中国科学院大学

University of Chinese Academy of Sciences

硕士学位论文

基于深度蒙特卡洛方法的斗地主算法

作者姓名：王成龙

指导教师：孙晓明 研究员

中国科学院计算技术研究所

学位类别：工学硕士

学科专业：计算机软件与理论

培养单位：中国科学院计算技术研究所

2022 年 6 月

Deep Monte Carlo Method for Doudizhu

A thesis submitted to

University of Chinese Academy of Sciences

in partial fulfillment of the requirement

for the degree of

Master of Engineering

in Computer software and theory

By

Wang Chenglong

Supervisor: Professor Sun Xiaoming

Institute of Computing Technoloy, Chinese Academy of Sciences

June, 2022

中国科学院大学
学位论文原创性声明

本人郑重声明：所呈交的学位论文是本人在导师的指导下独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明或致谢。

作者签名：

日　　期：

中国科学院大学
学位论文授权使用声明

本人完全了解并同意遵守中国科学院有关保存和使用学位论文的规定，即中国科学院有权保留送交学位论文的副本，允许该论文被查阅，可以按照学术研究公开原则和保护知识产权的原则公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存、汇编本学位论文。

涉密及延迟公开的学位论文在解密或延迟期后适用本声明。

作者签名：　　导师签名：

日　　期：　　日　　期：

摘 要

机器博弈是现阶段人工智能的核心问题，其研究能够推动人工智能技术由感知智能迈向决策智能。在完美信息博弈方面，以 AlphaGo 和 AlphaGo Zero 为代表的深度强化学习技术在围棋上取得了重大突破，打败了人类顶级玩家。在不完美信息博弈方面，以卡耐基梅隆大学和阿尔伯塔大学为代表的研究机构也在德州扑克的求解上取得了长足的进步。不完美信息博弈是对许多现实问题的抽象和模拟，具有博弈信息私有、游戏环境随机、信息动态变化以及状态空间复杂等特点，这使得研究和解决不完美信息博弈问题存在巨大的挑战。

斗地主游戏作为一种典型的不完美信息博弈，具有合作竞争并存、固定牌型复杂以及决策序列较长等特点，这使得研究和解决斗地主问题存在特有的挑战。近年来，以 DouZero 为代表的斗地主算法采用深度蒙特卡洛方法，在实战中展现了较强的智能水平。但现有的斗地主算法均存在以下三个问题：第一，斗地主玩家的初始手牌随机分发，并且手牌在游戏过程中动态变化，游戏的胜负受运气影响较大，造成算法的性能不稳定。第二，斗地主算法需要大量的对局记录作为训练样本，然而对智能体的随机初始化使得模型在开始阶段产生的样本大多只有稀疏奖励，导致经验池中绝大多数样本没有获得有用的价值信号。在经验池中随机采样会使得网络的训练周期较长，样本的利用率较低。第三，由于地主和农民在游戏中地位不对等，不同角色上的智能水平存在差异，农民的胜率明显高于地主，地主的智能水平较弱。为解决上述问题，本文从网络结构设计、样本采样方法和智能体训练方式三个方面展开研究，提出并实现了以下三个算法：

（一） 为解决斗地主算法性能不稳定的问题，在深度蒙特卡洛方法中采用残差网络作为深度强化学习的 Q 网络。本文基于残差网络模型设计深层的强化学习 Q 网络，既提高了 Q 网络的泛化能力，又可以利用残差网络的优势来避免模型退化。利用基于残差网络的 Q 网络和基于长短时记忆网络的状态网络，可以提取出斗地主游戏中更深层次的局面信息和出牌信息等特征。结合了残差网络的深度蒙特卡洛方法，降低了训练后智能体性能的不稳定性，提高了斗地主算法整体的智能水平，并且以 0.544 的平均胜率打败了当前领先的斗地主算法 DouZero。

（二） 为解决样本利用率低的问题，在深度蒙特卡洛方法中采用随机优先采样作为样本的采样方法。本文基于随机优先采样方法，在随机采样的基础上引入样本优先级，采样时样本被抽取的概率与优先级成正比，模型可以优先使用经验池中 TD-error 较大的样本。同时，由于随机优先采样也具有随机性，所有样本都有机会被采样到，这也增加了样本的多样性。结合了随机优先采样的深度蒙特卡洛方法，显著地减少了模型的收敛时间，提升了模型中样本的利用率。

（三） 为解决地主智能水平较低的问题，在深度蒙特卡洛方法中引入多体学习作为智能体的训练方式。本文基于多体学习算法，差异化地训练不同智能体来应对游戏中的不同角色，以实现农民与地主间的交互学习。通过将斗地主中的玩家分为农民和地主两组，可以构成两组零和博弈并求解纳什均衡，再利用极大极小 Q 算法对地主智能体进行单独训练，地主与农民间交互学习当前状态下的决策任务，实现了多体协同机制。结合了多体学习的深度蒙特卡洛方法，明显地增强了地主的对战能力，提高了地主的智能水平，并且以 0.086 的平均差异得分率打败了当前领先的斗地主算法 DouZero。

关键词： 不完美信息博弈，斗地主，深度蒙特卡洛，残差网络，随机优先采样，多智能体强化学习

Abstract

Computer game is the core problem of artificial intelligence research at this stage. Computer game can promote AI from cognitive intelligence to decision intelligence. In the perfect information game, Deep Reinforcement Learning techniques such as AlphaGo and AlphaGo Zero have made major breakthroughs in Go, defeating top human players. In the imperfect information game, CMU and the University of Alberta also made strides in solving Texas Hold'em. The Imperfect Information Game is an abstraction and simulation of many real-world problems that have the characteristics of private game information, a random game environment, dynamic information changes, and a complex state space, making exploration and solution a challenge.

Doudizhu is a typical game with imperfect information and is characterized by the coexistence of cooperative competition, complex fixed pieces, and long decision sequences. These problems makes its research a unique challenge. In recent years, the Doudizhu algorithm represented by DouZero has adopted the Deep Monte Carlo method and demonstrated a high level of intelligence in the real world. However, the existing Doudizhu algorithms have the following three problems. First, the initial cards are distributed randomly, and the winning of the game is affected by luck, making the performance of the agent unstable. Second, the deep Monte Carlo method requires a large number of games records as samples. However, the random initialization of the agent causes the model to generate mainly sparse rewards in the initial phase, resulting in the vast majority of samples in the experience pool not receiving useful value signals. The random selection of samples in the experience pool leads to low sample utilization. Third, due to the unequal status of landowners and farmers in the game, there are differences in the intelligence level of the different agents, and the winning Percentage and of Peasants are significantly higher than those of landlord, indicating that the intelligence level of landlord is weaker. In order to solve the above problems, three aspects are studied in this paper: the design of the network structure, the sampling methods and the intelligence training methods.

1. For unstable performance of DouZero, we use a residual network for reinforcement learning's Q-network. In order to improve the generalization ability of the network, the residual network model is introduced to design Q-network for reinforcement learning. Using the residual network and the long- and short term memory network as state network, we extracted the bucket features such as deeper situation information and history records in Doudizhu. At the same time, we combined the deep Monte Carlo method of residual networks to reduce the variance of the agent's winning percentage and the average difference in point after training. That improved the overall intelligence of the Doudizhu algorithm and beat the current leading DouZero algorithm with an average winning fraction of 0.544.

2. For low sample utilization, the prioritized experience replay is introduced to deep Monte Carlo method.In this paper, the sample priority is defined by TD-error, and the probability of sampling at the time of sampling is proportional to the priority, so that the model can preferentially use the samples with the larger TD-error in the experience pool. Since prioritized experience replay is also randomized, all samples have a chance of being captured, and sample diversity is also increased. By combining the deep Monte Carlo method with prioritized experience replay, we can significantly reduce the convergence time of the model and improve the use of samples in the model.

3. For landlords' low intelligence problem, multi-body learning algorithm is introduced as a training method for the agents. In this work, based on the multi-agent reinforcement learning algorithm, different agents are trained differently to cope with different roles in the game, and the multi-body cooperation is achieved by interactive learning of the decision task. By dividing the players into two groups: Peasants and landlords, forming two-team zero-sum game and solving the Nash equilibrium, the landlord is then trained individually with the Minmax Q-learning. We combine a Deep Monte Carlo approach with multi-agent reinforcement learning to significantly improve the landlords' winning percentage and increase their intelligence.

Keywords: Imperfect Information Game, Doudizhu, Deep Monte Carlo, Residual Network, Prioritized Experience Replay, Multi - agent reinforcement learning

目 录

第 1 章 引言 ····· 1
1.1 研究背景与意义 ····· 1
1.1.1 研究背景 ····· 1
1.1.2 研究意义 ····· 4
1.2 研究现状 ····· 5
1.2.1 机器博弈的国外研究现状 ····· 5
1.2.2 机器博弈的国内研究现状 ····· 6
1.2.3 斗地主算法的研究现状 ····· 7
1.3 主要研究内容 ····· 9
1.4 主要研究贡献 ····· 10
1.5 论文组织结构 ····· 12
第 2 章 相关基础知识 ····· 13
2.1 斗地主游戏规则 ····· 13
2.2 机器博弈的经典算法 ····· 15
2.2.1 博弈树搜索 ····· 16
2.2.2 遗憾值最小化算法 ····· 16
2.2.3 专家系统 ····· 17
2.3 深度强化学习发展历程 ····· 18
2.3.1 深度学习 ····· 18
2.3.2 强化学习 ····· 22
2.3.3 深度强化学习 ····· 24
2.4 本章小结 ····· 26
第 3 章 结合残差网络的深度蒙特卡洛方法 ····· 27
3.1 深度蒙特卡洛方法 ····· 27
3.2 基于残差网络的斗地主算法 ····· 30
3.2.1 斗地主算法框架 ····· 30
3.2.2 多演员并行算法 ····· 35
3.2.3 基于规则的残局策略 ····· 37
3.3 实验结果与分析 ····· 38
3.3.1 实验环境设置 ····· 38
3.3.2 对战实验结果与分析 ····· 43
3.4 本章小结 ····· 55

第 4 章 结合随机优先采样的深度蒙特卡洛方法 ····· 57
4.1 随机优先采样算法 ····· 57
4.2 基于随机优先采样的斗地主算法 ····· 59
4.3 实验结果与分析 ····· 63
4.3.1 实验环境设置 ····· 63
4.3.2 对战实验结果与分析 ····· 64
4.4 本章小结 ····· 72
第 5 章 结合多体学习的深度蒙特卡洛方法 ····· 73
5.1 多智能体强化学习 ····· 73
5.2 基于多体学习的斗地主算法 ····· 76
5.3 实验结果与分析 ····· 78
5.3.1 实验环境设置 ····· 78
5.3.2 对战实验结果与分析 ····· 79
5.4 本章小结 ····· 85
第 6 章 总结与展望 ····· 87
6.1 研究总结 ····· 87
6.2 未来展望 ····· 89
参考文献 ····· 91
致谢 ····· 95
作者简历及攻读学位期间发表的学术论文与研究成果 ····· 97

图形列表

1.1 研究内容关系图 ····· 11

2.1 一个神经网络结构的例子 (Goodfellow 等，2016) ····· 19
2.2 卷积神经网络结构图 (Goodfellow 等，2016) ····· 20
2.3 循环神经网络结构图 (Goodfellow 等，2016) ····· 20
2.4 残差神经网络结构图 (He 等，2016) ····· 21
2.5 强化学习过程 ····· 23

3.1 利用马尔科夫链表示的对局历史记录 ····· 28
3.2 手牌和动作编码示例。(a) 单张牌 (b) 对牌 (c) 三顺牌 (d) 单链 (e) 火箭 (f) 某种手牌编码的示例。 ····· 31
3.3 网络整体架构图 ····· 33
3.4 状态网络架构图 ····· 34
3.5 残差块结构图 ····· 35
3.6 残局策略流程图 ····· 38
3.7 RLCard 结构图 ····· 40
3.8 RLCard 中斗地主的对战流程图 ····· 42
3.9 DMC-ResNet 地主对战随机算法农民的胜率 ····· 44
3.10 DMC-ResNet 农民对战随机算法地主的胜率 ····· 44
3.11 DMC-ResNet 地主对战随机算法农民的 ADP ····· 45
3.12 DMC-ResNet 农民对战随机算法地主的 ADP ····· 45
3.13 DMC-ResNet 地主对战 RHCP 算法农民的胜率 ····· 47
3.14 DMC-ResNet 农民对战 RHCP 算法地主的胜率 ····· 47
3.15 DMC-ResNet 地主对战 RHCP 算法农民的 ADP ····· 48
3.16 DMC-ResNet 农民对战 RHCP 算法地主的 ADP ····· 48
3.17 DMC-ResNet 地主对战 CQL 算法农民的胜率 ····· 49
3.18 DMC-ResNet 农民对战 CQL 算法地主的胜率 ····· 49
3.19 DMC-ResNet 地主对战 CQL 算法农民的 ADP ····· 50
3.20 DMC-ResNet 农民对战 CQL 算法地主的 ADP ····· 50

4.1 随机优先采样的流程图 ····· 59
4.2 PER 在深度蒙特卡洛上的流程图 ····· 60
4.3 对战随机算法的胜率比较 ····· 64

4.4 对战随机算法的 ADP 比较 ····· 65
4.5 对战 RHCP 算法的胜率比较 ····· 66
4.6 对战 RHCP 算法的 ADP 比较 ····· 66
4.7 对战 CQL 算法的胜率比较 ····· 67
4.8 对战 CQL 算法的 ADP 比较 ····· 68
4.9 对战 DouZero 算法的胜率比较 ····· 68
4.10 对战 DouZero 算法的 ADP 比较 ····· 69

5.1 对战随机算法的地主胜率比较 ····· 79
5.2 对战随机算法的地主 ADP 比较 ····· 79
5.3 对战 RHCP 算法的地主胜率比较 ····· 80
5.4 对战 RHCP 算法的地主 ADP 比较 ····· 81
5.5 对战 CQL 算法的地主胜率比较 ····· 81
5.6 对战 CQL 算法的地主 ADP 比较 ····· 82
5.7 对战 DouZero 算法的地主胜率比较 ····· 82
5.8 对战 DouZero 算法的地主 ADP 比较 ····· 83

表格列表

2.1 手牌牌型 ····· 15

3.1 地主角色编码的特征 ····· 32
3.2 农民角色编码的特征 ····· 33
3.3 斗地主状态类型 ····· 41
3.4 斗地主动作类型 ····· 42
3.5 与 DouZero 比较胜率 ····· 52
3.6 与 DouZero 比较 ADP ····· 52
3.7 与 DouZero 比较胜率的方差 ····· 53
3.8 与 DouZero 比较 ADP 的方差 ····· 53
3.9 不同数量演员的胜率比较 ····· 55
3.10 不同数量演员的 ADP 比较 ····· 55

4.1 收敛时间与算法性能比较 ····· 71

5.1 多体学习下地主胜率的比较 ····· 84
5.2 多体学习下地主 ADP 的比较 ····· 84
5.3 地主的胜率比较 ····· 84
5.4 地主的 ADP 比较 ····· 85

符号列表

符号

Symbol	Description
s_t	State of time t
a_t	Action of time t
r_t	Reward of time t
Q_t	Q-value of time t

缩写

AI	Artificial Intelligence
GPS	General Problem Solver
CQL	Combinational Q-Learning
DMC	Deep Monte Carlo
DNN	Deep Neural Network
CNN	Convolutional Neural Network
RNN	Deep Residual Network
ResNet	Residual Network
RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
DQN	Deep Q-learning
WP	Winning Percentage
ADP	Average Difference in Points
PER	Prioritized Experience Replay
DPER	Distributed Prioritized Experience Replay
MAS	Multi - agent reinforcement learning
NashQ	Nash Q-learning

MinmaxQ	Minmax Q-learning
LSTM	Long short-term memory
MDP	Markov Decision Process

第 1 章　引言

1.1　研究背景与意义

1.1.1　研究背景

机器博弈也叫计算机博弈，是目前人工智能（Artificial Intelligence，AI）领域的一个重要研究方向。通过对机器博弈的研究，可以帮助人工智能技术从简单的感知智能发展为决策智能，同时也可以帮助实现人与机器的智慧交互。机器博弈通过机器模拟人类的决策行为，按照不同游戏环境下的博弈规则，训练得到具有一定决策能力的智能体。

