.978 0	0.973 7	0.976 0	0.976 8	0.976 0	0.977 4	0.975 7	**0.980 0**
BreastEW	0.953 9	0.954 6	0.954 2	0.950 4	0.954 2	0.951 8	0.952 1	**0.958 1**
Clean1	0.908 0	0.908 0	0.930 3	0.923 1	0.929 0	0.908 4	0.898 3	**0.939 1**
Clean2	0.972 4	0.965 9	0.976 4	0.977 1	0.975 4	0.972 4	0.971 4	**0.979 3**
CongressEW	0.962 7	0.959 0	0.965 9	0.968 3	0.965 4	0.963 1	0.960 4	**0.968 7**
HeartEW	0.863 0	0.845 9	0.841 5	0.867 4	0.854 8	0.848 9	0.850 4	**0.869 6**
IonosphereEW	0.883 2	0.927 1	0.920 2	0.908 8	0.932 8	0.883 2	0.903 7	**0.933 8**
KrvskpEW	0.952 9	0.933 1	0.973 1	0.972 0	0.963 5	0.946 8	0.948 1	**0.975 9**
Lymphography	0.891 9	0.855 4	0.891 9	0.887 8	0.881 1	0.889 2	0.855 4	**0.901 4**
PenglungEW	0.912 3	0.923 3	0.931 5	0.926 0	**0.950 7**	0.9151	0.915 1	0.934 2
SonarEW	0.876 9	0.874 0	0.894 2	0.892 3	0.899 0	0.876 0	0.866 3	**0.922 1**
SpectEW	0.837 5	0.842 7	0.842 7	0.836 7	0.848 7	0.841 2	0.834 5	**0.858 4**
Tic-tac-toe	**0.844 3**	0.817 3	0.837 4	0.841 1	0.838 4	0.842 0	0.842 0	0.843 4
Vote	0.956 7	0.958 7	0.960 7	0.958 0	0.958 7	0.954 7	0.953 3	**0.962 7**
WaveformEW	0.828 2	0.818 8	0.845 3	0.840 4	0.843 2	0.832 8	0.830 0	**0.847 2**
WineEW	0.965 2	0.961 8	0.961 8	0.966 3	0.960 7	0.962 9	0.956 2	**0.970 8**

表 7 列出了平均特征选择率的结果. BICSA 在 4 个数据集优于其他比较算法, 虽然其他算法的特征选择率更低, 但这样的特征选择子集对分类准确率产生了一定的影响. 综合来说, 对于特征选择问题, 分类准确率是更加优先的考虑因素, 因为要删除冗余的特征, 同时要尽可能不影响后续学习算法的准确率, 所以应该在追求高准确率的前提下尽量减少特征数值. 且选择的属性数取决于正在处理的数据集和算法本身, 比如 Tic-tac-toe 数据集的属性表示棋盘中的位置, 如果棋盘上信息太少, 将无法确定输赢. 所以为了确保分类的准确性, 数据集的维度不能进行过度压缩.

图 1 显示了所有算法在 16 个数据集上的收敛曲线, 在特征选择问题中, 过早收敛和早期就陷入局部最优解的问题是不容忽视的, 比如在 Breastcancer 数据集上, WOA 在第 11 次迭代时就陷入了局部最优解, 一直到算法迭代结束也无法跳出局部最优解, 从收敛曲线图中可以看出, BICSA 在这两方面要优于其他算法. 同时, 从大部分曲线图中可以观察到, BICSA 在初始适应度值上是很好的, 这说明了算法的初始化策略为算法能够取得最优解起到了明显的效果. 再观察实验结果, BICSA 在 14 个数据集上超过了其他算法, 显示出 BICSA 算法的高效性能.

