# 基于 PAGA 的 RTS 游戏多单元控制方法研究

杨 震 张万鹏 刘鸿福 魏占阳

(国防科技大学智能科学学院 长沙 410000)

**摘 要** 实时战略游戏(RTS)中的单元控制在人工智能(AI)领域是一个具有挑战性的问题。这类游戏是实时约束的,并且具有庞大的状态和行动空间,智能算法已不能很好地解决这类问题。在脚本空间搜索策略对战斗场景中的多单元进行控制,可以有效地克服巨大的分支因子带来的不利影响。文中运用自适应遗传算法(Adaptive Genetic Algorithm)在脚本空间进行搜索,为战斗场景中的多单元提供良好的行动序列,实现了对单元的有效控制。实验结果表明,提出的 PAGA(Portfolio Adaptive Genetic Algorithm)是可行且有效的,在大规模单元控制中的性能优于现行算法。

**关键词** RTS 游戏,AI,自适应遗传算法,多单元控制

**中图法分类号** TP273+.2 **文献标识码** A

## Research on Multi-units Control Method in RTS Games Based on PAGA

YANG Zhen ZHANG Wan-peng LIU Hong-fu WEI Zhan-yang

(College of Intelligence Science and Engineering, National University of Defense Technology, Changsha 410000, China)

**Abstract** Unit control in real-time strategy games (RTS) is a challenging issue in the field of artificial intelligence (AI). Such games are constrained in real time, and have a large state and action space, which make intelligent algorithms do not solve this type of problem. By controlling the multi-units in the battle scene by searching strategy in the script space, it is possible to effectively overcome the adverse effects caused by the huge branching factor. This paper used adaptive genetic algorithm to search in scripting space to provid a good sequence of actions for multi-units in the battle scene and control the unit. Experiments show that the proposed PAGA (Portable Adaptive Genetic Algorithm) is feasible and effective, and its performance is superior to the current algorithms in large-scale unit control.

**Keywords** RTS game, AI, Adaptive genetic algorithm, Multi-units control

即时战略游戏(RTS)是人工智能(AI)领域的主要挑战之一。近几年来,许多研究者在基于案例推理和规划[1]、机器学习[2]、深度学习[3]以及启发式和对抗式搜索[4]领域将即时战略游戏(RTS)作为测试平台来研究具有挑战性的问题。即时战略游戏(RTS)面临的一个巨大挑战是它们的状态和行动空间巨大,且必须实时控制大量的单元,因此它们在很多研究层次上都非常复杂。在游戏中,玩家可以控制多单元一起战斗,集合资源,建立基地,训练部队,并用各种策略消灭对手。此外,与 Go 等游戏相比,"星际争霸"中的 AI 算法必须处理隐藏的信息(战争迷雾);对手的基地最初是隐藏的,必须在整个游戏过程中不断探索才能知道(或猜测)对手所处的状态。在商业 RTS 游戏中,AI 玩家大多是通过脚本实现的。脚本是一个固定的策略,告诉单位在某些条件下应该做什么。这些脚本不能给玩家提供很强的 AI 对手,但是它们非常高效。虽然商业 RTS 游戏仍然主要使用脚本化的 AI 玩家,但研究者已经开发出了一些更好的方法。例如,Churchil 等[5]提出的投资组合贪婪搜索(PGS)算法以及 Justesen 等[6]提出的基于脚本和群集的 UCT 算法,都是建立在不为每个单元选择采取哪种行动的基础上,而是建立在为每个单元选择一个行动所使用的脚本的基础上,从而大大缩减了搜索空间。与其他搜索算法相比,这些算法在大中型单位战斗中的性能明显优于纯脚本玩家,并且显示出了巨大的效率。Wang 等[7]提出了一个针对策略选择的投资组合在线进化算法(POE)来控制 RTS 作战单位,没有像 PGS 那样使用爬山算法,而是用进化算法在脚本任务空间中进行搜索,并验证了他们的算法优于 PGS。Usunier 等[8]提出用深度神经网络控制器从游戏引擎给出的原始状态特征来处理微观管理情况的方法,提出了一种启发式强化学习算法,允许使用确定性策略来收集学习痕迹,将策略空间中的直接探索与反向传播相结合以达到控制单元的目的。