1.1.1.1　人工智能发展背景

1956 年，John McCarthy 等人在达特茅斯会议上首次提出“人工智能”这一概念，人工智能这一学科也由此诞生 (Buchanan, 2005; Kaplan 等, 2019)。在人工智能技术发展的早期，研究人员通过搜索和专家系统等不同的研究方法取得了许多突出的成果。在搜索式推理方面，艾伦 · 纽厄尔和赫伯特 · 西蒙通过“通用解题器（General Problem Solver，GPS）”程序解决基本的二维“迷宫”游戏，并将该方法推广到一般性的迷宫问题上 (Berlinski, 2001)。此外，研究人员还利用搜索算法证明了几何与代数中的一些基本问题，例如赫伯特 · 吉宁特开发的几何定理证明机和马文 · 李 · 闵斯基开发的 SAINT 代数定理证明机 (Buchanan, 2005)。在自然语言处理方面，Joseph Weizenbaum 发明了世界上第一个聊天机器人 ELIZA(Winston, 1992)。上世纪 70 年代后，“专家系统”算法开始登上历史舞台。一些商业公司也开始利用“专家系统”来处理日常问题。“专家系统”可以通过人类专家经验知识产生逻辑规则程序，并以此来解决许多现实问题 (Berlinski, 2001)。1982 年，物理学家 John Hopfield 提出一种新型的神经网络（现被称为“Hopfield 网络”），该网络会用一种全新的方式学习和处理信息，随后 David Rumelhart 在此基础上推广了反向传播算法，使联结主义逐渐成为研究的主流 (McCorduck 等, 1977, 2004)。

进入 21 世纪后，随着计算能力的快速提升，深度学习技术的提出成功解决了许多现实问题。神经网络专家 Hinton 提出深度神经网络的反向传播算法，从而使得神经网络的能力大大提高，深度学习在学术界和工业界开始展露头角 (LeCun 等, 1998)。Yann LeCun 和吴恩达等 (LeCun 等, 2015) 提出了卷积神经网

络，该方法使得网络在训练时能够快速收敛。并且卷积神经网络所占用的内存较小，可以扩展出深层的神经网络，非常适合处理图像的识别问题。在 2012 年的 ImageNet 视觉识别比赛中，多伦多大学研究人员所设计的深度卷积神经网络首次超过了人类识别的成功率 (Krizhevsky 等, 2012)。之后，结合了深度学习和强化学习的深度强化学习技术在机器博弈上取得了突出的成绩。在 2016 年 3 月，谷歌公司开发设计的 AlphaGo 算法击败韩国棋手李世石，成为第一个击败职业围棋运动员的智能算法 (Chen, 2016)。随后在 2017 年 5 月，DeepMind 公司开发了 AlphaGo Zero 系统，该系统通过自博弈的方式训练学习，最终成功击败了当时世界排名第一的棋手柯洁 (Zhang 等, 2020)。

1.1.1.2 机器博弈发展背景

对于不同的博弈游戏，可以借助不同的分类标准来进行划分：

(一) 基于玩家的数量，可以分为两人博弈与多人博弈。在两人博弈里，每一个玩家都只需要考虑对手的决策策略，并以此找出相应的最优应对策略。多人博弈一般指游戏的玩家人数大于两人，此时任何一个玩家需要考虑除自身以外的其他所有玩家的决策策略，而且在博弈的过程中还会存在这些对手策略的相互作用。

(二) 基于游戏玩家间是否可以合作可分为合作性博弈和对抗性博弈。在对抗性博弈游戏中，所有玩家都不能合作，每一个玩家只需要考虑将自己的奖励最大化。而在合作性博弈里，在开始博弈之前数位玩家需要形成某种具体的协议，比如博弈奖励的分配方式、玩家出牌动作的标记约定等。在合作博弈中，玩家在做决策时，需要考虑其他有合作玩家的决策动作，使得合作的整体奖励最大化。比如在斗地主游戏中，两个农民会约定相同的奖励，需要共同对抗一个地主玩家。但在德州扑克中，所有玩家的目标都是最大化自身奖励，是对抗性博弈。

(三) 基于在游戏过程中是否可以获得所有的信息，可以将博弈分为完美信息博弈和不完美信息博弈。在完美信息博弈游戏中，所有玩家在所有状态下都可以获得所有的信息，不存在某些玩家的私有博弈信息，比如围棋和中国象棋等。如果不同玩家所获得的博弈信息不对等，存在某些私有的博弈信息使得不是所有玩家都可以获得所有的博弈信息，这类博弈游戏称为不完美信息博弈。比如德州扑克、斗地主游戏、星际争霸和王者荣耀等。因为在不完美信息博弈中不能获得游戏中所有的博弈信息和状态，提升了机器博弈的研究难度。

对于状态空间较小的完美信息博弈游戏，研究人员通过建立价值评价函数和搜索生成博弈树寻找最优解。基于纳什均衡 (Nash Jr, 1950) 和搜索算法，研究人员提出了许多算法博弈论技术，如针对状态价值的极大极小搜索算法 (Thomas H 等, 2016)、对动作价值负极大值的搜索算法 (Bench-Capon 等, 2007)、并行剪枝的搜索算法 (Takeuchi 等, 2010) 等。1992 年，美国的塞缪开发出了第一个能够战胜开发者的跳棋计算机程序，并随后战胜了人类顶级玩家。1997 年，美国的 IBM 公司所开发的“深蓝-思考 IBM”系统在和人类顶级专家进行的国际象棋比赛中获胜 (Bench-Capon 等, 2007)。这说明使用博弈树搜索算法的机器博弈系统，在象棋、跳棋等这类简单的完美信息博弈游戏上的智能水平已经超过了顶尖人类专家。

但是对于状态空间较大的完美信息博弈游戏上，简单的博弈树搜索无法遍历所有的可能状态。所以对于围棋这类复杂的完美信息博弈游戏，谷歌公司基于深度强化学习方法，开发出了 AlphaGo 和 AlphaGo-Zero 系统，并且这两个系统分别在与李世石、柯洁的围棋对局中取得了胜利 (Chen, 2016; Zhang 等, 2020)。这表明使用深度强化学习算法的机器博弈系统，在如围棋这种复杂的完美信息博弈游戏领域上的智能水平也已经接近和超过了人类顶尖玩家。

相较于完美信息博弈游戏，不完美信息博弈游戏由于存在隐藏的私有信息，使得游戏的状态空间复杂，游戏的求解复杂度也非常大。许多的不完美信息博弈游戏的纳什均衡解被证明为 NP 困难问题 (Zinkevich 等, 2007)。2017 年在美国匹斯堡，来自美国卡耐基梅隆大学的博士生 Noam Brown 所开发的的德州扑克博弈系统冷扑大师（Libratus）在与四名人类顶尖德州扑克选手之间进行了多局对战，并且获得了胜利 (Brown 等, 2018)。随后，在多人无限注德州扑克领域，Noam Brown 所设计的智能算法也取得了突出的胜绩 (Brown 等, 2019)。这宣告在德州扑克这种大规模的非完美信息机器博弈游戏中，机器博弈的智能水平已经超过了人类顶尖玩家。

一直以来，起源于中国的斗地主游戏都被视为一个极具影响力的非完美信息博弈游戏，在中国和世界范围内获得了许多人的喜爱和参与。斗地主游戏分为两个农民角色和一个地主角色，在游戏中两个农民角色与地主角色之间存在着零和博弈的对抗性，但是这两个农民角色之间也存在着合作性。并且斗地主游戏是非完美信息博弈，每一个玩家不能看到其他玩家的手牌和全局的所有信息。

而且游戏中包含随机手牌的“运气”成分，即游戏中玩家的胜负受随机所发的手牌质量的影响较大。因此，斗地主有非常复杂的博弈树以及非常大的状态空间。

目前学术界针对斗地主游戏开发的博弈算法较少，部分基础算法依然采用基于人类经验的专家系统 (彭文, 2020; 沈恒恒, 2021)。近期，随着深度强化学习等技术的发展，研究人员提出了 DeltaDou(Jiang 等, 2019)、CQL(You 等, 2020) 和 DouZero(Zha 等, 2021) 等斗地主智能系统。但是相较于其他机器博弈游戏的发展成果而言，斗地主算法的智能水平还有巨大的提升空间。

1.1.2 研究意义

相较于完美信息博弈游戏，不完美信息博弈游戏，如德州扑克、斗地主等游戏的研究难度更大。玩家不能获得当前对局状态下的全部信息，每个玩家都有其私有的信息。尤其在多人游戏中，对手玩家的策略还存在不稳定的情况，这也增加了不完美信息博弈研究的难度。不完美信息博弈的特点主要可以归纳为：

（一） **博弈信息的私有性**。在非完美信息博弈游戏中，玩家只能获得部分信息，每一个玩家都具有单独属于自己的私有信息。例如在斗地主游戏中，每一个玩家都只能获得自身的手牌信息和历史的出牌记录，无法获得对手的手牌状态。并且不对称的玩家信息使得智能体不能根据已知信息来计算出最优策略，这就使得智能体需要依据自身的历史信息进行决策，这也是非完美信息博弈游戏无法直接使用完美信息博弈游戏决策方案的主要原因。

（二） **游戏环境的随机性**。斗地主、德州扑克等非完美信息博弈游戏相比于围棋、象棋等完美信息博弈游戏还有游戏中增加的随机性，博弈中的玩家手牌是随机发放的。在对局中，玩家需要根据自身随机获得的手牌，利用有限的信息做出使自身损失尽可能小的决策，这就使得游戏中玩家的胜负关系会受随机所发的手牌质量的影响。

（三） **信息动态变化**。在非完美信息博弈游戏中，每一位玩家的获得信息是动态变化的。比如在斗地主中，随着玩家手牌的减少，游戏中的公开信息会逐渐增加，例如历史对局记录，但是每个玩家自身的私有信息会逐渐减少，例如手牌数量。这就使得每一个玩家都需要随时更新自身的状态信息。

（四） **状态空间复杂**。相比于简单的两人零和博弈，麻将、斗地主和多人无限注德州扑克游戏状态空间和决策空间的规模更大，也更具有广泛的现实意义。不完美信息博弈逐渐因为其较高复杂性而成为当前的研究热点。

不完美信息博弈是对许多现实问题的模拟，对其的研究具有实际应用价值，有利于未来面向现实生活的智能决策系统研究，同时也为人机交互等其他人工智能研究建立基础。不完美信息博弈可以为解决实际问题提供有效的方法指导，其研究具有重要的理论价值与应用前景。

“斗地主”作为风靡全国的扑克博弈游戏，具有上述不完美信息博弈游戏的研究难点。该游戏的博弈特点突出，游戏环境十分适用于不完美信息博弈的研究和算法验证。除此之外，斗地主游戏还有三个独特的挑战：

（一）**合作竞争并存：**在斗地主中，农民玩家和地主玩家整体上形成零和博弈，但是两个农民角色要相互配合对抗地主，这使得游戏中的玩家不仅要考虑对抗竞争，还需要注意农民角色间的合作。

（二）**固定牌型复杂：**相比于德州扑克简单的牌型组合和动作空间，斗地主游戏中具有复杂的牌型结构，例如单张、三带一、对子、炸弹等等，这使得动作集规模更大。

（三）**决策序列较长：**相比于德州扑克可以快速得到胜负关系完成一次对局，斗地主游戏过程中的决序列较长，这带来了决策过程中的稀疏奖励问题。

斗地主游戏还具有不完备性、快速决策、重复博弈等特点，集中了不完美信息博弈中的许多核心问题。此外，游戏中还存在农民之间如何协调合作的问题，这使得斗地主的研究对于整个人工智能领域有着极其重要的意义。作为典型的非完美信息博弈游戏，斗地主研究不仅在理论和现实应用上都具有重要的意义，而且因为其合作竞争并存的游戏规则，使得智能体所面临的困难相比于德州扑克等其他游戏更符合现实环境。

1.2　研究现状

1.2.1　机器博弈的国外研究现状

相较于完美信息博弈，不完美信息博弈游戏更符合现实的生活环境。因为不完美信息博弈的局面信息不全面，其状态空间和动作空间也更加复杂。国外的不完美信息博弈研究主要集中在桥牌和德州扑克游戏上。

在利用深度学习技术之前，不完美信息博弈的相关研究工作取得了一定的成绩。Bampton 在 1994 年引入蒙特卡洛算法 (Bampton, 1994)。通过在每一次蒙特卡洛采样时，将不完美信息博弈简化设置为完美信息博弈，进而利用已有的价值

搜索方法获得每一次采样的最优决策。在 1988 年的国际桥牌锦标赛中，Matthew Ginsberg 利用蒙特卡洛采样方法开发的智能体 GIB，以显著优势获得了最终的冠军 (Ginsberg, 1988)。这标志着人工智能在桥牌这类简单的非完美信息博弈游戏上超过了人类顶尖玩家。

在德州扑克这类大规模的非完美信息博弈游戏上，Dahl 等人引入强化学习算法 Q leaning，并在两人有限注德州扑克游戏中取得一定的成绩 (Dahl, 2001)。2003 年，Billings 等 (2003) 等人提出了在两人有限注德州扑克游戏上的理论最优解。随后在 2007 年的世界扑克锦标赛上，基于理论解和遗憾最小化算法（Regret Matching，RM）所开发的智能体 Polaris 首次击败人类玩家。2008 年，Zinkevich 等人提出了使用虚拟遗憾最小化算法（Counterfactual Regret Minimization，CFR）来求解近似纳什均衡，这种算法也在两人的无限注德州扑克游戏中取得了不错的成绩 (Zinkevich 等, 2007)。在 2009 年的世界扑克锦标赛中，Zinkevich 团队基于 CFR 算法设计的智能体最终获得了冠军。但在多人不完美信息博弈中，CFR 方法计算复杂度巨大，并且如果任意一个玩家的决策不按照纳什均衡策略，CFR 算法都将无法收敛，这使得 CFR 算法在多人德州扑克上难以获得理想的结果。2015 年，加拿大阿尔伯塔大学的科研团队在双人无限注德州扑克上开发出了 DeepStack 系统，并且该智能体首次打败了职业玩家 (Moravčík 等, 2017)。同年，美国卡耐基梅隆大学的博士生 Noam Brown 等人所开发的的德州扑克博弈系统冷扑大师（Libratus）在两人德州扑克游戏中，分别与四名人类顶尖德州扑克选手之间进行了多局对战，并获得了胜利 (Brown 等, 2018)。随后该课题组将上述方法扩展到多人德州扑克游戏并开发了 Pluribus 系统 (Brown 等, 2019)，该系统在之后的 ACPC 六人德州扑克比赛中战胜了人类玩家。

1.2.2 机器博弈的国内研究现状

相比于国外在机器博弈的许多研究成就，国内的机器博弈研究成果较少。这主要是因为国内的人工智能研究起步较晚。中国人工智能学会在 2006 年首次在沈阳举办了中国机器博弈锦标赛，这项赛事吸引了许多国内研究人员对于机器博弈研究的关注，也为国内机器博弈研究领域起到了积极的推广作用。

在完美信息博弈游戏上，国内的研究者主要研究的游戏包括六子棋和中国象棋等。在六子棋上，东北大学的徐长明团队引入了局面表示方法 (徐长明, 2010)。重庆理工大学的张小川团队首次提出基于遗传算法的评估函数，该方法达到了

人类玩家的智能水平 (张小川 等, 2010)。在中国象棋游戏上，东北大学的王骄团队基于象棋开局、循环排定规则和评价函数进行了研究，并在中国机器博弈竞赛中取得理想的成绩 (王骄 等, 2005, 2012)。

在非完美信息博弈研究领域，国内的研究者主要研究的游戏包括四国军棋、德州扑克。针对四国军棋游戏，哈尔滨工业大学的王轩团队基于时序差分、强化学习等及其改进算法，集中研究了两人机器博弈问题 (马骁 等, 2010; 王骄 等, 2012)。针对德州扑克游戏，该团队基于对手建模和状态评价函数结合的方法，在 ACPC 的二人有限制德州扑克游戏比赛中获得了第四名，这也是目前国内在 ACPC 上的最好成绩 (马骁 等, 2010)。