表 7 BICSA 及其比较算法的平均特征选择率

Dataset	CSA	SCA	CASO	GWOPSO	BMNABC	CCSA	WOA	BICSA
Breastcancer	0.688 9	**0.444 4**	0.688 9	0.733 3	0.577 8	0.600 0	0.644 4	0.666 7
BreastEW	0.466 7	0.140 0	0.386 7	0.533 3	**0.120 0**	0.433 3	0.273 3	0.293 3
Clean1	0.606 0	**0.174 7**	0.518 1	0.716 9	0.319 3	0.630 1	0.690 4	0.475 9
Clean2	0.481 9	**0.210 8**	0.488 0	0.680 7	0.333 7	0.481 9	0.502 4	0.462 7
CongressEW	0.412 5	0.275 0	0.287 5	0.312 5	0.270 0	0.437 5	0.287 5	**0.262 5**
HeartEW	0.538 5	**0.292 3**	0.476 9	0.538 5	0.353 8	0.492 3	0.400 0	0.553 8
IonosphereEW	0.470 6	0.194 1	0.258 8	0.382 4	0.141 8	0.470 6	0.147 1	**0.125 0**
KrvskpEW	0.650 0	**0.322 2**	0.566 7	0.711 1	0.477 8	0.761 1	0.877 8	0.583 3
Lymphography	0.611 1	**0.277 8**	0.555 6	0.500 0	0.411 1	0.611 1	0.533 3	0.555 6
PenglungEW	0.557 5	0.187 7	0.434 5	0.566 2	**0.153 2**	0.553 2	0.341 5	0.304 0
SonarEW	0.570 0	0.253 3	0.466 7	0.623 3	**0.206 7**	0.530 0	0.503 3	0.416 7
SpectEW	0.609 1	**0.236 4**	0.509 1	0.572 7	0.300 0	0.554 5	0.381 8	0.409 1
Tic-tac-toe	0.955 6	**0.644 4**	0.777 8	0.822 2	0.733 3	0.999 9	0.955 6	0.955 6
Vote	0.562 5	0.262 5	0.275 0	0.575 0	0.245 0	0.487 5	0.250 0	**0.212 5**
WaveformEW	0.670 0	**0.320 0**	0.570 0	0.670 0	0.470 0	0.680 0	0.730 0	0.640 0
WineEW	0.661 5	0.499 2	0.553 8	0.630 8	0.488 5	0.676 9	0.480 0	**0.476 9**