本文主要研究在 RTS 游戏下战斗场景中的多单元控制方法。针对 RTS 游戏的实时性、不确定性和巨大的状态动作空间造成的巨大分支因子问题,本文将自适应遗传算法运用到 RTS 游戏的脚本策略空间并进行搜索,而不是直接运用在单元的动作选择上。这样可以有效地克服多单元战斗场景中的巨大分支因子,为单元提供良好的行动序,以此来获得大规

本文受 2017 年国家自然科学基金项目(61403411),高动态环境下低可探测性飞行器自主任务规划方法研究项目资助。

**杨 震**(1994-),男,硕士生,主要研究方向为任务规划、博弈推理,E-mail:yangzhen7319@163.com(通信作者);**张万鹏**(1981-),男,博士,副研究员,主要研究方向为智能规划、博弈推理;**刘鸿福**(1983-),男,博士,副研究员,主要研究方向为人工智能、任务规划;**魏占阳**(1994-),男,硕士生,主要研究方向为任务规划、集群对抗。

模作战单元的战斗胜利。实验结果表明,本文所提出的 PA-GA 可以有效地对战斗场景中的多单元进行控制,并获得比现行算法更好的性能。

## 1 问题描述

### 1.1 即时战略(RTS)游戏

即时战略(RTS)游戏是复杂的对抗性领域,通常模拟大量单位之间的战斗,对人工智能提出了重大挑战。在即时战略(RTS)游戏中,玩家需要发展经济(收集资源和建立基地)和军事力量(训练单位和研究技术),以击败对手(摧毁他们的军队和基地)。从理论角度来看,RTS 游戏和传统棋类游戏(如国际象棋)之间存在如下主要区别。

1)即时战略游戏(RTS)是同时进行的,其中多个玩家可以同时发出行动。此外,这些行动是持续的,即行动不是即时的。RTS 游戏是"实时的",实际上意味着每个玩家都有很少的时间来决定下一步的行动。在国际象棋游戏中,玩家可能有几分钟的时间来决定下一个动作;在星际争霸中,游戏以 24 帧/s 的速度执行,这意味着在游戏状态改变之前,玩家控制单元的行动速度可以达到 42 ms 便执行一次动作。

2)大多数 RTS 游戏都是部分可观察的,即玩家只能看到已经探索过的地图部分。

3)大多数 RTS 游戏是非确定性的。

4)最后,RTS 游戏的复杂性,无论是在状态空间大小还是在每个决策周期可用的行动数量方面,都是非常大的。对于 RTS 作战场景,任何状态下可能的行动次数都是每个单位所有可能行动的组合,大约为 $L^U$。其中,$L$ 是每个单位合法行动的平均数量,$U$ 是可以执行行动的单位数量[9]。

受这些因素的影响,没有一定程度的抽象定义或其他一些简化,用于经典棋盘游戏的标准技术(如游戏树搜索)不能被直接应用于 RTS 游戏。现有的研究通常定义一个名为 Portfolio 的脚本策略集合,用于对游戏进行抽象,从而有效地减少了直接在状态和动作空间搜索造成的巨大分支因子问题。本文基于这种形式的抽象来进行算法的设计。