1.2.3　斗地主算法的研究现状

CFR 算法一般应用于解决两人零和博弈问题，每位玩家都需要按照纳什均衡策略做出动作选择，所以研究者发现 CFR 算法在斗地主游戏中并未取得显著效果 (Jiang 等, 2019)。针对斗地主游戏，Jiang 等 (2019) 提出的 DeltaDou 是第一个可以和真人斗地主玩家媲美的人工智能算法。针对不完美信息博弈游戏，DeltaDou 系统结合了寻找纳什均衡和长期收益最大化两种思想，提出了虚拟博弈蒙特卡洛搜索树 FPMCTS（Fictitious Play MCTS）。但是 DeltaDou 计算成本很高，即使在开始时使用人类专家的成功经验来对模型进行监督训练，也需要两个多月的时间来完成模型的收敛。

随着深度强化学习技术的发展，上海交通大学的研究人员在 2019 年提出了基于手牌拆分和组合强化学习 (Combinational Q-Learning，CQL) 的斗地主算法 (You 等, 2020)。根据模仿人类玩家的决策思路，将动作网络拆分为二个策略决策网络。组合建议网络负责对手牌中可能存在的牌组进行拆分，动作建议网络负责决定具体的出牌动作。其中组合建议网络采用了 pointNET 网络，结合了多层感知神经网络和最大池化层，使得提取的手牌特征保持线性不变性，在每一个牌组上保证了测度集的稳定。动作建议网络利用组合建议网络提取出的个体特征和整体特征进行拼接，从而保证了动作建议网络不仅可以考虑单次出牌的奖励，还可以很好的利用前后出牌策略的连贯性。最终该智能体在与基于规则的 RHCP 对战中取得较好的效果，稳定战胜随机策略和以深度 Q-learning 为基础的智能体。但是这种基于人类玩家决策思路的算法，分割了手牌组合和出牌决策的价值状态函数，使得学习训练时间过长。同时因为 CQL 算法是启发式的，长序

列决策带来的稀疏奖励问题并没有很好的解决。算法不能很好的保证策略收敛，网络的训练难度大，模型的泛化能力也弱。

2021 年，由快手 AI 平台部的研究者提出的的 DouZero 斗地主算法保持着目前较为领先的智能性和实战水平 (Zha 等, 2021)。DouZero 算法利用深度蒙特卡洛方法，是目前最好的斗地主算法。强化学习的目标是针对一个特定的任务去学习一个策略来最大化奖励。强化学习中状态是指一个策略在某一时刻所能看到所有信息。在斗地主中，状态包括玩家当前的手牌、过去的出牌历史。动作是指在某一个状态下做出的一个行为。在斗地主中，动作指玩家打出的牌型。DouZero 方法基于深度蒙特卡洛模型 (Deep Monte Carlo，DMC)、 0 和 1 的动作编码以及多 Actor 并行计算等方法，提出了一个目前领先的斗地主算法。算法的主要思想是，利用动作价值网络对不同状态下的动作进行价值评估，网络的输入是当前状态和一个具体的动作，网络的输出是在当前状态做该动作的期望价值。在最终决策时，价值网络选择动作价值最大的牌型作为当前状态下的最优动作。DouZero 算法具有以下几个优点：

（一） 利用 0 和 1 对动作进行编码，使得相近牌型的动作集在编码后的特征结构上相似，在神经网络训练时更容易发现牌型的关联性；

（二） 深度蒙特卡洛方法不会出现对动作价值的过大评估，这是因为模型在训练时将每一个完整对局的结果作为训练监督，这深度蒙特卡洛方法适用于斗地主这种长序列决策游戏；

（三） 利用训练 Learner 和对战 Actor 两个模块分离的方法，DouZero 可以快速产生大量的自博弈对局记录，并将这些作为样本提供给 Learner 用于训练；

虽然基于深度蒙特卡洛方法的斗地主算法对动作和状态价值进行了有效的评估，但是现有的斗地主算法均存在以下三个问题：

（一）性能不稳定： 在其与随机出牌、CQN 算法和 RHCP 算法等的实际对战中，深度蒙特卡洛模型的胜绩不够稳定，对战胜率受玩家初始手牌的影响较大。这说明模型的泛化能力较差，不能很好的应对“运气”所带来的影响，算法的平均胜率和平均差异等分率的方差较大。

（二）样本利用率低： 虽然模型的训练时间与 CQL 等算法相比已经有了极大的减少，但是在实际中样本的利用率低，训练周期较长，模型学习的时间在一个月左右。算法在采样样本时随机选择经验池中的样本，而样本池中的样本大

多为稀疏奖励，很多样本的实际信息含量低，这使得模型在训练中的样本利用效率较低。

（三）地主智能水平较低： 虽然深度蒙特卡洛模型可以分别为三个玩家训练了不同的智能体，但斗地主游戏本身就具有角色的不对等性，地主相比农民需要更保守的策略，这使得 DouZero 算法在实际对战中地主角色的智能水平相较于农民较低。

1.3 主要研究内容

针对现有斗地主算法所出现的不足，本文将从网络结构设计、样本采样方法、智能体训练方式等方面展开研究，提出并实现了以下三个算法：

（一）结合残差网络的深度蒙特卡洛方法：

针对算法性能不稳定的问题，在深度蒙特卡洛方法中采用残差网络作为深度强化学习中的 Q 网络。在斗地主游戏中，玩家的初始手牌随机分发，并且玩家的手牌存在动态变化，游戏的胜负受运气影响较大。而如果直接通过增加网络的深度来提高模型的泛化能力，会导致梯度消失和爆炸，并且网络在训练中会出现退化问题，即随着网络层数的增加，在训练集上的准确率反而会减少。这说明虽然深层网络在理论上泛化能力强，但是实际效果很难训练到全局最优解。

为了保留深层网络的深度，同时又可以利用浅层网络的优势来避免退化，引入残差网络作为 Q 网络。在斗地主游戏中，采用结合残差网络的深度蒙特卡洛方法，减少初始时的运气和过程中手牌变化所带来的影响。通过对深度强化学习的 Q 网络采用残差网络模型，设计更深层的网络架构，可以提升模型的泛化能力，降低了斗地主算法性能的不稳定性。

（二）结合随机优先采样的深度蒙特卡洛方法：

针对样本利用率低的问题，在深度蒙特卡洛方法中引入随机优先采样作为采样方法。稀疏奖励是深度强化学习在解决实际任务中经常出现的问题。在强化学习中，动作序列在不同状态下获得的奖励是训练时的信号，智能体可以根据奖励进行策略优化。但因为在开始训练时，算法需要对策略进行随机初始化，而最终结果需要经过较长的动作序列。这使得初始时产生稀疏奖励的样本，导致经验池中大多数样本没有获得有用的价值信号，因此深度强化学习的训练较为困难。稀疏奖励问题还会导致强化学习算法迭代缓慢，甚至策略最终无法收敛。

在斗地主游戏中，通过结合随机优先采样的深度蒙特卡洛方法，可以在随机性的基础上引入样本优先级的概念。智能体与环境交互所产生的样本会存储在样本池中。在深度 Q 网络训练时，深度蒙特卡洛模型可以对样本池进行采样，样本被抽取的概率与优先级成正比，优先使用经验池中具有较大 TD-error 的样本。该方法之所以能有效，是由于某些经验数据相较于其他经验数据，可能包含更多值得学习的信息，所以给予这些包含更丰富信息量的经验更多的回放机会，有助于使得整个学习进度更为快速和高效。同时，由于优先经验回放法在采样时也具有随机性，所有样本都有机会被采样，这也增加了样本的多样性，可以提升训练中样本的使用效率。

（三）结合多体学习的深度蒙特卡洛方法：

针对地主智能水平较低的问题，在深度蒙特卡洛方法中引入多体学习方法。由于地主和农民在游戏中的地位不对等，基于深度蒙特卡洛方法的斗地主算法在不同角色上的智能水平存在差异，农民角色的胜率明显高于地主。

在斗地主游戏中，通过结合多体智能学习的深度蒙特卡洛方法，可以针对性地提升地主的对战水平。设置上述实验中训练的智能体做为地主和农民的预训练模型，采用多体学习算法，差异化地训练不同智能体来应对游戏角色的地位不对等问题，以实现农民与地主间的交互学习。因为主要是针对地主角色来提升胜率，实验中设置已有的农民智能体为固定模型，并将三位玩家分为两组：农民和地主，由此构成了两组零和博弈。在训练中，利用极大极小 Q 算法对已有的地主智能体进行训练，以提高地主角色的智能水平。

1.4 主要研究贡献

对于现有斗地主算法，本文可以针对性地改进性能不稳定、样本利用率低、地主智能水平较低等问题，具体的研究内容关系如图1.1所示，每个研究点具体的研究贡献如下：

研究点一： 本文引入残差网络来设计深层的 Q 网络，既提高了 Q 网络的泛化能力，又可以利用残差网络的优势来避免退化。在利用多 Actor 并行计算产生大量历史对局的条件下，利用基于残差网络的 Q 网络可以提取出更多层次的特征信息，提升模型的鲁棒性和泛化能力。结合了残差网络的深度蒙特卡洛方法，降低了训练后智能体性能的不稳定性，并且以平均胜率为 0.544 的成绩打败

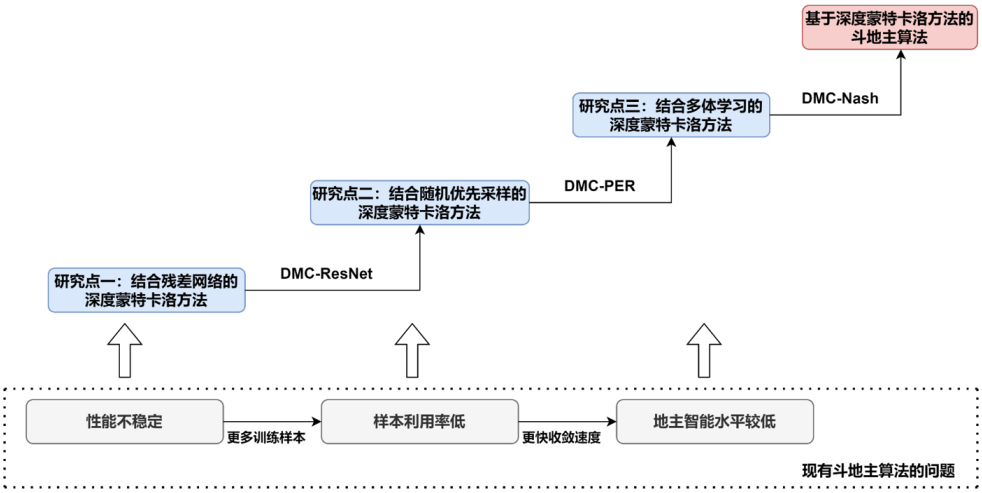

图 1.1 研究内容关系图

Figure 1.1 Relation chart of research parts

了当前最好的 DouZero 算法。

研究点二： 在研究点一中训练智能体 DMC-ResNet 时，需要更多的对战历史记录作为训练样本，这使得模型的训练周期较长，现有的随机采样方法使得样本的利用率低。本文基于随机优先采样方法，在随机采样的基础上引入样本优先级，采样时样本被抽取的概率与优先级成正比，使得模型可以优先使用经验池中较大 TD-error 的样本。同时，由于随机优先采样也具有随机性，所有样本都有机会被采样到，这也增加了样本的多样性。结合了随机优先采样的深度蒙特卡洛方法，显著地减少了模型的收敛时间，提升了模型中样本的利用率。

研究点三： 在研究点一和二中，训练地主和农民智能体时都是采用单体学习的训练方式，这使得算法无法应对游戏角色的不对等性。本文基于多体学习算法，差异化地训练不同智能体来应对游戏中的不同角色，以实现农民与地主间的交互学习。通过将斗地主中的玩家分为农民和地主两组，可以构成两组零和博弈并求解纳什均衡，再利用极大极小 Q 算法对地主智能体进行单独训练，地主与农民间交互学习当前状态下的决策任务，实现了多体协同机制。结合了多体学习的深度蒙特卡洛方法，明显地增强了地主的对战胜绩，提高了地主的智能水平，整体算法以 0.086 的平均差异得分率打败目前最好的 DouZero 算法。

本文针对性地改进了基于深度蒙特卡洛方法的斗地主算法，提升了模型的泛化能力和训练样本的利用效率，并对游戏中的地主和农民给出了差异化的策略，做出了具有创新性的研究工作。

1.5 论文组织结构

本文主要分为六章，论文的具体结构如下：

第一章为论文的引言部分。首先介绍了机器博弈的研究背景，说明了斗地主这类非完美信息博弈游戏的研究意义。然后分别对国内外的机器博弈研究现状进行了介绍，其中重点介绍了现有的斗地主智能算法，分析了现有斗地主算法的优缺点，并以此引出本文的主要研究内容及贡献。最后简要介绍了论文的组织结构。

第二章主要对本论文的相关基础知识进行介绍。首先介绍了斗地主的游戏规则，之后介绍了机器博弈的经典方法，最后介绍了深度强化学习中的基本概念和基础算法，为后续章节使用这些算法进行研究做好准备工作。

第三章为结合残差网络的深度蒙特卡洛方法。首先介绍了深度蒙特卡洛方法，对比分析了在斗地主算法中采用深度蒙特卡洛方法的优点。之后介绍了结合残差网络的斗地主算法框架，并详细介绍了本次实验中的多 Actor 算法和残局规则等实验细节。最后在 RLCard 框架中，将智能体与不同算法进行直接、间接的比较，并对结果进行了详细的分析。

第四章为结合随机优先采样的深度蒙特卡洛方法。首先介绍了随机优先采样算法，之后介绍了结合随机优先采样的深度蒙特卡洛模型，对所提出算法的实现细节进行了详细描述。最后详细分析了与其他的斗地主算法的对战实验结果。

第五章为结合多智能体强化学习的深度蒙特卡洛方法。首先介绍了多智能体强化学习算法。之后介绍了结合多智能体强化学习的深度蒙特卡洛模型，其中重点介绍了基于极大极小 Q 算法的地主角色训练方法。最后通过相关实验，分析了该训练方法下地主智能体的实际智能水平。

第六章为总结与展望。首先总结了本次毕业设计的研究结果，之后针对目前算法中还存在的问题，展望了未来斗地主算法可以提升的研究路径。

第 2 章　相关基础知识

本章将介绍与本文研究内容相关的基础知识。首先简单介绍了斗地主的游戏规则，其次介绍机器博弈的经典方法，再简要介绍了深度强化学习中的基础概念和算法，为后续章节使用这些算法进行本文的研究做好准备。

2.1　斗地主游戏规则

斗地主游戏俗称二打一，因为其玩法简单、娱乐性强，已成为国内最受喜爱的牌类游戏之一。在 2013 年国家体育总局正式认定斗地主游戏为正式比赛的项目之一。本章节详细介绍本次研究的斗地主游戏规则。

（一）术语约定

(1) **局：**一副牌包括发牌、叫牌、出牌、记分的博弈过程，称为一局。

(2) **轮：**三个参赛选手共坐一桌完成若干局，其中保持初始手牌相同，但是玩家的角色互换。

(3) **牌型:** 玩家可以出牌的手牌组合，主要包括火箭、炸弹、单牌和其他牌型。

(4) **地主:** 作为地主角色的玩家，可以计为 $Landlord$ 或 L

(5) **农民一:** 在地主前出牌的农民玩家，可以计为 $Up-Peasant$ 或 U

(6) **农民二:** 在地主后出牌的农民玩家，可以计为 $Down-Peasant$ 或 D

（二）比赛过程

一局牌需三个玩家参与，过程包括发牌、叫牌、出牌和计分四个步骤。叫牌环节，一个玩家成为地主（庄家），其余两个玩家作为农民（防守方）与地主对抗。如果某一玩家首先出完手牌，牌局结束并判定先出完手牌的角色胜利。在一局游戏中可以分为以下步骤:

(1) **发牌:** 斗地主中一副牌总共为 54 张，其中包含四个不同的花色：黑桃、红心、方块、梅花。每张牌自大到小的次序为大王、小王、2、A、K、Q、J、10（T）、9、8、7、6、5、4、3，其中大小王只有这两个花色，不同花色的手牌大小相同。每个玩家先发 17 张，剩余 3 张作为公共牌，在地主未确定之前所有玩家都看不到，地主确定后，公共牌可以分给地主并对所有玩家开放。