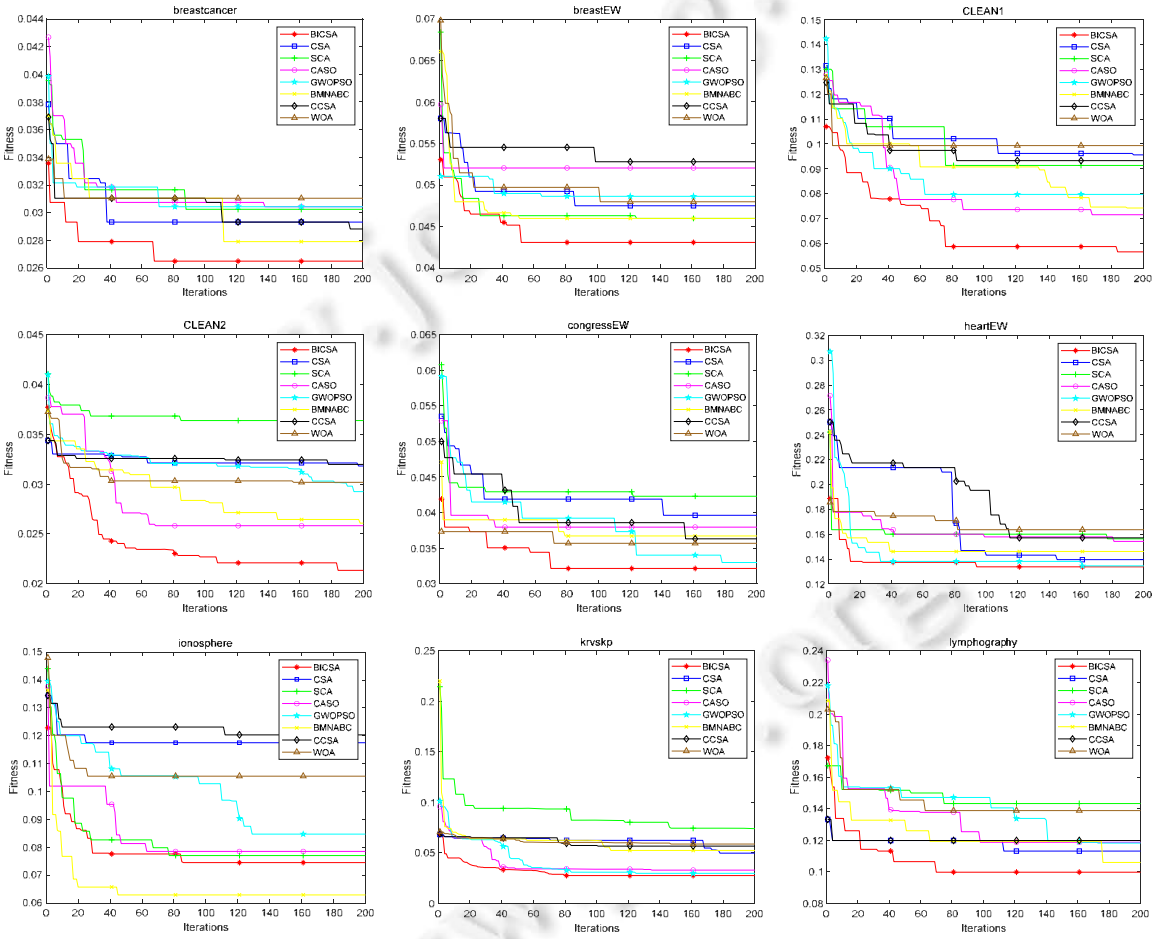

图 1 所有算法在所有数据集上的收敛曲线图

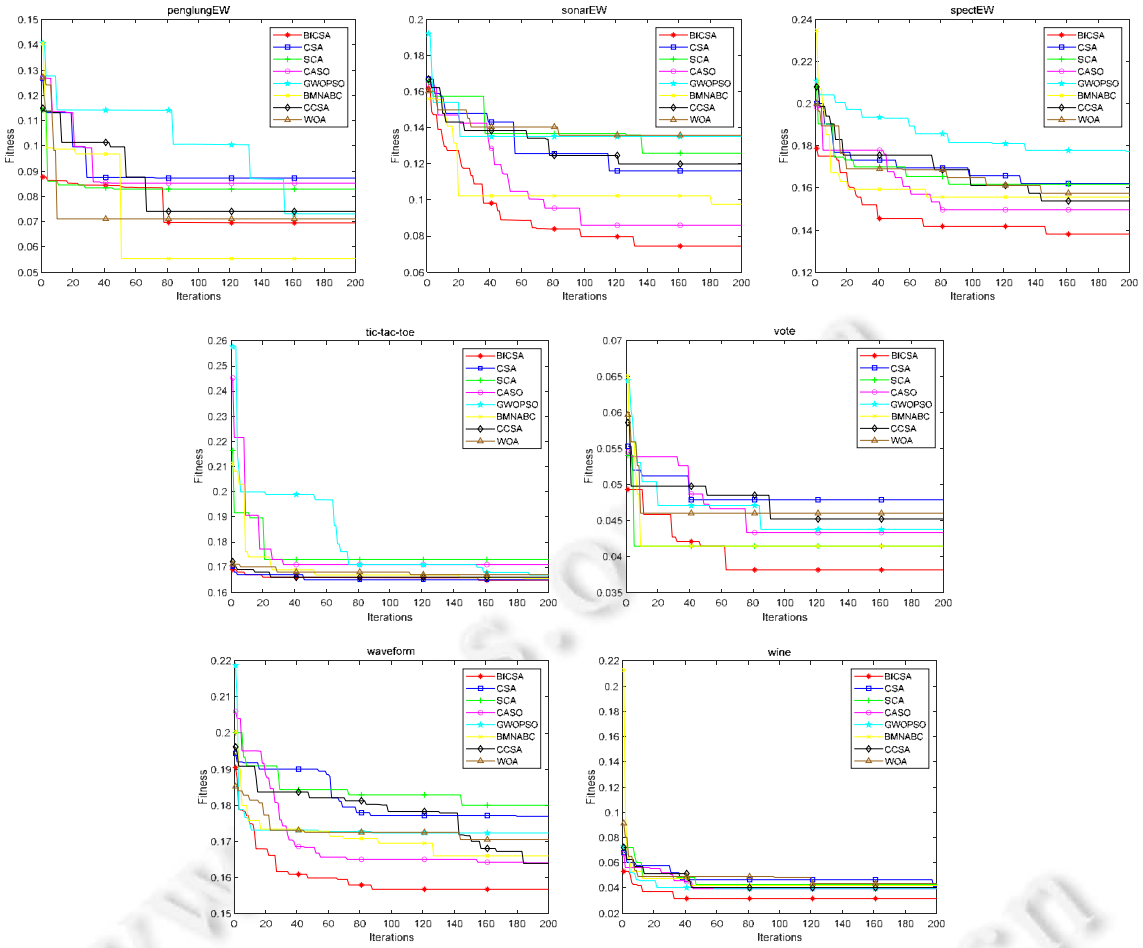

图 1　所有算法在所有数据集上的收敛曲线图(续)