### 1.2 星际争霸实验平台

StarCraft:Brood War 是暴雪公司于 1998 年发布的一款非常受欢迎的 RTS 游戏。在游戏中,玩家必须选择 3 个种族之一:人族,神族或虫族。每个种族都有自己的优势和劣势。每个玩家从 4 个可以聚集资源和建造建筑物的工人开始,为了赢得游戏,玩家必须首先收集资源(矿物和瓦斯气体)。随着资源越来越充足,玩家需要分配它们来创造更多的建筑物,研究新技术和训练攻击单位。单位必须分配完成不同的任务,如侦察、防御和攻击。在执行所有这些任务的同时,玩家还需要战略性地了解当前地图的几何形状,以便决定将新建筑物放在哪里,或者设置防御性前哨的位置。最后,当两个玩家的进攻单位相遇时,每个玩家必须快速地操纵每个单位以便进行战斗,这需要对每个单位进行快速和被动的控制。Star Craft 提供了游戏的不完整状态信息,因为对手的基地最初是隐藏的,必须由侦察单位进行探索,这与多个智能体必须实时控制的事实相结合,为策略选择和推荐提供了具有挑战性的环境。一般将策略决策过程分解为微观管理任务和宏观管理任务。把微观管理定义为对个体单位和建筑物的战术控制;把宏观管理定义为在整个游戏中推进哪些技术,以便在适当的时候获得适当的单位组合的长期高层规划和决策[10]。本文中主要侧重于微观管理,即对单元进行控制的研究。

由于星际争霸的源代码不可用,为了模拟游戏中的游戏战斗,Churchill 等[5]创建了一个用 C++编写的 Sparcraft 的开源系统。这个系统很好地模拟了星际争霸中的战斗功能,其作战单元可以被赋予攻击、移动和等待命令。作战单元的属性,如武器的类型、冷却时间、速度、尺寸等,完全是基于 StarCraft 建模的。本文使用的是 Sparcraft 的 Java 版本,名为 Jarcraft,由 Tillman 开发。

### 1.3 游戏脚本策略

在 RTS 游戏中,对单元进行控制最简单和最常用的技术是使用硬编码的脚本(script)策略。在这里定义一个策略,其中策略 $\bar{\sigma}: S\times U\rightarrow M$ 是静态的脚本规则。对于给定的状态 $s$ 和在状态 $s$ 下的单元 $u$,通过执行所选的策略 $\bar{\sigma}$ 来产生下一时刻要执行的动作 $m$,即 $m=\bar{\sigma}(s,u)$。基于对现行算法的研究和算法间的对比实验,本文使用以下策略来组成脚本策略集以验证算法。

1)NOKAV:指示一个单元 $u$ 在可攻击的范围内攻击 $dpf(u)/hp(u)$ 最高的敌方单元,其中 $dpf(u)$ 代表敌方单位可以造成的伤害,$hp(u)$ 代表敌人的生命值。如果没有敌人在范围内,单位将移向最近的敌人。NOKAV 策略让友方单位配合对敌人造成最高的伤害。

2)Back:指示一个单元 $u$ 在重新装载时远离最近的敌方单元。

3)BackFar:指示一个单元 $u$ 在重新装载时远离最近的友方单元。

4)BackClose:指示一个单元 $u$ 在重新装载时靠近最近的友方单元。

5)Forward:指示一个单元 $u$ 在重新装载时靠近最近的敌方单元。

6)ForwardFar:指示一个单元 $u$ 在重新装载时靠近最远的敌方单元。

## 2 自适应遗传算法

### 2.1 遗传算法

遗传算法[11]是由美国 Holland 教授于 1975 年提出的,是通过借鉴生物进化的一些特征,模拟生物的自然选择和遗传机制而形成的一种自适应全局优化概率搜索算法。在不断变化的环境中,只有那些适应性好的个体才能存活下来,通过遗传机制将更能适应环境的特性传递给后代。使用选择、交叉、变异等运算操作进行有组织的但又随机的信息交互,用适应度来衡量个体的好坏,适应度高的个体不断被继承下来,重复此过程,直到种群满足某种收敛指标为止。该算法的特点是全局寻优能力、适应性强、解决非线性问题时具有较强的鲁棒性、对问题没有特定限制、计算过程简单、对搜索空间没有特殊要求、易于与其他算法结合等特点,在函数优化、图像处理、系统辨识、自动控制、经济预测和工程优化等领域得到了广泛的应用[12],在求解 NP 完全问题方面是一种较为有效的全局方法。利用遗传算法求解问题的流程[13]如图 1 所示。