(2) **叫牌：**每轮从西方玩家开始叫牌，并按出牌的顺序轮流进行，每人叫一

次牌。叫牌时可以叫 “1 分”，“2 分”，“3 分” 或 “不叫”。后面的玩家只能叫比前面高的分数或者不叫。如果有玩家叫分后，另外两人选择不叫或有牌手叫到 “3 分” 则结束叫牌，叫牌结束后所叫分值最大的牌手确定为地主。

(3) **出牌：**每局由地主先出，按座位上逆时针顺序，玩家依次出牌或者不出。后续跟牌者须按照同样牌型和张数进行跟牌，也可出炸弹或者火箭，后面的出牌必须大于前面一个玩家的出牌。如果连续两次都不出则最后的出牌玩家可再出任意牌型。为了方便研究，本文的智能算法只考虑出牌阶段，即在地主和农民确定之后所采取的出牌策略。

(4) **出牌大小的比较：**对于单牌，自大到小的牌张分值次序为大王、小王、2、A、K、Q、J、10、9、8、7、6、5、4、3。各花色之间不区别大小。对于组合牌，除火箭及炸弹外，必须在牌型与张数均相同时方可进行比较。其中对牌、三条、单顺、双顺、三顺以最大牌张比较大小；三带一、三顺带牌、四带二可以直接按照其中的三条、三顺、四条的牌张比较大小。火箭大于炸弹，火箭及炸弹均大于其他牌型，炸弹之间按牌张大小进行比较。

(5) **胜负和积分判定：**如果地主先于两个农民出完手牌，则地主获胜。如果任意一个农民先于地主出完手牌则农民获胜。同时在得分上，三位玩家都有自己的积分，初始时均为一分，在比赛中每打出一个炸弹则分数翻倍，比赛结束时可以计算得分。

在比较不同算法的实战水平时，可以利用**平均胜率**（Winning Percentage，WP）和**平均差异得分率**（Average Difference in Points，ADP）来进行裁定，以下简称为**胜率**和 **ADP**。其中平均胜率等于智能体获胜的局数除以所有对战局数，平均差异得分率等于每局最终算法的得分相减后取平均。如果算法一在对战算法二时胜率大于 50%，则说明算法一的对战胜率水平大于算法二。如果算法一在对战算法二时 ADP 大于 0，则说明算法一的实战得分水平大于算法二。不同算法在地主和农民角色上的实战水平也有比较大的差异，需要利用胜率和 ADP 共同作为衡量算法智能性的标准。在本文中，考虑到胜率这一指标更符合传统的胜负判断，将作为主要参考指标和模型的训练目标，而 ADP 将用于辅助分析策略的进攻性和保守性。

（三）牌型介绍

在斗地主游戏中，规则规定火箭大于炸弹和普通牌型、炸弹又大于普通牌

型。这里要注意的是首次匹配成功则确定为该牌型，之后不再向下匹配手牌，所带牌点数可以相同。有关的牌型见下表 2.1。

表 2.1 手牌牌型

Table 2.1 Card type in ”Doudizhu”

牌型	描述与备注
弃权	不出牌
火箭	出大王和小王，是最大的牌型
炸弹	四张相同点数的牌型（比如 AAAA）
单牌	一张单独的牌型（如 A）
单顺	五张及以上的连续单牌（如：ABCDE 或 ABCDE...）
对牌	两张点数相同的牌型（如 AA）
双顺	三对及以上的连续对牌（如：AABBCC 或 AABBCC..）
三条	三张点数相同的牌型（如 AAA）
三顺	两个及以上的连续三张点数相同（如：AAABBB 或 AAABBBCCC...）
三带一	三张点数相同或加上一张单牌或一对的牌型。（如：AAA+B 或 AAA+BB）
三顺带牌	三顺带同数量的单牌或同数量的对牌
	（如：AAABBB+C+D 或 AAABBB+CC+DD 或 AAABBB... +...+Y+Z 或 AAABBB... +...+YY+ZZ）
四带二	四张同点数牌带 2 张单牌或 2 对牌（型如 AAAA+B+C 或 AAAA+BB+CC）

2.2　机器博弈的经典算法

在深度强化学习技术发展前，机器博弈的经典算法主要可以总结为博弈树搜索、遗憾值最小化算法和基于规则和人类经验的专家系统等。在简单的两人零和完美信息博弈中，博弈树搜索算法已经能较为完美地处理如跳棋、象棋和五子棋等游戏。但对于其它状态空间较大的完美信息博弈，基于博弈树的搜索算法会因为搜索深度的层层递增，使得需要处理的状态数量也呈指数增长，无法实现博弈树的完全搜索，如围棋这种大规模完美信息游戏。基于纳什均衡的遗憾值最小化算法，可以通过求解博弈问题的纳什均衡解，做出最终的动作决策。但是由于在不完美信息博弈上直接求解出均衡难度较大，因此产生了基于虚拟遗憾值最小化算法求解近似纳什均衡的方法。此外，基于规则和专家经验的知识库算法在机器博弈中也可以成为重要的一个技术路径。

2.2.1 博弈树搜索

博弈树搜索算法首先成为机器博弈理论中的研究热点。博弈树是将游戏的初始状态作为根节点，其他所有可能的后续状态作为子节点，叶子节点代表了游戏从初始状态出发到牌局结束的所有可能对局记录。两个节点之间的边表示玩家选择的出牌动作，出牌前状态为父节点，出牌后的状态为子节点。

在基于博弈树求解机器博弈问题时，可以通过搜索博弈树在当前状态下的所有子节点状态，选择出收益最大的动作作为最佳决策。对于五子棋、国际象棋这种简单的完美信息博弈游戏而言，遍历博弈树就可以获得合理的动作。但是对于状态空间复杂的游戏例如围棋、德州扑克等游戏，因为博弈中的状态空间规模巨大，无法对整个博弈树进行遍历。所以剪枝技术成为了博弈树搜索中的关键技术。*Alpha -Beta* 剪枝利用博弈树中不同分支的价值存在差异，在搜索前对父节点价值不明确的分支直接剪去，不进行搜索 (Fuller 等, 1973)。贪婪搜索主要是利用了贪心的方式，对每一次搜索的决定都选择价值最大的子节点 (Lu 等, 2008)。极小窗口算法主要是利用搜索过程中的探索与发现相平衡的方法，查找所有的子节点，最终利用最小窗口的划分来查找最终的动作状态 (Marsland 等, 1982)。除了剪枝搜索以外还可以利用随机搜索比如蒙特卡洛搜索等方法。采样次数的增加可以提升算法搜索的成功率，利用随机的方法进行大量重复实验可以提高最终结果的正确概率。

2.2.2 遗憾值最小化算法

在博弈问题中，当纳什均衡时局势达到稳定，此时在所有玩家所作出的策略组合上，任意一个玩家单独改变动作策略都无法提升自己的奖励值。Nash Jr (1950) 表明博弈中玩家为有限个并且如果每位玩家的动作集大小都有限时，博弈必存在纳什均衡，玩家的策略可以是混合策略。在这样的局面中，任意一个玩家如果按照纳什均衡策略来决策，都可以处于不败。但是复杂的多人拓展式博弈问题很难直接求解出纳什均衡，这主要是因为在“斗地主”等游戏中状态空间和决策空间的规模很大，使得计算复杂性高。Hart 等 (2013) 等人提出了遗憾最小化算法，算法的主要思想是利用玩家在每次做出动作决策时都会产生遗憾值，而基于对算法中的策略进行迭代使得整体的遗憾值收敛，最终对所有迭代策略取平均来作为最终所求策略，数学上证明平均策略可以随着遗憾值的减小而逐

渐趋近于纳什均衡解策略。

但是因为在不完美信息博弈的游戏中存在不开放的博弈信息，这些信息是只针对特定玩家的私有信息，这就使得上述算法在计算遗憾值时不能遍历出所有的可能情况，因此遗憾值最小化算法也无法解决非完美信息博弈问题。针对不完美信息博弈中的私有信息，Zinkevich 等 (2007) 等人提出了虚拟遗憾最小化算法。通过对不同游戏状态下的玩家信息集，求其虚拟价值和虚拟遗憾值，不断更新迭代策略，实现虚拟遗憾值不断趋近于无穷小直到为零，最终同样可以求出近似纳什均衡策略。

2.2.3　专家系统

因为在博弈游戏过程中经过训练的人类玩家可以产生大量的经验，会逐渐形成自己的知识经验。这些高水平玩家通常可以在博弈比赛中拿到很好的成绩。因此如果将高水平玩家的经验形成规则为主的知识库体系，并基于这样的规则知识库可以形成智能博弈系统。专家系统可以通过利用现有的专家知识和经验进行博弈，提升了机器博弈的智能水平。

获得基于规则和人类经验的专家库一般分为以下步骤 (彭文, 2020)：

(一) **知识的获取：**寻找专家和高水平玩家的博弈对局记录和经验成果，进行知识库的总结归纳。

(二) **知识的表示：**规范化的表示上面所获取的专家规则和经验成果等知识，形成基于规则的专家知识库，以方便后期查找和使用。

(三) **知识的运用：**分析查找当前的博弈状态，可以根据知识库中的知识进行合理的总结和归纳，快速找到应对当前状态的合理动作策略。

(四) **知识库的更新：**每一次利用知识库解决新状态下的决策都可以得到新的经验和知识。在利用现有的专家库进行博弈时，需要根据实际的对局博弈情况及时更新现有规则知识库中的局部最优解，保证知识库迭代更新。

虽然基于规则和专业知识基础的智能游戏系统已经取得了一些成功，但是很难推广它的使用。这种方法仍然存在一些缺点 (马骁 等, 2010)：

(一) 获得专业知识往往很困难。这样的知识和经验往往记录在历史游戏日志中。例如，中国象棋记录往往需要很长时间才能总结。此外，搜索各种历史记录并不容易；

(二) 获得的知识还需要利用形式化的方式进行归纳整理并且识别正确与错误，再保存到电脑上。这需要大量时间来总结和记录，无法适应当今的大量数据快速生成的社会环境；

(三) 专家知识库内无法包含游戏中遇到的所有状态，这就使得知识库存在没有见过的对战状态，自然也无法指导人类做出合理的决策。并且专家知识库通常难以解释游戏环境中遇到的所有条件；

2.3 深度强化学习发展历程

深度强化学习算法结合了深度学习和强化学习各自的优势，可用于解决具有复杂状态空间的机器博弈游戏。目前在诸如围棋 (Zhang 等, 2020)、象棋 (王骄等, 2005) 和德州扑克 (Brown 等, 2018) 等不同机器博弈游戏上，深度强化学习已经取得了突破性进展。基于深度强化学习的决策过程是利用深度神经网络从博弈游戏中提取出具体的特征，并将其作为深度强化学习算法的输入。随后，深度强化学习算法根据这些状态信息，选择动作策略作用于当前的游戏状态，并得到最终的奖励值。整个过程不断迭代优化，最终形成在不同机器博弈上的高水平智能体。

本节主要介绍深度强化学习在机器博弈和智能决策领域的背景知识和发展历程。首先介绍深度学习了的主要网络模型，然后介绍强化学习的基本概念，接着介绍了一些常见的深度强化学习基础算法。

2.3.1 深度学习

深度学习的起源是基于对深度神经网络的研究。通过模拟人大脑中神经网络的学习和思考过程，研究人员提出可以利用深度网络来逼近任意的函数。上世纪九十年代，研究人员通过模拟大脑中的复杂层状结构，提出了多层感知机以及反向传播算法。但是因为当时存在梯度弥散问题，并且计算机的计算能力不足，神经网络的研究一直没有取得其他的进展。

进入 21 世纪，随着计算能力和储存空间的不断提升以及网络中数据的急速增长，深度学习在图像识别、自然语言处理等领域取得了突破性的进展。Hinton 等人提出了一种用于预训练中间层神经网络的无监督方法，再采用监督方法对整个网络进行优化 (LeCun 等, 1998)。Dahl (2001) 等建立了深度神经网络和隐马尔科夫模型，并首次应用于单词识别问题，识别效果相比于传统模型也取得了显著

的提升。Kirzhevsky 等 (2012) 等提出了深度卷积神经网络模型，最终在 ImageNet 上将图像识别的错误率降低到 37.5%。Graves (2012) 等利用结合了长短时记忆的递归神经网络，在语音处理上比传统神经网络更有效。之后深度学习在很多领域都取得了突破性进展，发展出了包括注意力，生成对抗学习以及深度残差网络等多种模型。

深度神经网络的特点是许多个连续的隐藏层，每一层都包含一个激活函数。这些激活函数是非线性变换。通过不同的转换序列形成对数据不同层次的抽象学习，以一个神经网络的结构为例如图 2.1(Goodfellow 等, 2016)。

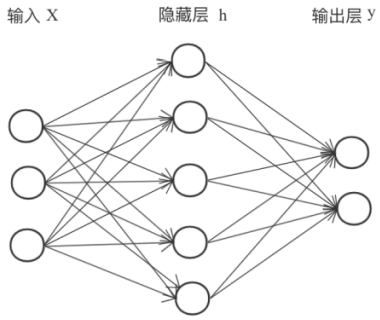

图 2.1 一个神经网络结构的例子 (Goodfellow 等，2016)

Figure 2.1 Example of a network structure(Goodfellow et al.,2016)

2.3.1.1　卷积神经网络

在上述的前馈神经网络的基础上，卷积神经网络还加入了卷积层。通过模拟人脑如何对数据进行分类并实现不同级别的属性提取和处理，设计了卷积层和池化层。并且为减少深度神经网络中参数爆炸的问题，卷积神经网络还实现了权值共享，这大幅缩减了参数数量，提高了网络的学习效率。

卷积神经网络由卷积层和池化层交替的层叠所形成。卷积层采用权重共享机制，使得网络的参数减少，池化层包括了最大值池化或均值池化两种方式来降低图像维度，并且学习到的特征具有平移、旋转不变性 (LeCun 等, 1998)，其结构如图 2.2(Goodfellow 等, 2016)。在前向计算中，图像数据经过多层转换和特征缩减后从输入层中提取出来，然后它被发送到全连接层并接收网络输出。在反向传播时，算法将损失函数的 Loss 传播回每一层网络中，并使用梯度下降法优化神经网络的参数。卷积神经网络非常适合处理图像识别等相关问题，He 等 (2016) 为了卷积神经网络的层数可以进一步扩大化，提出了深度残差网络，最终

该模型也在 ImageNet 图像分类任务上面取得了当时最好的成绩。

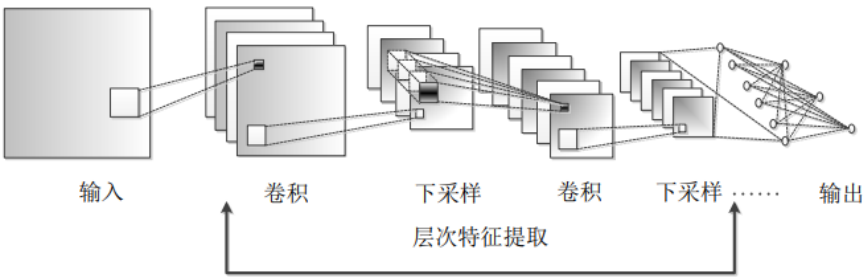

图 2.2 卷积神经网络结构图 (Goodfellow 等，2016)

Figure 2.2 The structure of convolutional neural network(Goodfellow et al.,2016)

2.3.1.2 循环神经网络

与卷积神经网络模型所不同的是，循环神经网络不考虑数据之间的卷积关系。循环神经网络需要存储之前的信号数据，并将其应用于当前层的计算结果。也就是说，隐藏层之间的节点是相连的 (Sigtia 等, 2015)，其结构如图 2.3所示 (Goodfellow 等, 2016)。

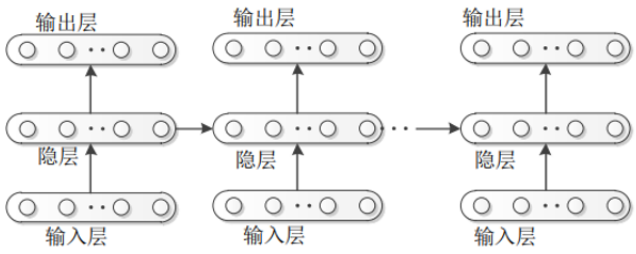

图 2.3 循环神经网络结构图 (Goodfellow 等，2016)

Figure 2.3 The structure of recurrent neural network(Goodfellow et al.,2016)