4.4 BICSA算法有效性与显著性分析

为了系统地展示 3 种算子对算法的改进效果，根据改进的两个阶段，提出并实现了 BICSA 的两个子版本——BICSA_INIT 和 BICSA_DE，以检验改进算子的有效性和对算法的影响.

- BICSA_INIT: 对 CSA 算法只添加第 3.1 节提出的初始化方法. 在该算法中，利用混沌 logistic 映射产生混沌初始种群，再通过 OBL 方法选取更加优秀的初始种群，代替了 CSA 中随机生成种群的方法.
- BICSA_DE: 只使用第 3.2 节更新阶段中的 DE 算子来改进 CSA 算法. DE 的添加，可以增强种群多样性，使得算法不易陷入局部最优解，增加了算法找到最优解的概率.

表 8 给出了 BICSA_INIT, BICSA_DE, BICSA 和 CSA 算法的平均适应度值. 从表 8 可以看出, BICSA_DE 和 BICSA 算法在所有数据集上的平均适应度值都超过了 CSA 的结果，BICSA_INIT 算法在 16 个数据集中的 12 个数据集上的结果超过 CSA，仅在 HeartEW, KrvskpEW, Lymphography 和 Tic-tac-toe 这 4 个数据集上表现不佳. 为了能够更加直观地展示比较结果，图 2 给出了平均适应度值的柱状图. 从图 2 可以清晰观察出，在大多数数据集上, BICSA_INIT 和 BICSA_DE 的结果优于 CSA 算法，且 BICSA 的结果要好于 BICSA 的两个子算法.比较结果说明了 3 种改进算子对于算法改进都起到了促进作用，且它们的促进作用可以有效地结合起来，而不会被其中任何一种算子所压制，证实了改进算子对于算法的有效性.

表 8 CSA, BICSA 与 BICSA 子算法的平均适应度值

Dataset	CSA	BICSA_INIT	BICSA_DE	BICSA
Breastcancer	0.028 4	0.027 1	0.028 2	**0.026 4**
BreastEW	0.051 4	0.049 4	0.046 3	**0.044 8**
Clean1	0.094 2	0.092 6	0.069 8	**0.064 9**
Clean2	0.034 1	0.033 1	0.026 5	**0.023 7**
CongressEW	0.040 1	0.037 6	0.035 1	**0.034 1**
HeartEW	0.142 4	0.159 6	0.135 9	**0.135 0**
IonosphereEW	0.120 6	0.110 2	0.084 1	**0.073 4**
KrvskpEW	0.053 6	0.062 1	0.030 9	**0.029 7**
Lymphography	0.114 5	0.129 6	0.109 4	**0.101 0**
PenglungEW	0.091 2	0.086 2	0.060 8	**0.058 2**
SonarEW	0.127 0	0.119 1	0.086 8	**0.084 6**
SpectEW	0.168 2	0.166 3	0.146 8	**0.144 4**
Tic-tac-toe	0.165 2	0.166 0	0.164 9	**0.164 8**
Vote	0.048 7	0.046 0	0.041 4	**0.038 9**
WaveformEW	0.174 6	0.171 4	0.159 5	**0.157 7**
WineEW	0.040 1	0.038 6	0.036 1	**0.034 2**