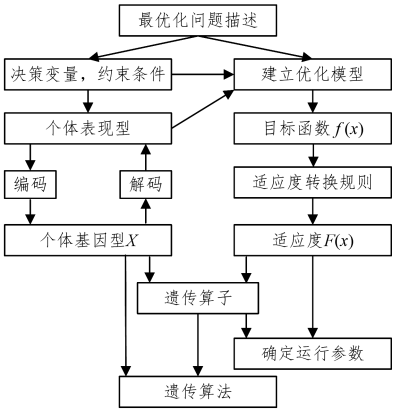



图 1　遗传算法的求解过程

### 2.2　遗传算法的改进

简单的遗传算法在处理大规模、多变量问题时的收敛速度较慢、耗时较多,其中操作参数(如种群大小 $N$、杂交率 $P_c$、变异率 $P_m$)的选择对遗传算法优化结果的影响是举足轻重的。本文主要研究固定种群大小在战斗场景下的单元控制,所以杂交率 $P_c$ 和变异率 $P_m$ 直接影响了算法的性能。然而,传统的遗传算法存在着严重的缺点,其中的杂交率 $P_c$ 和变异率 $P_m$ 的值是固定的,优良个体和劣质个体都经过相同概率的交叉和变异进行操作。对于优良个体,我们应该减小交叉变异概率,使之能够尽量保存;而对于劣质个体,我们应该增大交叉变异概率,使之能够尽可能地改变劣质的状况。交叉概率 $P_c$ 的大小决定着种群的丰富程度,$P_c$ 越大,种群的丰富程度越高,但是优良个体被破坏的可能性也越大;变异概率 $P_m$ 的大小决定着能否跳出局部极值而找到全局最优解,$P_m$ 越大,越容易跳出局部极佳;从而找到全局最优解,但是过大的 $P_m$ 值又会使算法沦为随机搜索算法。另一方面,如果 $P_c$ 和 $P_m$ 值太小,则不容易产生新的个体,会使得进化停滞不前。针对以上情况,本文采用 Srinivas 等[14]提出的自适应遗传算法,动态调整 $P_c$ 和 $P_m$ 的值。该算法旨在通过不同的方式实现搜索和随机性之间的权衡,根据适合度值自适应地改变 $P_c$ 和 $P_m$ 的值,当群体倾向于停留在局部最优(也就是群体适应度集中,多样性比较差)时 $P_c$ 和 $P_m$ 的值增加,并且当群体在解空间中散布(也就是群体适应度分散,多样性比较高)时减小。交叉概率 $P_c$ 和变异概率 $P_m$ 按式(1)、式(2)进行自适应调整。

$$P_c=\begin{cases}k_1(f_{\max}-f')/(f_{\max}-f_{\text{ave}}), & f'\geqslant f_{\text{ave}}\\ k_2 & f'\leqslant f_{\text{ave}}\end{cases} \tag{1}$$

$$P_m=\begin{cases}k_3(f_{\max}-f)/(f_{\max}-f_{\text{ave}}), & f\geqslant f_{\text{ave}}\\ k_4, & f\leqslant f_{\text{ave}}\end{cases} \tag{2}$$

其中,$f_{\max}$为种群的最大适应度;$f_{\text{ave}}$为种群的适应度平均值;$f'$为两个交叉个体中适应度较大者的适应度;$f$ 为变异个体的适应度。$0<k_1,k_2,k_3,k_4\leqslant 1$ 为常数。

### 2.3　运用到 RTS 游戏的算法原理

遗传算法以适应度为依据的逐代搜索过程,主要由编码机制、控制参数、适应度函数和遗传算子 4 部分组成。本文将自适应遗传算法应用到 RTS 游戏的单元控制上,提出 PAGA。为了解决 RTS 游戏由于不确定性和复杂性造成的巨大分支因子,将自适应遗传算法运用在对脚本策略的优化选择上,而不是直接运用在对单元动作的选择上。主要计算过程如下:

1)确定编码机制,生成初始种群。在本文的实验中,初始化单元编码全部使用 NOKAV 策略进行单元控制,在初始的几个控制步骤中,采用自适应遗传算法进行策略选择。例如,单元的基因组(genome)共有 30 轮的策略选择,设定前 10 轮策略选择采用自适应遗传算法在策略集中选择进行控制,决定单元的下一步动作;剩下的 20 轮策略采用初始化的 NOKAV 进行控制,对手的单元全部采用 NOKAV 进行初始化控制,决定单元的初始动作。这样做可以在有限的计算时间内快速地在战斗前期找到最优的单元控制序列。

2)计算种群中每个个体的适应度值。评估函数接受当前的状态 $s$,给定一个单元控制的基因组 $G$,创建一个当前状态的副本 $s'$,并使用 $G$ 中的脚本从当前的状态 $s'$ 进行模拟,直至策略集模拟结束或者个体的生命值为 0;然后使用 LTD2[15]方法来评估最终的状态。LTD2 评估函数如下:

$$LTD2(s)=\sum_{u\in U_1}\sqrt{hp(u)}\times dpf(u)-\sum_{u\in U_2}\sqrt{hp(u)}\times dpf(u) \tag{3}$$

LTD2 分别使用由玩家 1 和玩家 2 控制的单元集 $U_1$ 和 $U_2$ 以及每个单元的生命值 $hp(u)$和伤害值 $dpf(u)$来计算在状态 $s$ 下的适应度值。

3)交叉算子。由式(1)自适应地计算交叉概率。交叉概率 $p_c$ 选择若干父体并进行配对,按照交叉算法的规则生成新个体。

4)变异算子。由式(2)自适应地计算变异概率。为了保持种群个体的多样性,防止陷入局部最优,需要按照某一变异概率 $p_m$ 随机确定变异个体,并实行相应的变异操作。

5)迭代终止条件。若满足预定的终止条件(达到最大迭代次数),则停止迭代,所得到的当前最优或者次优的策略集用于控制单元的行动;否则转至 2),计算新一代种群中每个个体的适应度值。

PAGA 的伪代码如算法 1 所示。

**算法 1**　PAGA

```
1. 函数　PAGA(状态 s)
2. 基因组[] aga
3. 初始化(aga,s)
4. While 当前迭代次数<迭代限制 do
5. 交叉(aga)
6. 变异(aga)
7. 选择(aga)
8. Return AGA 中最好的基因组

9. 函数　初始化(基因组[] aga,状态 s)
10. For x=1 to 种群大小 do
11. N=个体数
12. M=基因数量
13. 基因组 g=new 基因组(N,M)
14. aga.add(g)
15. fill(g.NOKAV)

16. 函数　交叉(基因组[] aga,状态 s)
17. For 基因组 g in aga do
18. For i in M do
19. For j in N do
20. If random(0,1)<PM then
21. 对 g[M][N]执行交叉操作
```

```
22. 函数 变异(基因组[] aga)
23. For 基因组 g in aga do
24. For i in M do
25. For j in N do
26. If random(0,1)<PM then
27. 对 g[M][N]执行变异操作
28. 函数 选择(基因组[] aga,状态 s)
29. For 基因组 g in aga do
30. 评估(g,s)
31. 对 g 执行选择操作
32. 函数 评估(基因组[] aga,状态 s)
33. 状态 s′=s
34. s′=模拟(s′,g,时间限制)
35. Return s′. LTD2()
36. 函数 模拟(基因组[] aga,状态 s,时间限制 t)
37. For 时间=1 to t do
38. 友方单元脚本策略=g
39. 敌方单元脚本策略=NOKAV
40. Return s
```