循环神经网络通过处理输入的序列数据，并且每层网络都同时处理过去所有序列的历史信号。循环神经网络是非常强大的动态系统，通常使用反向传播来训练，算法可以解决非长期依赖关系的影响。但是如果循环神经网络的输入序列太长，在反向传播的继承过程中会导致梯度增大或减小到零。长期和短期记忆细胞可以处理上述问题 (Hochreiter 等, 1997)。递归神经网络主要用于处理时序数据，常应用在自然语言处理和语音识别中。

2.3.1.3　残差网络

深度神经网络因为包含了更多的隐藏层和参数而具有更强的表征能力和泛化能力。一般而言，如果网络的层数越大，深度神经网络能够提取特征的能力也就会越强。但是在随着网络层数的加深时，模型会出现网络退化、模型精确度下降以及训练时间过长等问题。而且如果仅仅只通过增加网络层数来增加深度神经网络的泛化能力，也只会导致计算资源的过度消耗。模型也会出现过拟合现象，即随着网络层数的增加，在训练集上的准确率出现饱和甚至下降的现象。这说明虽然深层全连接网络网络的泛化能力强，但是很难训练到全局最优解。为了保留深层网络的深度，又可以有浅层网络的优势来避免退化，He 等 (2016) 在深层网络中引入短路连接的方式，其网络结构如图2.4所示 (He 等, 2016)。其中 x

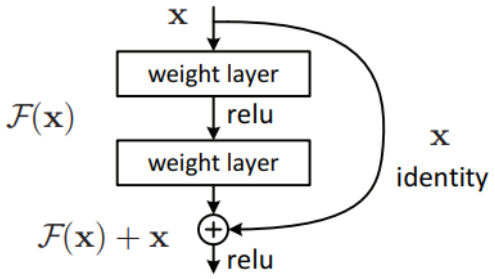

图 2.4 残差神经网络结构图 (He 等，2016)

Figure 2.4 The structure of ResNet(He et al.,2016)

和 $F(x)$ 为网络中的信号，Relu 为激活函数。根据上述结构可以得到

$$H(x) = F(x) + x \Rightarrow F(x) = H(x) - x$$

只要 $F(x) = 0$ 就构成了一个恒等映射 $H(x) = x$，这里 $F(x)$ 是指残差模块。残差网络利用了两种模块来解决退化问题，分别为短路连接部分以及残差模块。$F(x)$ 是之前的网络信号，$H(x)$ 是求和后的网络信号。在网络的前向传播过程中，残差结构的信号可以表达为：

$$x_{l+1} = x_l + F\left(x_l, W_l\right)$$

由此，任意的第 L 层深层的信号的表达为：

$$x_L = x_l + \sum_{i=l}^{L-1} F\left(x_i, W_i\right)$$

其中第 L 层单元的信号，x_L 等于浅层网络的信号 x_l 加 $\sum_{i=1}^{L-1} F$ 的残差部分，表明了任何单元 L 和 l 之间都具有残差的特性。同样的，对于任意深的单元 L，它的特征为：

$$x_L = x_0 + \sum_{i=0}^{L-1} F\left(x_i, W_i\right)$$

即为残差网络的信号的总和加上 x_0。而在网络的反向传播时，如果损失函数设定为 E，根据链式法则可以得到：

$$\frac{\partial \varepsilon}{\partial x_l} = \frac{\partial \varepsilon}{\partial x_L}\frac{\partial x_L}{\partial x_l} = \frac{\partial \varepsilon}{\partial x_L} \times \left(1 + \frac{\partial}{\partial x_l}\sum_{i=1}^{L-1} F\left(x_i, w_i\right)\right)$$

上面的式子可以分为两个部分：

(1) 不通过权重层的传递：

$$\frac{\partial \varepsilon}{\partial x_L}$$

(2) 通过权重层的传递：

$$\frac{\partial \varepsilon}{\partial x_L}\left(1 + \frac{\partial}{\partial x_l}\sum_{i=1}^{L-1} F\left(x_i, w_i\right)\right)$$

第一个部分可以保证信号能够直接传回到任意的浅层 x_l，同时也说明了不会出现梯度消失现象的原因，因为

$$\frac{\partial}{\partial x_l}\sum_{i=l}^{L-1} F\left(x_i, w_i\right) \neq -1$$

残差网络在训练时容易优化，并且能够通过增加网络的深度来提高模型的泛化能力。残差网络在内部使用了残差块和短路连接，缓解了在深度神经网络中增加网络层数所带来的梯度消失和退化等问题。将残差网络作为斗地主算法中的 Q 网络，可以有效的降低智能体胜率的不稳定性，同时也可以在整体上提高了算法的智能水平。

2.3.2 强化学习

强化学习（Reinforcement Learning，RL）是机器学习中的一个重要研究领域。强化学习通过模拟人类和环境做互动，从而学习出决策的。环境可以根据智能体的行为反馈相应的奖励值，智能体根据当前的状态和奖励值采取下一步的行为。在强化学习中包含如下几个基础概念：

(1) **外部环境：**是指跟智能体进行交互的环境，输入是智能体当前的状态和行为，输出是当前的奖励值和智能体的下一步状态。

(2) **智能体：**是指在游戏中能够采取行动的玩家。在现实生活中智能体就是玩家本人，在游戏中强化学习算法就是智能体。

(3) **行动：**是指智能体做出的行为反应。根据环境给出的反馈，智能体能够做出相应的行为。

(4) **状态：**是指智能体当前的情况。智能体做出行为前后，智能体所处的环境，并且智能体的接下来的行为也要考虑到当前的状态。

(5) **奖励：**奖励就是智能体在做出某个行动后获取的反馈值。例如：在斗地主游戏中，若地主赢了该场比赛，则地主的奖励值为正值。对于智能体的一个行为，环境给出相应的奖励值，可正可负。奖励可以是实时的也可以是滞后的，它能够使智能体及时改变原有的行为轨迹。

(6) **策略：**是指智能体在当前的情况下要采取接下来的行为所用的策略。

(7) **Q 值：**和奖励基本相同，但一般来讲 Q 值严格指当前的行为的奖励值。

强化学习的智能体可以在训练过程中与环境进行实时的交互, 其主要训练过程如下图所示 2.5。

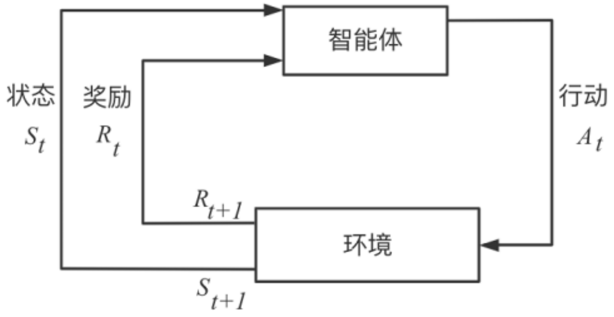

图 2.5 强化学习过程

Figure 2.5 Reinforcement learning model diagram.

智能体根据环境提供的当前状态 S_t 选择一个行动 A_t，环境可以反馈给智能体该状态下采取该行动的奖励 R_t，并且环境会发生变化，变为下一个时刻的状态 S_{t+1}。智能体通过不断的试错，将获得的奖励作为监督信号，最终可以训练迭代成具有一定决策能力的智能体。因此，强化学习算法可以应用在机器博弈、智能决策等领域。

2.3.3 深度强化学习

深度强化学习（Deep Reinforcement Learning，DRL）结合了深度学习和强化学习，既具备深度学习强大的理解力，也具备强化学习强大的决策能力，可以实现端到端的学习，从而能够对原始输入到输出进行直接控制。自深度强化学习提出以来，在很多需要对原始输入进行控制的操作中，都得到了实质性的应用。本章节主要介绍三种主要的深度强化学习算法，包括基于值函数、基于策略梯度和基于搜索与监督的深度强化学习算法，并着重介绍本文应用到的深度 Q 学习（Deep Q-learning，DQN）算法。

（一） 基于奖励值的深度强化学习算法

Watkins 等 (1992) 等人首次提出 Q 学习模型，Q-learning 算法需要创建状态和动作存储表，具体过程见算法 1。

算法 1 Q-learning 算法

输入： 初始化的状态 s、行动 a 和 Q 值

1: **for** $iteration = 1, 2, \ldots$ 到收敛 **do**

2: 根据当前状态 s_t, 选取最优动作 a_t 和下一个状态 s_{t+1}

3: 分析状态 s_{t+1}，返回奖励 r_{t+1} 更新 Q 值

4: $Q\left(s_t, a_t\right) \leftarrow Q\left(s_t, a_t\right) + \alpha\left[r_{t+1} + \gamma \max_a Q\left(s_{t+1}, a\right) - Q\left(s_t, a_t\right)\right] \quad (0 \leq \gamma \leq 1)$

5: 更新状态: $s_t \leftarrow s_{t+1}$

6: **end for**

但是对于状态空间大的博弈问题，Q 表的存储空间和相应的计算时间都会大大增加。因此，为了解决一些状态空间非常大的博弈问题，Mnih 等 (2013) 提出了深度 Q 学习（Deep Q-network，DQN）算法，用神经网络代替 Q 表来计算 Q 值，利用游戏的奖励作为监督训练网络，并将其用于解决 Atari 游戏的决策问题。DQN 算法是深度强化学习算法中较为经典的算法，该算法是基于传统的 Q-learning 算法进行改进的，神经网络不用进行存储 $Q(s, a)$，因此能够解决传统 Q-learning 算法的问题，具体过程见算法 2。

DQN 算法是和 Q-learning 算法一样，都是通过当前动作计算 Q 值进而进行决策的。但是在非完美信息博弈游戏中很多情况下都需要考虑历史信息，不能仅根据当前动作和状态就决定下一步决策，例如斗地主游戏中，玩家出牌前需要考虑其他玩家之前几手牌，进而选择一个最终获益最大的出牌。

算法 2 Deep Q-learning 算法

输入： 样本池 $\mathcal{D}$ 初始化容量 N，Batch 大小 M，学习率 ψ

随机初始化动作值函数 Q
for $iteration = 1, 2, \ldots$ 到收敛 **do**
　初始化序列 $s_1 = \{x_1\}$ 并预处理序列 $\varphi_1 = \varphi(s_1)$
　for $t = 1, 2, \ldots, T$ **do**
　　以概率 ϵ 选择一个随机动作 a_t
　　否则选择 $a_t = \max_a Q^*(\varphi(s_t), a; \theta)$
　　在模拟环境中执行操作并观察奖励 r_t 和样本 x_{t+1}
　　令 $s_{t+1} = s_t, a_t, x_{t+1}$，预处理 $\varphi_{t+1} = \varphi(s_{t+1})$
　　将 $(\varphi_t, a_t, r_t, \varphi_{t+1})$ 存储到 $\mathcal{D}$ 中
　　从 $\mathcal{D}$ 中提取随机小批量样本 $(\varphi_j, a_j, r_j, \varphi_{j+1})$
　　$y_j = \begin{cases} r_j & \text{为} \varphi_{j+1} \\ r_j + \gamma \max_{a'} Q(\varphi_{j+1}, a'; \theta) & \text{非} \varphi_{j+1} \end{cases}$
　　在 $(y_j - Q(\varphi_j, a_j \theta))^2$ 梯度方向上进行梯度下降
　end for
end for

（二）　基于策略的深度强化学习算法

策略梯度主要是通过优化策略的深度强化学习算法，如 A3C 算法等 (Sutton 等, 1999)。在训练过程中，通过计算出策略奖励的期望，以及其对参数的梯度来更新策略网络中的参数。因此相比于直接搜索最优策略，在深度强化学习问题中用策略梯度方法来优化策略，省略了许多不必要的中间游戏步骤。但策略梯度方法需要更强大的硬件支持，应用到多人非完备信息博弈中计算资源的要求过高。

（三）　基于搜索的深度强化学习算法

该方法需要添加人类专家经验来完成搜索，例如蒙特卡洛树搜索算法就是一个经典的启发式策略搜索方法，因此大部分策略搜索都是通过蒙特卡洛树搜索，例如 AlphaGo 就基于蒙特卡洛树搜索对围墙局面的所有可能状态进行搜索，并在人类顶级玩家的比赛中取得了理想的成就 (Chen, 2016)。但是如果将该方法应用到斗地主游戏中，需要大量的人类经验作为监督数据，而且训练时间较长，实际效果也一般 (Jiang 等, 2019)。

2.4 本章小结

本章主要介绍了本次研究的相关基础知识。首先介绍了本次研究的斗地主游戏规则，之后回顾了机器博弈的经典算法，包括博弈树搜索、虚拟遗憾值最小化和专家系统。最后通过介绍深度强化学习，引出本次研究的基于奖励值的深度强化学习方法。

第 3 章　结合残差网络的深度蒙特卡洛方法

本章介绍基于深度蒙特卡洛方法和残差网络所设计的斗地主算法框架，并且通过结合基于规则的残局策略，合理运用两个农民间的合作关系，进一步地提升整体策略的智能水平。将改善后的斗地主算法作为最终决策策略，最后对实验结果进行了分析。

3.1　深度蒙特卡洛方法

深度蒙特卡洛方法（Deep Monte-Carlo Methods，DMC）是一种典型的强化学习算法，在博弈问题上可以利用自博弈学习来训练智能体。深度蒙特卡洛方法非常适合用于解决“斗地主”中的特有挑战，算法可以快速生成大量的斗地主游戏对局记录作为样本，提高了训练的效率。

深度蒙特卡洛方法起源于传统的蒙特卡洛方法（Monte-Carlo Methods，MC），通过使用大量重复实验来随机模拟现实环境，以解决实际问题。如果在利用强化学习解决实际问题时没有完整的环境知识，使得计算过程过于复杂难以得到解析解，或者该问题根本不存在解析解的时候，蒙特卡洛方法可以利用估计的价值函数来寻找最优策略。蒙特卡洛方法的主要思想是：利用智能体直接与环境交互的历史经验，在交互中采样，获得状态、动作和收益的组合序列。再根据平均的样本回报收益来确定动作的实际价值，最终形成合理的动作策略。在利用蒙特卡洛方法时，状态的价值可以表示为未来折扣奖励的累计值期望，根据实验获得的大量数据，对由该状态出发之后产生的回报收益求平均，最终该平均值会逐渐收敛到动作的期望价值。

在对动作价值进行蒙特卡洛估计时，需要计算动作的价值，在机器博弈中是指“状态和动作”的二元组价值，也叫做动作价值函数。该方法主要是估计 $Q_\pi(s,a)$，即在状态 s 下，遵循策略 π 所做出动作 a，得到的期望回报为 $Q_\pi(s,a)$。基于动作价值和基于状态价值的蒙特卡洛方法在步骤上基本相同，只不过现在求解的是“状态和动作”而不是状态。一个“状态和动作”对 (s,a) 即是在一个回合里，访问到状态 s，并做出动作 a。在每次访问 MC 方法中，访问“状态和动作对”都会计算回报的平均值。而在首次访问 MC 方法中，每个回合只计算在这

个状态下第一次访问所获得的回报平均值。当访问次数非常大时，每次访问 MC 方法和首次访问 MC 方法都会以二次函数的方式收敛到期望值。

将蒙特卡洛方法应用到离轨策略下的时序差分控制算法时，可以利用 Q - Learning 算法进行蒙特卡洛模拟，通过迭代的方法对 Q 表格进行更新。具体的算法步骤如下：

(1) 基于初始策略 π_0 生成完整的历史对局，利用马尔科夫链表示为 (s_t, a_t, R_t)，如图3.1所示；

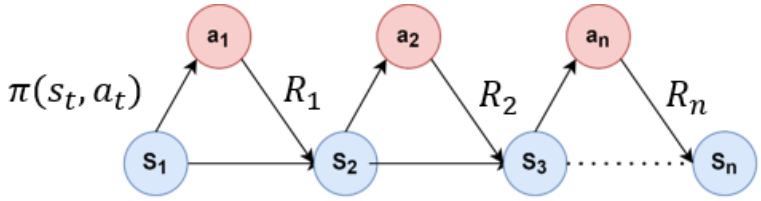

图 3.1 利用马尔科夫链表示的对局历史记录

Figure 3.1 Using MDP for match history

(2) 利用对局中成对出现的“状态-动作”对，计算奖励的期望 $Q(s,a)$:

$$Q_\pi(s,a) \doteq \mathbb{E}_\pi\left[G_t \mid S_t = s, A_t = a\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right]$$