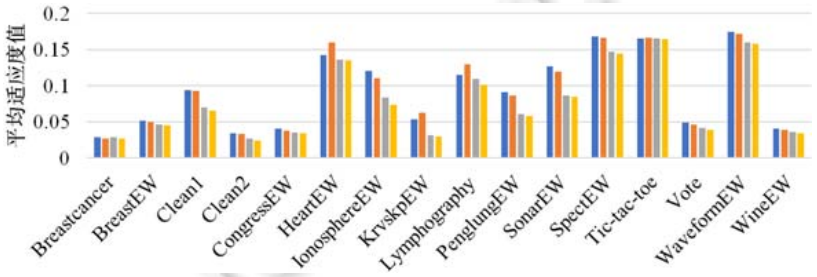

图 2 CSA, BICSA 与 BICSA 子算法的平均适应度值

为进一步分析 BICSA 的显著性, 采用了非参数威尔森秩和检验方法(Wilcoxon rank-sum test)[28], 该检验能够说明 BICSA 作为对照组与其他比较算法(即 CSA, SCA, CASO, GWOPSO, BMNABC, CCSA 和 WOA 算法)在统计学上是否有显著差异(显著性水平设置为 0.05). 根据表 9 给出的检验结果可以看出, 在统计意义上, BICSA 与其他算法在处理大多数数据集上具有显著差异, 这验证了 BICSA 算法的显著性.

表 9 BICSA 及其比较算法的 Wilcoxon 测试结果

Dataset	CSA	SCA	CASO	GWOPSO	BMNABC	CCSA	WOA
Breastcancer	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05
BreastEW	<0.05	<0.05	<0.05	<0.05	0.092	<0.05	<0.05
Clean1	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05
Clean2	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05
CongressEW	<0.05	<0.05	<0.05	0.612	<0.05	<0.05	<0.05
HeartEW	<0.05	<0.05	<0.05	0.101	<0.05	<0.05	<0.05
IonosphereEW	<0.05	0.426	<0.05	<0.05	<0.05	<0.05	0.089
KrvskpEW	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05
Lymphography	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05
PenglungEW	<0.05	0.121	<0.05	<0.05	0.064	<0.05	<0.05
SonarEW	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05
SpectEW	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05
Tic-tac-toe	0.374	<0.05	0.255	0.138	0.820	<0.05	0.421
Vote	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05
WaveformEW	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05
WineEW	<0.05	<0.05	0.091	<0.05	<0.05	<0.05	0.100

5 总 结

本文针对乌鸦搜索算法的一些缺点, 使用了 3 种算子(混沌系统、反向学习方法和差分进化算法)来改进 CSA 形成 ICSA 算法, 然后对 ICSA 进行二元化, 从而提出一种新的乌鸦特征选择算法: BICSA. BICSA 算法在初始化阶段结合了混沌系统和 OBL 方法两种算子, 提出了一种新的种群初始化策略, 大大提升了种群初始解

的质量, 加快了算法的收敛速度; 更新阶段结合 DE 算法增加了种群的多样性, 增强了算法的全局搜索能力. 最后, 实验采用 16 个 UCI 数据集与近年来提出的特征选择算法进行了比较. 实验结果证明, BICSA 对于解决特征选择问题具有一定的优势. 然而, 算法在压缩特征维度上还存在不足之处. 因此, 改进算法的搜索策略, 使其能够在保证较高的分类准确率的同时减少选定特征的数量, 是未来研究的重点. 此外, 在今后的工作中, 将尝试使用自适应参数来代替固定参数, 相信这将会进一步提升算法的性能.

References:

[1] Bennasar M, Hicks Y, Setchi R. Feature selection using joint mutual information maximisation. Expert Systems with Applications, 2015, 42(22): 8520−8532.

[2] Ekbal A, Saha S. Joint model for feature selection and parameter optimization coupled with classifier ensemble in chemical mention recognition. Knowledge-based Systems, 2015, 85: 37−51.

[3] Dash M, Liu H. Feature selection for classification. Intelligent Data Analysis, 1997, 1(1-4): 131−156.

[4] Askarzadeh A. A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. Computers & Structures, 2016, 169: 1−12.

[5] Gleick J. Chaos: Making a New Science. New York: Viking Penguin, 1987.

[6] Tizhoosh HR. Opposition-based learning: A new scheme for machine intelligence. In: Proc. of the Int'l Conf. on Computational Intelligence for Modelling, Control and Automation and Int'l Conf. on Intelligent Agents, Web Technologies and Internet Commerce. 2005. 695−701.

[7] Wang H, Wu Z, Rahnamayan S, Liu Y, Ventresca M. Enhancing particle swarm optimization using generalized opposition-based learning. Information Sciences, 2011, 181(20): 4699−4714.

[8] Price K. Differential evolution: A fast and simple numerical optimizer. In: Proc. of the North American Fuzzy Information Processing. 1996. 524−527.