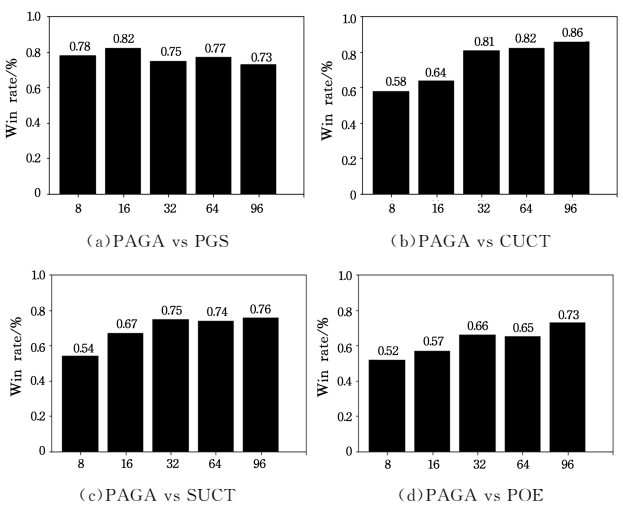
## 3 实验结果分析

### 3.1 实验环境设置

本文的所有实验都在 Windows 10 的 Intel(R) Core i7-7700HQ @ 2.80GHz 四核的 CPU 和 8GB DDR4 2400MHz 的 RAM 上运行。用 Java 编程软件 eclipse 运行星际争霸模拟包,jarcraft 进行算法的调试和验证。运行模拟包后会显示一个地图,地图中的单元可以在地图上的任意位置移动。为了保证算法验证的公平性,初始化时双方的单位垂直排列在地图的两边,且不在对方的攻击范围之内。在每个算法的验证过程中,每个战斗玩家可以控制 $N$ 个作战单元,其中设置一半为龙骑兵(星际争霸中远程单位),一半为狂热分子(星际争霸中近战单元)。算法测试的作战单元的设置为 $N=8,16,32,64,96$。

### 3.2 实验结果的对比分析

实验分别对比了 PAGA 与基于脚本和集群的 UCT 算法、PAGA 与 PGS 算法、PAGA 与 POE 算法的性能。每组比对算法在相同单元下各进行 1000 次的对战实验。环境的初始化设置保持一致,实验结果如图 2 所示。




(a)PAGA vs PGS (b)PAGA vs CUCT

(c)PAGA vs SUCT (d)PAGA vs POE

图 2 PAGA 与各个算法之间的赢率对比

实验的结果以单元数量为横坐标,以赢率为纵坐标呈现。当一方作战单元完全消灭另一方单元时,判定控制这一方单元的算法在此回合的对比中胜出。其中,不同算法间的对比在相同的作战单元下各进行 1000 次的对战实验,最后计算算法各自获胜的比例。从结果图可以得到如下结论:

1)在 PAGA 与 PGS 算法的实验对比中可以看到,本文提出的算法在不同规模的单元中都有很高的胜率,且胜率的波动很小,表明本文提出的算法具有一定的稳定性。

2)在 PAGA 与基于脚本和集群的 UCT 算法的对比中可以发现,在对大规模的单元进行控制时,本文算法获得了更高的胜率。但是在对规模较小的单元进行控制时,其胜率相对较低。研究发现,在较小规模的单元中,单元之间的配合会相对较少,限制了该算法的性能。而在大规模的单元控制中,单元之间的配合较多,PAGA 能有效地在脚本策略空间进行搜索,提高赢率。

3)在 PAGA 与 POE 算法的对比中,除了对较大规模的单元控制有明显的优势胜率之外,其他一般规模的胜率不是特别明显。之所以在较大规模的单元能获得明显的优势,是因为本文算法采用了参数自适应的方式进行算法调节,对环境有很强的鲁棒性。

**结束语** 本文的主要贡献在:针对即时战略(RTS)游戏中的多单元控制,提出了一种更加有效的自适应控制算法。通过与现有 3 种最先进算法的对比表明,本文提出的 PAGA 均获得了很好的性能。针对大规模的单元进行控制时,PAGA 获得了更好的性能,这说明了它能适应更复杂的情况。下一步的工作是,进一步优化对自适应参数的学习,使之更加适应实验的环境。例如,可以加入基于案例推理和强化学习,离线地在即时战略游戏数据集中学习参数,优化模型。这对即时战略游戏更高层控制的研究具有重要的意义。

## 参 考 文 献

[1] MISHRA K,SUGANDH N,RAM A. Case-Based Planning and Execution for Real-Time Strategy Games[C]// International Conference on Case-Based Reasoning: Case-Based Reasoning Research and Development. Springer-Verlag,2007:164-178.