其中 γ 为衰减系数，计算平均值后更新 Q 表格；

(3) 最优策略为使得状态价值最大的动作，对于之后的决策策略为：

$$\pi(s) \leftarrow \arg\max_a Q(s,a)$$

而深度蒙特卡洛方法也采用了上述蒙特卡洛采样再训练的步骤，不同于 MC 方法的是：

(1) 在第一步中，可以生成一个策略回合 π，并使用 ε-贪心策略平衡探索与利用，使收益函数每次都收敛到局部最优解。

(2) 在第二步中，不再使用 MC 法的收益函数 Q 表格，而将其替换为深度神经网络，也叫 Q 网络。对于每个出现在回合中的状态 s 和动作 a，都使用均方误差损失函数 (MSE) 来更新收益函数 $Q(s,a)$ 的网络。

(3) 在第三步中，对于回合中的每个状态 s，取使得期望收益最大的动作 a，并计算对应的动作：

$$\pi(s) \leftarrow \begin{cases} \arg\max_a Q\left(s_t, a\right) & \text{概率为}(1-\epsilon) \\ \text{随机选择动作} & \text{概率为}\epsilon \end{cases}$$

深度蒙特卡洛方法非常适用于“斗地主”这类扑克游戏，因为算法是通过自博弈学习的方式来模拟真实状态空间中的动作价值函数，其中的 $Q(s,a)$ 值可以利用大量历史对局中的“状态和动作对”二元组进行估计，平均的回报收益不会受到随机偏差值的影响。并且因为深度蒙特卡洛方法将价值学习和模拟采样分开计算，这样网络的收敛速度和对局的蒙特卡洛模拟无关，能够通过自博弈的方式同时生成大量的对局记录用于训练。虽然深度蒙特卡洛方法的结果方差很大，导致动作价值评估函数不易稳定。但是诸如“斗地主”这类扑克游戏的历史对局都是完整的，而且 DMC 方法可以很方便的利用多演员进行并行采样。通过扩大算法采样的样本规模，可以更快和高效地生成许多历史对局样本，缓解了大方差问题。下面将深度蒙特卡洛方法与其他经典的强化学习方法相比较，以说明深度蒙特卡洛方法更适用于斗地主游戏。

（一）与策略梯度方法相比

策略梯度方法比如 A3C 算法等是通过对环境和状态空间的建模后，直接使用梯度下降法来优化策略。在策略梯度方法中，通常需要一个分类器函数，其输出的维度大小与博弈中的行动集的规模相关。虽然策略梯度方法在一些较大的行动集空间上的实验效果较好，但算法的输出不能使用行动的相同特征来关联训练以前未见过的行动。例如，如果斗地主智能体因为选择了行动出牌 $KKKQQ$ 而获得了很好的奖励，但是该方法在斗地主中无法关联评估相似牌型例如 $QQQKK$ 的动作价值。因为许多“状态和动作对”在采样数据中并不经常出现，而这对处理非常大的动作空间和加速学习动作价值非常重要，策略梯度算法无法解决上述问题。

DMC 方法可以利用动作的相似特征来学习未见过的动作。虽然如果动作规模很大，它的执行复杂度可能会很高，但是在斗地主的大多数状态下，只有一个动作子集是合法的，所以深度蒙特卡洛方法就不需要对所有的动作进行迭代。因此，DMC 方法对于斗地主来说总体上是一种高效的算法。

（二）与深度 Q 学习方法相比

基于动作价值的算法如 DQN（算法 2）是根据该状态下的动作和下一个状态下的 Q 值进行迭代更新。虽然深度蒙特卡洛和深度 Q-Learning 算法都要求收敛到 $Q(s,a)$，但基于深度蒙特卡洛方法的斗地主算法具有以下几个优点：

（一） 在 DQN 学习中，算法所选取的近似最大行动价值往往大于实际价

值的期望，Q 值的高估偏差导致函数无法收敛，并且在行动空间集规模非常大的情况下变得更加明显。虽然经验重放技术可能会缓解这个问题，但在实践中发现 DQN 算法非常不稳定，经常在“斗地主”游戏中发散。而深度蒙特卡洛方法在估计 Q 值时不容易产生偏差，因为它直接通过最终的奖励值来计算 Q 值。

（二） 斗地主游戏是一个长序列的决策游戏，存在稀疏奖励问题，智能体在比赛中需要经历一长串没有反馈的状态，而且唯一产生非零奖励是在游戏结束时。这直接导致了 Q-learning 的收敛速度减慢，因为估计当前状态下 Q 值的需要等到下一个状态下的 Q 值。与 DQN 算法不同的是，深度蒙特卡洛方法直接利用已经完整生成的历史记录作为样本，模型的收敛不受动作序列长度的影响。

（三） 在“斗地主”游戏中，由于决策空间规模较大且动态变化，DQN 算法在每次更新中都需要在深度神经网络上迭代所有合法动作，这将导致很高的计算开销。此外，因为合法动作集在不同的状态下是不同的，这使得 DQN 算法不方便进行批次学习，因此 DQN 算法在训练收敛上需要大量时间。虽然深度蒙特卡洛方法可能会受到高方差的影响，但因为蒙特卡洛方法很容易进行并行化，每秒产生成千上万的样本作为批样本来加速训练。所以深度蒙特卡洛方法的高方差被它提供的可扩展性大大抵消了，深度蒙特卡洛方法在训练时非常高效。

3.2 基于残差网络的斗地主算法

本节将首先介绍基于深度蒙特卡洛方法的斗地主算法框架。然后，为了提高样本的生产速度和网络训练的效率，引入多演员并行计算，分为学习者（Learner）和演员（Actor）两个并行计算模块。最后，为了可以更好的利用两个农民玩家的合作关系，在策略中加入基于规则的残局策略，以提高算法的智能性。

3.2.1 斗地主算法框架

本节将主要介绍斗地主算法的主要框架。首先将介绍算法输入部分：状态和动作编码方式，然后介绍算法中的网络结构设计，包括了反映历史记录的状态网络和利用残差网络模型的 Q 网络。

3.2.1.1 状态、动作编码

基于深度蒙特卡洛方法的斗地主算法受随机分发的手牌影响较大，在与随机出牌、CQL 和 RHCP 等现有斗地主算法对战时会出现胜绩不稳定的现象。这

主要是因为全连接网络的泛化能力较弱，不能很好地应对“运气”所带来的影响，该算法的智能水平可以进一步提升。

将状态和历史行动的手牌都编码为 4×15 的 $0-1$ 的矩阵，其中每一列对应于四个不同的牌型花色，每一行对应于特定的牌数或大小王，对于“斗地主”游戏中的多种牌型所构成的 27472 种动作决策也可以利用上述矩阵进行编码，手牌和动作编码方式示例如图3.2所示。

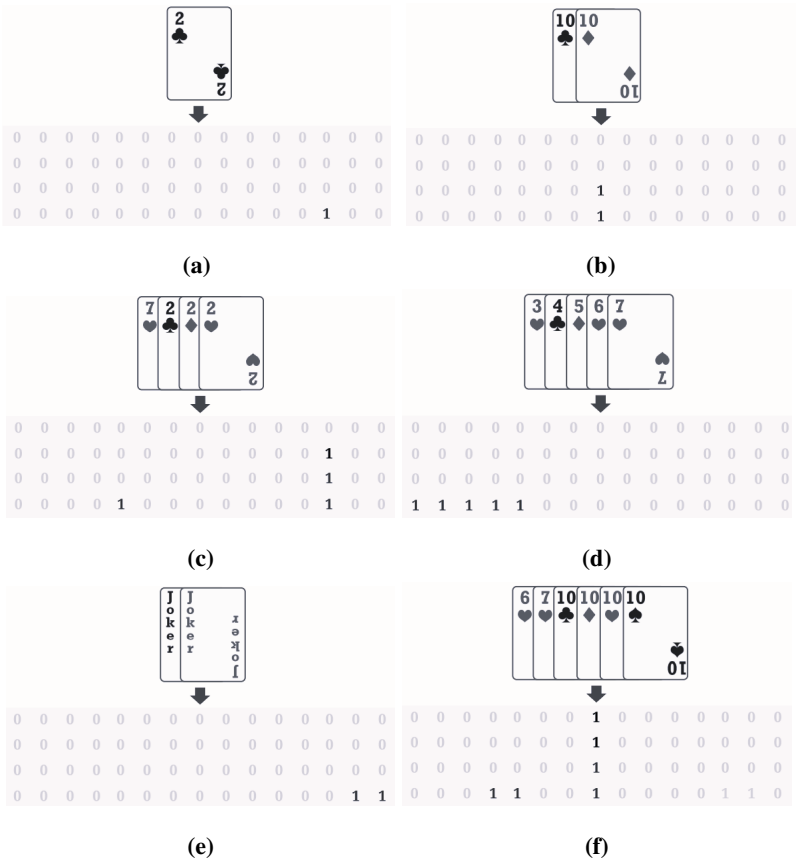

图 3.2 手牌和动作编码示例。(a) 单张牌 (b) 对牌 (c) 三顺牌 (d) 单链 (e) 火箭 (f) 某种手牌编码的示例。

Figure 3.2 Example of cards and action states' matrix. (a) Solo, (b) Pair (c) Quad with solo (d)Chain of solo (e)Rocket (f) A example of cards' state.

由此可以分别对农民角色和地主角色的所有状态和动作进行特征表示。针对三个玩家在网络中建模了三个模型，分别是第一个农民，第二个农民和地主，其中农民一的出牌在地主前，农民二的出牌在地主后面。两个农民的模型结构，状态完全一样，但是各自使用自己的模型，均使用大小为 4×15 的 0 和 1 的手牌矩阵。

地主角色需要编码的特征包括了地主自身的行动和手牌状态，这两个特征都可以用上述的的 0 和 1 矩阵表示。此外，通过去除地主手上的手牌和三位玩家已经打掉的牌，可以得到两个农民角色当下手牌的并集，这个可以帮助分析牌局目前的优劣走势，为之后基于规则的残局策略做准备。针对对手的情况，还需要记录两个农民最近的出牌和他们当前的手牌数量。因为农民在最初发牌时仅有 17 张手牌，所以可以用一个 0-1 矩阵来表示游戏中农民手牌的数量，维度为 17。

对于历史出牌记录，只选取最近五轮，也就是最近十五个出牌记录作为 LSTM 网络的训练样本。如果出牌次数不到五轮，就直接使用 0 对矩阵进行补全。因为有三个玩家，每个玩家每轮出牌一次，所以一轮的动作由三个 0-1 矩阵拼接在一起，所以过去 5 轮出牌的历史记录可以用 5*162 的矩阵表示，这里去除了表示大王和小王中不存在的花色的矩阵元素。具体的地主角色编码特征见下表 3.1。

表 3.1 地主角色编码的特征

Table 3.1 Features of the Landlord

特征	元素维度
地主的行动	60
地主的手牌状态	60
最近一次的出牌	60
其余玩家的手牌并集	60
农民一最近的出牌	60
农民二最近的出牌	60
农民一的手牌数量	17
农民二的手牌数量	17
现有的炸弹数量	15
过去 5 轮出牌的历史记录	5*162

而地主的手牌多三张，与农民的手牌状态稍有不同。但是对于农民玩家的特征结构基本相同。需要注意的是地主角色可以用来区分农民一和农民二的位置，农民角色需要编码的特征见下表 3.2。对地主和农民角色特征的不同编码可以更好的体现角色的不对等和差异。

表 3.2 农民角色编码的特征

Table 3.2 Feature of the Peasants

特征	元素维度
该农民行动	60
该农民手牌状态	60
另一个农民最近出牌	60
地主最近出牌	60
其余玩家手牌总和	60
地主手牌数量	20
另一个农民手牌数量	17
现有炸弹数量	15
过去 5 轮出牌的历史记录	5*162

3.2.1.2　网络架构设计

斗地主游戏作为一种典型的不完美信息博弈，具有长序列决策过程和多位玩家互相影响的特点。因此在考虑当前状态下的行动时，需要根据之前玩家的出牌记录做出决策，对状态的定义需要经过基于长短时记忆网络（Long short-term memory, LSTM）的状态网络的历史记录作为特征。此外，由于斗地主游戏具有状态空间和行动空间规模巨大、对手策略不固定等特点，游戏的胜负受运气的影响较大，容易出现胜绩不稳定的现象。因此需要提高网络模型的泛化能力，将基于残差网络设计 Q 网络。整体的网络框架可以总结如图 3.3所示。

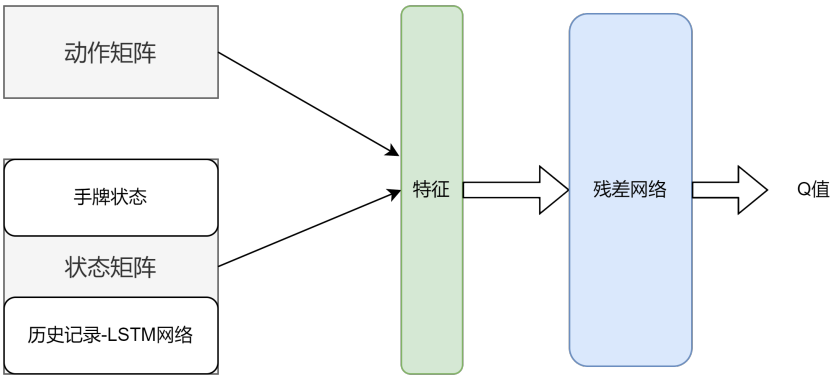

图 3.3 网络整体架构图

Figure 3.3 The structure of Doudizhu-network

（一）状态网络

对于每个 15 × 4 的手牌矩阵，DouZero 算法会将矩阵展平为大小为 60 的一维向量，然后删除六个始终为零的条目，因为有只有一个黑色或红色的大小王，即 DouZero 算法中每个卡片矩阵都被转换为大小为 54 的 one-hot 一维向量。但是在本文中，均使用 15 × 4 矩阵进行编码。这是因为首先本文采样的残差网络使用了卷积层，使用 15 × 4 的矩阵可以利用矩阵中元素的位置信息来反映动作种类的关联性，比如对牌、单链、炸弹等特定牌型在矩阵中的相对位置存在相似性，可以利用残差网络进行处理学习。

此外，进一步使用 one-hot 向量来表示其他两个玩家当前的手牌数量。例如，对于农民，向量的大小是 17，其中每个条目对应于当前状态下的手牌数量。而地主手牌数量的向量大小是 20，因为地主最多可以有 20 张牌。同样，可以使用 15 维向量来表示当前状态下的炸弹数量。对于历史动作，考虑最近的 15 个动作和连接每三个连续移动的表示。也就是说，历史出牌记录被编码成一个 5 × 162 矩阵作为 LSTM 的输入，这里不同于手牌矩阵的 15 × 4 编码方式，在对历史记录进行编码时会去除大王小王中不存在的花色元素，每个动作矩阵都压缩为只有 54 个元素的一维向量。如果历史记录上少于 15 个手牌，将零矩阵用于缺少的历史步数。

将牌局上最近的十五次历史动作，也就是每位玩家最近的五次出牌作为历史记录，进行编码后作为输入到历史动作的 LSTM 网络，输出为特征矩阵。具体结构如图 3.4所示。其中，每一个 LSTM 网络的输入维度为 5*162，隐藏层的特征维度即隐藏层神经元个数为 128，LSTM 隐层的层数默认为 1，并将最后一个状态的隐藏层的神经元输出作为历史记录特征 h。

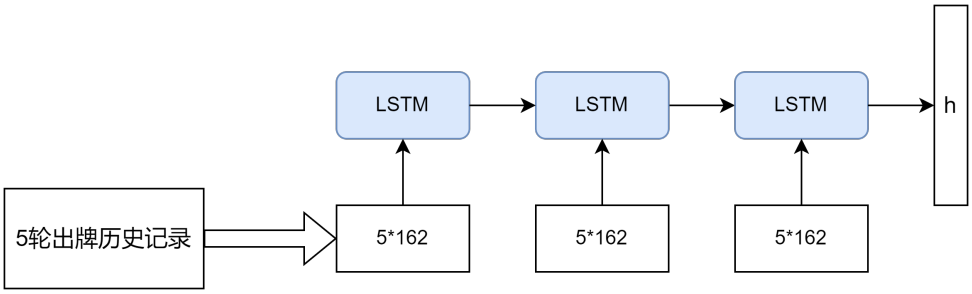

图 3.4 状态网络架构图

Figure 3.4 The structure of state-network

（二）Q 网络

利用残差网络结构构造深度强化学习的 Q 网络，其输入为“行动-状态”二元组的拼接特征，输出为预测给定状态和行动对的 $Q(s,a)$ 值。其中每一个残差模块的具体结构如图 3.5所示。Q 网络使用该残差块层叠出 32 层的残差网络，其中，BN 表示 Batch Normalization，是指对输入特征的批归一化，这种预处理方式可以在一定程度上缓解不同样本取值不同产生分布偏移过大的问题。卷积层利用维度为 $3*3$ 卷积核构造卷积网络进行下采样。而 Relu 是指利用 Relu 作为激活函数的一层线性网络。并且经过处理后的输入要与输出的尺寸和深度等维度相同，如果不相同，需要添加卷积层和 BN 层来变换为同一维度。