[9] Chu B, Li ZS, Zhang ML, Yu HH. Research on improvements of feature selection using forest optimization algorithm. Ruan Jian Xue Bao/Journal of Software, 2018, 29(9): 2547−2558 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5395.htm [doi: 10.13328/j.cnki.jos.005395]

[10] Emary E, Zawbaa HM, Hassanien AE. Binary ant lion approaches for feature selection. Neurocomputing, 2016, 213: 54−65.

[11] Guyon I, Gunn S, Nikravesh M, Zadeh LA. Feature Extraction: Foundations and Applications. New York: Springer, 2006.

[12] Xue B, Zhang M, Browne WN, Yao X. A survey on evolutionary computation approaches to feature selection. IEEE Trans. on Evolutionary Computation, 2016, 20(4): 606−626.

[13] Yang XS. Nature-inspired Metaheuristic Algorithms. Luniver Press, 2008.

[14] Aziz MAE, Hassanien AE. Modified cuckoo search algorithm with rough sets for feature selection. Neural Computing & Applications, 2018, 1−10.

[15] Hou Y, Li J, Yu H. BIFFOA: A novel binary improved fruit fly algorithm for feature selection. IEEE Access, 2019, 7: 81177−81194.

[16] Kashef S, Nezamabadi-Pour H. An advanced ACO algorithm for feature subset selection. Neurocomputing, 2015, 147: 271−279.

[17] Mafarja MM, Mirjalili S. Whale optimization approaches for wrapper feature selection. Applied Soft Computing, 2018, 62: 441−453.

[18] Al-Tashi Q, Kadir SJA, Rais HM. Binary optimization using hybrid grey wolf optimization for feature selection. IEEE Access, 2019, 7: 39496−39508.

[19] Aziz MEAE. A hybrid method of sine cosine algorithm and differential evolution for feature selection. In: Proc. of the Int'l Conf. on Neural Information Processing. Cham: Springer, 2017. 145−155.

[20] Sayed GI, Hassanien AE, Azar AT. Feature selection via a novel chaotic crow search algorithm. Neural Computing & Applications, 2019, 31(1): 171−188.

[21] Ibrahim RA, Elaziz MA, Oliva D. An opposition-based social spider optimization for feature selection. Soft Computing, 2019, 23: 1−21.

[22] Lu H, Wang X, Fei Z, Qiu M. The effects of using chaotic map on improving the performance of multiobjective evolutionary algorithms. Mathematical Problems in Engineering, 2014, Article ID 924652. [doi: 10.1155/2014/924652]

[23] May R. Simple Mathematical Models with Very Complicated Dynamics. New York: Springer, 2004.

[24] Wang H, Wu Z, Rahnamayan S. Enhancing particle swarm optimization using generalized opposition-based learning. Information Sciences, 2011, 181(20): 4699−4714.

[25] Zawbaa HM, Hafez AI, Emary E. Sine cosine optimization algorithm for feature selection. In: Proc. of the Int'l Symp. on Innovations in Intelligent Systems & Applications. IEEE, 2016. 1−5.

[26] Beheshti Z. BMNABC: Binary multi-neighborhood artificial bee colony for high-dimensional discrete optimization problems. Cybernetics and Systems, 2018, 49(7): 452−474.

[27] Too J, Abdullah AR. Chaotic atom search optimization for feature selection. Arabian Journal for Science and Engineering, 2020, 45: 6063−6079.

[28] Zhang B, Yang X, Hu B. OEbBOA: A novel improved binary butterfly optimization approaches with various strategies for feature selection. IEEE Access, 2020, 8: 67799−67812.

附中文参考文献:

[9] 初蓓, 李占山, 张梦林, 于海鸿. 基于森林优化特征选择算法的改进研究. 软件学报, 2018, 29(9): 2547−2558. http://www.jos.org.cn/1000-9825/5395.htm [doi: 10.13328/j.cnki.jos.005395]

廉杰(1997－), 女, 硕士, 主要研究领域为演化计算, 特征选择.

李占山(1966－), 男, 博士, 教授, 博士生导师, CCF 专业会员, 主要研究领域为机器学习, 约束推理.

姚鑫(1996－), 男, 硕士, 主要研究领域为演化计算, 特征选择.