[2] SYNNAEVE G,BESSIERE P. A Bayesian Model for Opening Prediction in RTS Games with Application to StarCraft[J]. Proceedings of IEEE Cig Seoul South Korea,2011,32(14):281-288.

[3] FOERSTER J,NARDELLI N,FARQUHAR G,et al. Stabilising Experience Replay for Deep Multi-Agent Reinforcement Learning[J]. arXiv:1702.08887,2017.

[4] BARRIGA N A,STANESCU M,BURO M. Puppet Search:Enhancing Scripted Behavior by Look-Ahead Search with Applications to Real-Time Strategy Games[C]// AIIDE. 2015.

[5] CHURCHILL D,BURO M. Portfolio greedy search and simulation for large-scale combat in starcraft[C]// Computational Intelligence in Games. IEEE,2013:1-8.

[6] JUSTESEN N,TILLMAN B,TOGELIUS J,et al. Script- and cluster-based UCT for StarCraft[C]// Computational Intelligence and Games. IEEE,2014:1-8.

[7] WANG C,CHEN P,LI Y,et al. Portfolio online evolution in Star Craft[C]// AAAI Conference on Artificial Intelligence and InteractiveDigital Entertainment. 2016:114-120.

(下转第 125 页)

(3)实验表明,ESHC 算法更适合大量样本且复杂分布的数据集分类。将表 3、表 5 与表 4、表 6 进行对比发现 Enroll 数据集和 Adult 数据集上 ESHC 算法的性能改善比在其余两个数据集上的性能改善较大。从数据集样本数量来看,Enroll 数据集和 Adult 数据集的样本数比其余两个数据集的样本数大很多。如 Enroll 数据集和 Adult 数据集都有 40000 多个样本,而 Abalone 数据集和 Yeast 数据集分别只有 4177 个样本和 1484 个样本。

(4)一般来说,集成分类器比单一分类器的时间消耗大,这是因为集成分类器需要训练多个分类器,测试的时间比较多。但图 2 显示 ESHC 算法的训练时间在 Abalone 数据集上才是最高的;图 3 显示 ESHC 算法的测试时间在 4 个数据集上均不是最高的,这可以说明 ESHC 算法的训练时间和测试时间并没有太大增加。这是因为该算法从第二层开始并不是用整个训练数据集来训练基分类器,而是各基分类器在它对应的训练子集上进行训练,减少了时间消耗。

**结束语**　目前,集成学习在各领域被成功应用,因此集成学习的理论和算法的研究成为了机器学习领域的一个热点,越来越多的集成学习算法被应用于分类中。在考虑如何提高集成分类器的性能时,应充分认识分类器的局部有效性,利用合理的结构和算法避免其对分类的不利影响。本文采用梯度优化的思想,分层训练各个子分类器,利用层次化结构对误分类的样本进一步学习和预测,使分类性能得到提高。

在梯度优化算法的训练阶段,如何选择训练样本是一个重点,如果选择的训练样本与预测时进入到该分支的样本分布不一致,将会严重影响分类性能。该算法的另一个缺点是误差传递,如果在某一结点处发生了错误判别,这个错误将向下传递。这些问题将是进一步研究的重点。

## 参 考 文 献


[1] DIETTERICH T G. Machine learning research four current directions[J]. AI Magazine,1997,18(4):97-136.

[2] 唐伟,周志华. 基于 Bagging 的选择性聚类集成[J]. 软件学报,2005,16(4):496-502.

[3] 周志华. 机器学习[M]. 北京:清华大学出版社,2016.

[4] 唐春生,金以慧. 基于全信息矩阵的多分类器集成方法[J]. 软件学报,2003,14(6):1103-1109.

[5] WOLPERT D H. Stacked generalization[J]. Neural Networks,1992,5(2):241-259.

[6] ZHOU Z H,WU J X,TANG W. Ensembling neural networks:many could be better than all[J]. Artificial Intelligence,2002,137(1):239-263.

[7] KO A R,SABOURIN R,BRITTO A S. From dynamic classifier selection to dynamic ensemble selection[J]. Pattern Recognition,2008,41(5):1718-1731.