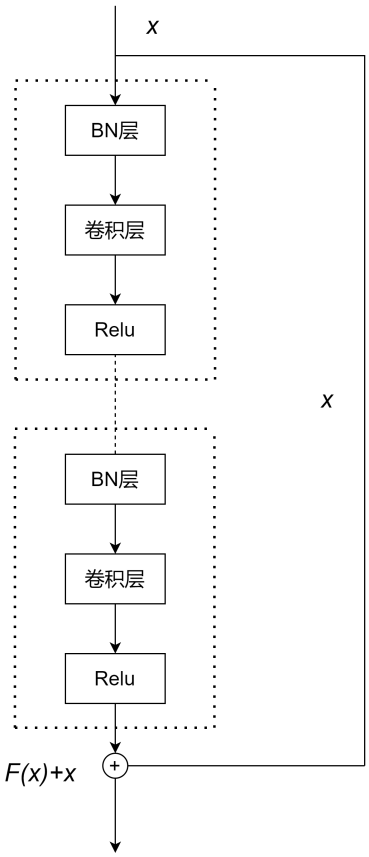

图 3.5 残差块结构图

Figure 3.5 The structure of Residual-block

3.2.2　多演员并行算法

为了提高计算效率，本文引入多演员并行计算的方法，用同时对战的多个演员（Actor）进程和一个学习者（Learner）进程来并行化深度蒙特卡洛过程。学

习者为三个玩家维护三个全局的 Q 网络，并根据演员对战所提供的历史记录数据，利用 MSE 作为损失函数来更新网络，以接近奖励的期望值。每个演员维护三个本地 Q 网络，这些网络定期与全局网络同步。演员将从游戏引擎中重复采样轨迹，并计算每个状态和动作对的累积奖励。学习者和演员可以通过三个共享缓冲区 $\mathcal{B}_L, \mathcal{B}_U$ 和 $\mathcal{B}_D$ 来实现通信，每一个训练情节结束以后都需要及时的更新演员中的本地 Q 网络，而每个阶段演员对局结束后的历史记录都需要作为新的样本训练网络。其中，演员的运行算法见算法 3，学习者的运行算法见算法 4。

算法 3 演员算法

输入: 共享缓存区 $\mathcal{B}_L, \mathcal{B}_U$ 和 $\mathcal{B}_D$ ，容量均为 $L = B \times S$ ，探索利用参数 ϵ，损失系数 γ

初始化本地网络 $Q_{\mathrm{L}}, Q_{\mathrm{U}}$ 和 Q_{D}，以及本地缓存区 $\mathcal{D}_L, \mathcal{D}_U$ 和 $\mathcal{D}_D$
for *iteration* $= 1, 2, \ldots$ 到收敛 **do** ▷ 生成 episode
    与学习者同步网络参数 $Q_{\mathrm{L}}, Q_{\mathrm{U}}$ 和 Q_{D}
    for $t = 1, 2, \ldots, T$ **do**
        $Q \leftarrow Q_{\mathrm{L}}, Q_{\mathrm{U}}, Q_{\mathrm{D}}$
        生成动作 $a_t \leftarrow \begin{cases} \arg\max_a Q\left(s_t, a\right) & \text{概率为}(1-\epsilon) \\ \text{随机选取行动} & \text{概率为}\epsilon \end{cases}$
        生成下一步状态 s_{t+1} ，奖励 r_t
        在 $\mathcal{D}_L, \mathcal{D}_U$ 以及 $\mathcal{D}_D$ 中储存 $\left\{s_t, a_t, r_t\right\}$
    end for
    for $t = T-1, T-2, \ldots, 1$ **do** ▷ 获得累积奖励
        $r_t \leftarrow r_t + \gamma r_{t+1}$ 并更新 $\mathcal{D}_L, \mathcal{D}_U$ 和 $\mathcal{D}_D$ 中的 r_t
    end for
    for $p \in \{L, U, D\}$ **do** ▷ 多进程优化
        if $\mathcal{D}_p$.length $\geq L$ **then**
            请求并等待一个空的缓存区 $\mathcal{B}_p$;
            将 L 个的 $\left\{s_t, a_t, r_t\right\}$ 从 $\mathcal{D}_p$ 移动到 $\mathcal{B}_p$;
        end if
    end for
end for

本实验的主要挑战是，学习者的训练收敛情况关系着最终模型的智能效果。在参数设置上，其中每一个共享缓存区的 $B = 100$，样本集大小为 $S = 100$，batch 大小为 $M = 32$，并且概率参数 $\epsilon = 0.01$。设置损失系数 $\gamma = 1$，这是因为斗地主

算法 4 学习者算法

输入: 共享缓存区 $\mathcal{B}_L, \mathcal{B}_U$ 和 $\mathcal{B}_D$ ，容量均为 $L = B \times S$ ，探索利用参数 ϵ，损失系数 γ

初始化全局网络 $Q_{\mathrm{L}}, Q_{\mathrm{U}}$ 和 Q_{D}
for $iteration = 1, 2, \ldots$ 到收敛 **do**
    for $p \in \{L, U, D\}$ **do**
        if $\mathcal{B}_p \geq L$ **then**
            从 $\mathcal{B}_p$ 中采样一个 batch 样本 $\left\{s_t, a_t, r_t\right\}$，大小为 $L = B \times S$
            利用 MSE 损失函数和学习率 ψ 来更新参数 Q_{p}^{g}
        end if
    end for
end for

游戏过程中只有一个非零的奖励值，所有在游戏的前期动作就非常重要。采用 RMSprop 优化器，其中的学习率为 $\psi = 0.0001$ 以及误差参数 $\epsilon = 10^{-5}$。为了在早期可以加快训练的速度和精确性，选用了 DouZero 系统中的 LSTM 网络作为 DMC-ResNet 方法中状态网络的预训练模型。

实验中使用了 45 个演员来进行对局记录的生成，并且维持一个学习者进程用于训练，同时定时从学习者拷贝新的网络模型参数过来给演员，实现信息的同步，再由自博弈学习产生新的历史对局记录。在本次实验中，选择将 2000 万次对局作为一个多演员的生产对局阶段，为了和 DouZero 系统做对比，设置相同的学习率和学习时间，整体的训练时间设置为 30 天左右，经过实际的对战实验可以发现，一般可以在一周内达到不错的胜率和 ADP 智能性效果。

3.2.3　基于规则的残局策略

和其他的不完美信息博弈游戏所不同的是，斗地主游戏存在农民角色玩家间的合作关系，这使得斗地主游戏成为了一个合作性的博弈游戏。为了更好的利用两个农民玩家的合作关系，引入基于规则的残局游戏策略。使用残局规则策略具体的流程如图 3.6所示。

针对农民角色的玩家，当前玩家在去除手牌后认定另外一名农民和地主的手牌存在弱势，即在某一位农民玩家在比赛中，该农民玩家的当前手牌可以在单独对战地主角色手牌时完全胜出，此时该农民玩家就接管比赛，不再利用当前的智能体所提供的决策过程，而直接采用完全致胜策略，以此时起，另外一名玩家

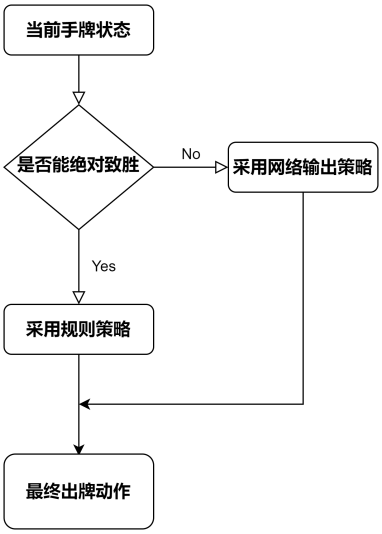

图 3.6 残局策略流程图

Figure 3.6 Flowchart of endgame rules' strategy.

就可以选择不出牌的设置。

该方法用于训练中的演员生成历史记录时，合作的另外一个农民保持不出牌。利用基于规则的残局策略可以增加对局样本的多样性，同时这样的合作机制具有明显的监督信号–不出牌，两个玩家间的合作机制很容易被学习到。当该方法用于最终对战策略时，只使用于当前玩家。基于残局的规则策略可以得到对农民地位的合理运用，整个规则策略只适用于农民角色。当在地主上的运行时，由于绝对致胜规则和基于深度蒙特卡洛的方法所得到的动作决策回大致相同，所以对于地主角色的胜率提升帮助很少，故不对地主玩家进行使用。

3.3 实验结果与分析

3.3.1 实验环境设置

本章节首先介绍了用于机器博弈研究的开源环境 RLCard 框架，再详细地介绍了 RLCard 中斗地主环境的设置，包括手牌的状态结构、玩家的动作编码、游戏最终胜负和得分的判定以及斗地主整体的对战流程。

3.3.1.1 RLCard 开源框架

本文的实验环境均采用目前已经开源的RLCard 平台作为智能体算法的研究和对战验证的平台。本文后续提出的深度蒙特卡洛方法模型都会在 RLCard 框架下训练和学习以及对战进行效果验证，下面对该框架进行简单的介绍。

Daochen Zha 等人提出的 RLCard 开源框架，是一个专门为机器博弈游戏所设计的游戏环境，其中包括了许多不同种类的游戏环境和基于深度强化学习的基础算法。强化学习已经在学术界认定为机器博弈算法中的一种主要的技术路径，该方法的主要思想是借助智能体和环境的不断交互，根据大量的实验历史记录以试错的方式学习到每个状态下的最优决策动作。同时可以利用深度学习中的大规模网络提升模型的表达能力。深度强化学习在诸如 Atari 游戏、飞行器控制和决策搜索等领域取得了理想的成绩。

但是当深度强化学习的训练过程中存在多个智能体时，会经常出现决策空间过大并且奖励稀疏的情况，在机器博弈的应用中算法依旧不成熟，会出现如下问题：

(一) 在多人游戏中往往会存在玩家间的合作关系。如在斗地主游戏中，农民需要互相合作共同对抗地主角色，但是地主需要对抗两个农民。

(二) 扑克游戏的状态空间往往会非常大。例如在斗地主牌中状态空间可以达到 10^{23}。每一个玩家都存在私有的信息会使得数据集的规模呈指数增加。

(三) 扑克游戏的行为空间也非常大。例如在斗地主游戏中的行为空间大小为 27472。

(四) 扑克游戏存在稀疏奖励。例如在斗地主游戏中需要经过较长的决策序列才能获得胜负结果，比如在斗地主游戏随机出牌时大约每 50 个回合才能确定输赢。

所以为了给其他的学术研究者们提供更加便捷的机器博弈游戏环境，RLCard 框架下还具有德州扑克、UNO、斗地主和麻将等诸多游戏环境，提前帮助其他研究者开发设计了多智能体强化学习的虚拟环境，为解决动作和状态空间规模大和长决策过程所带来的稀疏奖励等问题奠定了坚实的研究基础。它的大体结构如图 3.7

此外 RLCard 的开发者也针对斗地主和德扑等游戏开发了一些常见的基础智能体，而且在少数游戏中甚至可以获得在目前状态下的最优决策，对算法的智能性研究具有极大的便利。其中典型的学习算法包括强化学习算法中的深度 Q 学习，以及一些基于经典的机器博弈算法比如如 NFSP、CFR 及其变种和 DeepCFR。

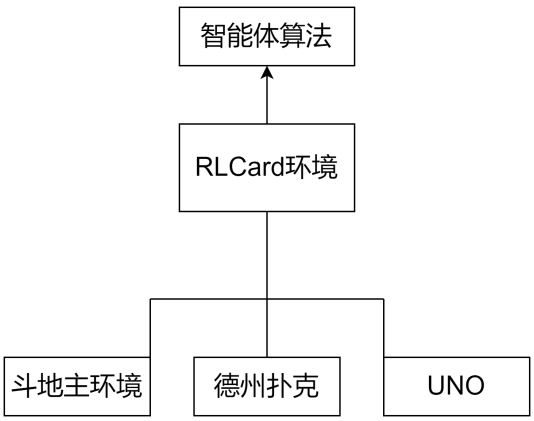

图 3.7 RLCard 结构图

Figure 3.7 The structure of RLCard

3.3.1.2 RLCard 中的斗地主环境设置

斗地主游戏作为非完美信息博弈游戏中的的一个典型多人扑克游戏，由于它的状态复杂度很高、动作决策空间的规模大等，所以就需要根据上述提出的标准斗地主游戏规则，来获得手牌的状态结构、玩家的动作的以及游戏的最终胜负和得分判定等信息。下面将介绍 RLCard 中斗地主环境的具体设置。

斗地主在中国以及东南亚地区可以称为是最受欢迎的扑克游戏之一。游戏中共有三个玩家，在桌面上一共存在 60 张牌，其中分别为四个花色的正常编码的牌型和一个为红色的大王以及另一个黑色的小王。在比赛的抢地主环节结束后，三个玩家中将只有一名玩家作为“地主”角色，所以可以分到额外的三张手牌，初始手牌共有 20 张。然而另外两个玩家则是“农民”角色，初始手牌分别为 17 张。根据游戏规则中的设定，两个农民玩家将共同对抗地主玩家。在斗地主游戏的每个回合中，开始出牌的玩家必须出牌，牌型可以是单张牌型或任意的组合牌型都可以，而其他两个玩家可以选择是出牌或者不出牌即为“过”。但是如果在游戏对局中出现了两个连续玩家都选择了不出牌即“过”，那么此游戏回合将直接结束。在上一局的历史对局记录中出牌最大的一个玩家就将成为下一轮游戏回合的第一人。斗地主游戏的胜负目标是三个玩家争取可以成为第一个出完手中所有卡牌的玩家。根据 RLCard 开源框架设定的斗地主规则，在抢地主环节，直接可以指定其中一位玩家为“地主”角色。本次研究将不包括抢地主的游戏环节，对于玩家的角色采用指定的方式进行对战实验比较，这样可以更好的比较不同类型的智能算法在不同角色上的智能水平。

（一）状态类型：在斗地主游戏的每次决策时，需要出牌的玩家可以观察到目前的所有状态，这时的决策状态有该出牌玩家能够看到的所有状态信息，状态表示的结构类型见表 3.3。

表 3.3 斗地主状态类型

Table 3.3 State type in ”Doudizhu”

关键字	描述
Deck	代表 60 张手牌, 为 52 张 $A \sim K$ 和两张大小王
Seen Card	地主发到的额外三张手牌，可以公开看到
Landlord	地主
Self	当前玩家
Initial Hand	当前玩家在比赛开始时的最初所有手牌
Trace	出牌记录
Played Cards	三位玩家出的手牌得到的排序。
Others Hand	另外两个玩家当前手牌集合
Current Hand	当前玩家的手牌状态
Actions	当前玩家采取的出牌动作

（二）动作类型：斗地主的动作空间大小是 27472，其中包括了弃权、火箭、炸弹、单牌、对牌、三条、三带一、单顺、双顺、三顺、三顺带牌和四带二。如果直接对每一个动作都进行编码会使得学习算法的规模太大，所以 RLCard 就简要总结出来 309 个动作种类，动作 ID 和数量见下表 3.4。

（三）胜负和得分判定：如果地主首先摆脱了手中的所有牌，则他将赢得并获得奖励 1，两个农民将获得奖励 -1。类似地，如果一位农民首先摆脱了手中的所有牌，两个农民都将获胜并获得奖励 1，地主将获得奖励 -1。同时三位玩家都有自己的得分记录，初始时均为 1 分，在比赛中每打出一个炸弹的玩家则分数翻倍，比赛结束时可以计算分差。

大量的重复对局以后可以利用**平均胜率**（Winning Percentage，WP ）和**平均差异 ADP**（Average Difference in Points，ADP）来进行裁定胜负，以下简称为**胜率**和 **ADP**。其中平均胜率等于智能体获胜的局数除以所有对战局数，ADP 等于每局最终算法的得分相减后取平均。当该算法在与基准算法对战的胜率大于百分之五十或者 ADP 大于零时，则该算法获胜。

表 3.4 斗地主动作类型

Table 3.4 Action type in Doudizhu

动作	初始动作数量	编码的动作 ID 数量
单牌	15	15
对牌	15	15
三张牌	13	13
三带一	182	13
三带对	156	13
炸弹带对	858	13
炸弹	13	13
王炸	1	1
链 (单牌、对牌、三张牌)	133	133
飞机带单牌	21822	38
飞机带对牌	2939	30
炸弹带单	1326	13
总计	27472	309

（四）RLCard 中斗地主的对战流程：基于对 RLCard 开源框架的应用可以设计对斗地主游戏的训练流程。游戏开始是需要初始化发牌人员和裁判，并且对于三个玩家初始化三个玩家模型，经过游戏运行的环节可以在一轮结束后得到奖励收益。整体的框架流程图可以简化为如图 3.8所示。

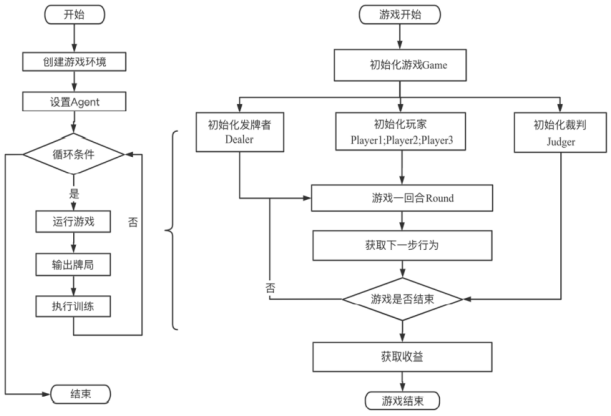

图 3.8 RLCard 中斗地主的对战流程图

Figure 3.8 Flowchart of Doudizhu Playing in RLCard .