[8] 方敏. 集成学习的多分类器动态融合方法研究[J]. 系统工程与电子技术,2006,28(11):1759-1762.

[9] MITCHELL H B. Ensemble learning in data fusion:Concepts and ideas[M]. Springer Berlin Heidelberg,2012.

[10] ROJARATH A,SONGPAN W,PONG-INWONG C. Improved ensemble learning for classification techniques based on majority voting[C]// IEEE International Conference on Software Engineering and Service Science (ICSESS). IEEE,2017:107-110.

[11] ZHANG L,ZHOU W. Sparse ensembles using weighted combination methods based on linear programming[J]. Pattern Recognition,2011,44(1):97-106.

[12] 朱波,陈科,徐君,等. 平均分布集成策略:一种新的分类器融合方法[J]. 小型微型计算机系统,2016,37(7):1546-1550.

[13] YU Z W,WANG D X,JANE Y,et al. Progressive subspace ensemble learning[J]. Pattern Recognition,2016,60(C):692-705.

[14] DUTTA A,DASGUPTA P. Ensemble learning with weak classifiers for fast and reliable unknown terrain classification using mobile robots[J]. IEEE Transactions on Systems,Man,and Cybernetics:Systems,2017,47(11):2933-2944.

[15] TOLOMEI G,SILVESTRI F,HAINES A,et al. Interpretable Predictions of Tree-based Ensembles via Actionable Feature Tweaking[C]// Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York:ACM,2017:465-474.

[16] 高锋,黄海燕. 基于邻域混合抽样和动态集成的不平衡数据分类方法[J]. 计算机科学,2017,44(8):225-229.

[17] LUO H Y,WANG D Y,YUE C Q,et al. Research and application of a novel hybrid decomposition-ensemble learning paradigm with errorcorrection for daily PM10 forecasting[J]. Atmospheric Research,2018,201:34-45.

[18] ZHANG L,SHAH S K,KAKADIARIS I A. Hierarchical multi-label classification using fully associative ensemble learning[J]. Pattern Recognition,2017,70:89-103.

[19] 张春霞,张讲社. 选择性集成学习算法综述[J]. 计算机学报,2011,34(8):1399-1410.

[20] BREIMAN L. Random forests[J]. Machine Learning,2001,45(1):5-32.

[21] WU Y,LIU D,JIANG H. Length-changeable incremental extreme learning machine[J]. Journal of Computer Science and Technology,2017,32(3):630-643.

[22] LIU D,WU Y,JIANG H. FP-ELM:An online sequential learning algorithm for dealing with concept drift[J]. Neurocomputing,2016,207(26):322-334.


(上接第 104 页)


[8] USUNIER N,SYNNAEVE G,LIN Z,et al. Episodic Exploration for Deep Deterministic Policies:An Application to StarCraft Micromanagement Tasks[J]. arXiv:1609.02993,2016.

[9] CHURCHILL D,BURO M. Portfolio greedy search and simulation for large-scale combat in starcraft[C]// Computational Intelligence in Games. IEEE,2013:1-8.

[10] JUSTESEN N,RISI S. Continual online evolutionary planning for in-game build order adaptation in StarCraft[C]// the Genetic and Evolutionary Computation Conference. 2017:187-194.

[11] Holland J H. Adaptation in natural and artificial systems[M]. MIT Press,1992.

[12] 沐阿华,周绍磊,于晓丽. 一种快速自适应遗传算法及其仿真研究[J]. 系统仿真学报,2004,16(1):122-125.

[13] 姜昌华. 遗传算法在物流系统优化中的应用研究[D]. 中山:华东师范大学,2007.

[14] SRINIVAS M,PATNAIKL M. Adaptive probabilities of crossover and mutation in genetic algorithms[J]. IEEE Transactions on Systems Man & Cybernetics,2002,24(4):656-667.

[15] KOVARSKY A,BURO M. Heuristic Search Applied to Abstract Combat Games[M]// Advances in Artificial Intelligence. Berlin:Springer,2005:66-78.