3.3.2　对战实验结果与分析

本文研究主要将基于残差网络和深度蒙特卡洛模拟的斗地主算法，与依据其他算法所开发的智能体进行逐对的实验对战。在 RLCard 开源框架下，本次研究选择的对战算法包括随机出牌的算法、基于规则的 RHCP 算法、基于手牌拆分的 CQL 以及 DouZero 算法。在简要的介绍上述算法后，将 DMC-ResNet 智能体分别作为地主和农民角色，与其他基础算法进行对战。

实验分别选取训练过程中的三十天内的算法智能体，每隔十二个小时就对 DMC-ResNet 的智能体进行保存并与其他基础算法进行对战实验。每次对战实验分别设置 DMC-ResNet 智能体作为地主和农民角色。为了消除随机发放的手牌等运气因素影响，对于每次初始发牌的牌局都安排 DMC-ResNet 智能体和对战基础算法轮流作为地主和农民。随机将 51 张手牌随机分发给三位玩家，首先指定 DMC-ResNet 智能体为地主角色并分发余下的三张公共牌，其中此时的 DMC-ResNet 智能体为训练中地主角色的智能体，对战的基准算法作为两个农民角色的决策算法。随后保持初始的发牌记录，指定 DMC-ResNet 智能体为农民角色，其中此时的 DMC-ResNet 智能体为训练中两个农民角色的智能体，对战的基准算法作为地主角色的决策算法。每次对战实验都进行 10,000 局的比赛后统计 DMC-ResNet 智能体胜率和 ADP，其中 5,000 局 DMC-ResNet 作为地主，另外 5,000 局 DMC-ResNet 作为农民。

实验中采用 CPU 和 GPU 混合计算，其中演员对战部分使用 CPU 进行计算，具体设备为英特尔至强金牌 5222（3.9GHz Turbo,10.4GT/s 2UPI16.5MB 缓存），内存为 128GB (2x64GB)DDR4 3200MHz RDIMM ECC 和 2.5” 512GB SATA Class 20 的固态盘。而学习者学习部分使用 GPU 进行计算，具体设备为 RTX 3090，搭载了 10496 个流处理器与 24G GDDR6X 显存，带宽为 936GB/S。

在对战中，如果算法一在对战算法二时胜率大于 50%，则说明算法一的对战胜率水平大于算法二。如果算法一在对战算法二时 ADP 大于 0，则说明算法一的实战得分水平大于算法二。不同算法在地主和农民角色上的实战水平也有比较大的差异，需要利用胜率和 ADP 共同作为衡量算法智能性的标准。在本文中，考虑到胜率这一指标更符合传统的胜负判断，将作为主要参考指标和模型的训练目标，而 ADP 将用于辅助分析策略的进攻性和保守性。

3.3.2.1 与随机算法比较

随机算法的斗地主算法的主要步骤是根据玩家的手牌和上次玩家的出牌记录，根据斗地主游戏中对牌型大小的规则设定，找到所有可以打出的牌型，（需要注意的是“过”也是可以打出牌型的一种），并从中随机选出一种作为最终的出牌动作。

DMC-ResNet 智能体作为地主对战随机算法的胜率如图3.9所示，DMC-ResNet 智能体作为农民对战随机算法的胜率变化如图3.10所示。

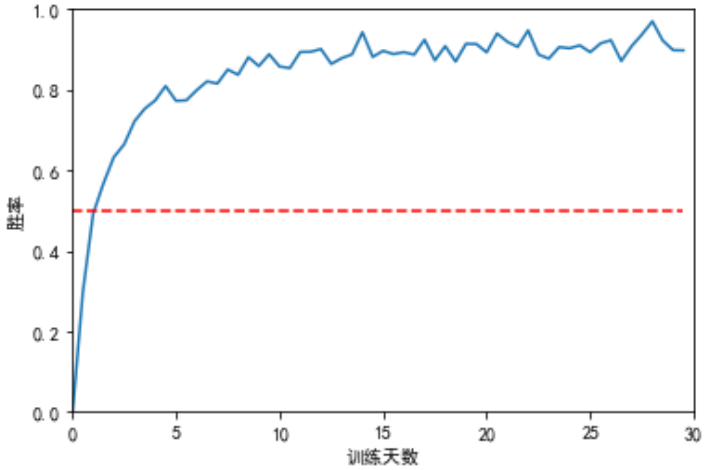

图 3.9 DMC-ResNet 地主对战随机算法农民的胜率

Figure 3.9 WP of DMC-ResNet Landlord against Random

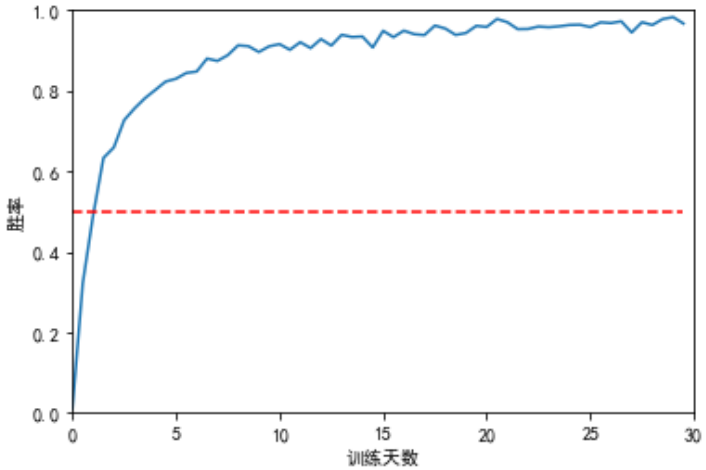

图 3.10 DMC-ResNet 农民对战随机算法地主的胜率

Figure 3.10 WP of DMC-ResNet Peasants against Random

从图中可以看出深度蒙特卡洛模型在训练一天后不论是地主还是农民角色，

就可以打败随机算法, 并且在一周内胜率达到稳定。随着训练过程，该模型作为地主和农民胜率都很快的得到提升，其中农民角色先于地主角色达到稳定，并且在经过 30 天的训练后农民角色的胜率明显高于地主角色，最终 DMC-ResNet 智能体作为农民的胜率为 0.983，而 DMC-ResNet 智能体作为地主角色的胜率出现了较大的不稳定性，在训练的第 27 天的模型胜率最高可以达到 0.971，但是在第三十天的模型胜率又下降为 0.955。

DMC-ResNet 智能体作为地主对战随机算法的 ADP 如图3.11所示，DMC-ResNet 智能体作为农民对战随机算法的 ADP 如图3.12所示。

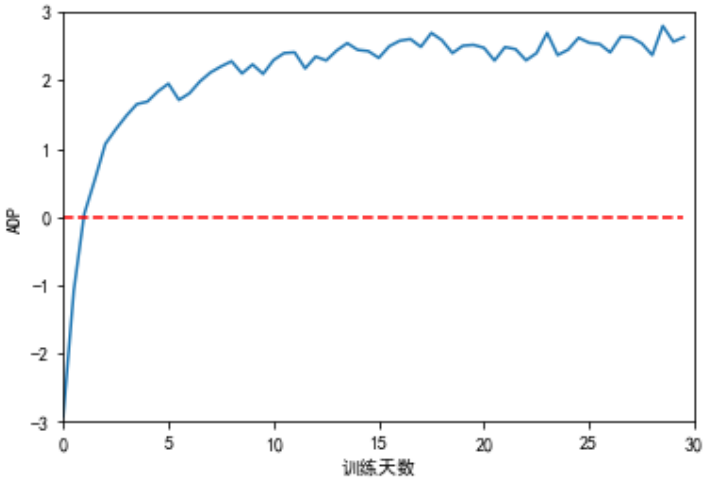

图 3.11 DMC-ResNet 地主对战随机算法农民的 ADP

Figure 3.11 ADP of DMC-ResNet Landlord against Random

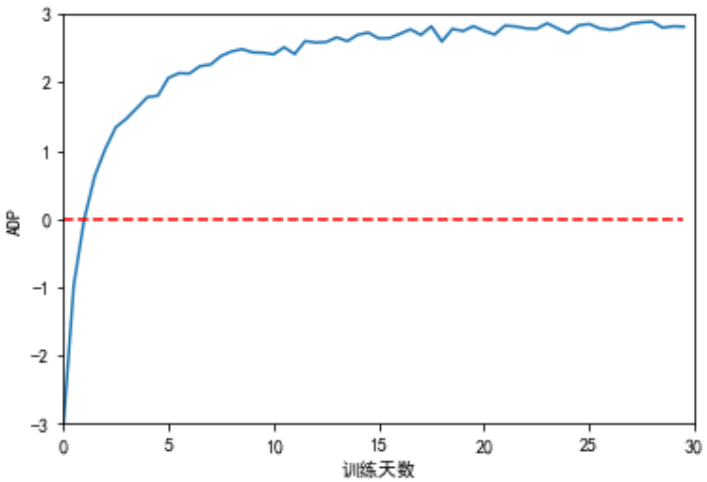

图 3.12 DMC-ResNet 农民对战随机算法地主的 ADP

Figure 3.12 ADP of DMC-ResNet Peasants against Random

和胜率的情形相似，从上图可以看出 DMC-ResNet 智能体作为农民和地主在对战随机算法时都可以很快获得优势地位，在 ADP 上，模型经过五天左右的训练就可惜显著战胜对手，这主要是由于随机算法的智能水平较低，可以被模型很快的利用其出牌方式。作为农民的 DMC-ResNet 智能体在对战地主角色的随机算法的 ADP 在三十天的训练后基本可以稳定在 2.910 左右，而 DMC-ResNet 智能体作为地主时的 ADP 在第二十八天时达到最高位 2.888，在第三十天时的 ADP 为 2.851。在训练后期时 DMC-ResNet 智能体作为地主角色时的 ADP 不如农民角色稳定。

出现上述实验现象是因为对手作为随机算法，出牌的随机性会直接影响到 DMC-ResNet 智能体发挥的稳定性，但是由于随机算法过于简单，所以整体的胜率和 ADP 都较高。同时因为地主和农民角色在游戏中的不对等地位，使得 DMC-ResNet 智能体在地主角色上的表现不如农民角色稳定。

3.3.2.2 与 RHCP 算法比较

RHCP 算法是网络中开源的斗地主算法，该方法主要基于专家经验的规则知识库对手牌的状态进行价值评估，即在考虑玩家手牌和上一个玩家的出牌后，找到当前状态下的所有出牌方式，并从中选取出打完该牌型后剩余手牌状态中价值最高的即为最终牌型。需要注意的是，手牌的拆分主要靠人类玩家的经验，比如在出现可以将手牌拆分为两个部分时，就优先打出牌型最大的，抢先获得下一轮的优先出牌权，如果无人出牌后就可以成为最后面的胜利手牌。与上面的实验设置类似，进行 10,000 局的比赛后统计 DMC-ResNet 智能体胜率和 ADP，其中 5,000 局 DMC-ResNet 智能体作为地主，另外 5,000 局 RHCP 智能体作为地主。

DMC-ResNet 智能体作为地主对战 RHCP 农民的胜率如图3.13所示，DMC-ResNet 智能体作为农民对战 RHCP 地主的胜率如图3.14所示。

从上图中可以发现，深度蒙特卡洛模型在训练两天后就可以战胜 RHCP 算法，并且地主角色的胜率在十天内达到稳定，而农民的角色相比较可以更快的在五天后稳定。在经过一个月的训练后，地主的胜率和农民的胜率差别不明显，最终地主的胜率为 0.887，而农民角色的胜率为 0.911。和随机算法相比，RHCP 算法因为基于规则和人类经验，出牌方式更加稳定，因此在训练后的胜率相比较于随机算法也更加稳定。

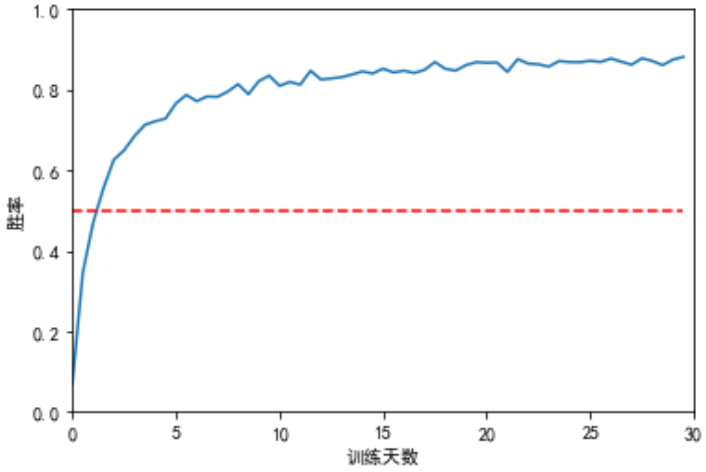

图 3.13 DMC-ResNet 地主对战 RHCP 算法农民的胜率

Figure 3.13 WP of DMC-ResNet Landlord against RHCP

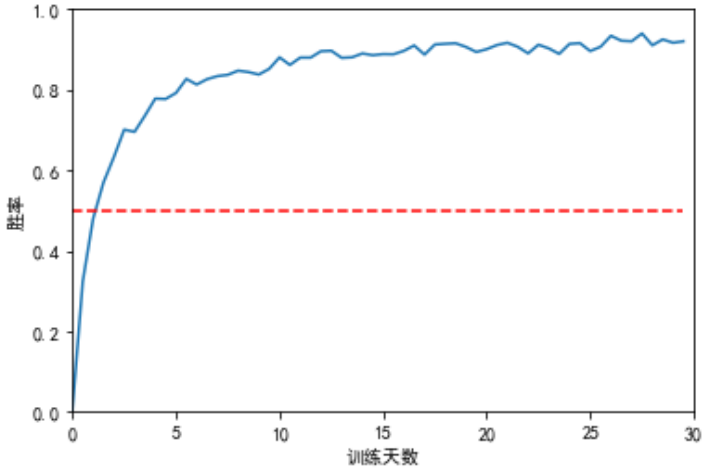

图 3.14 DMC-ResNet 农民对战 RHCP 算法地主的胜率

Figure 3.14 WP of DMC-ResNet Peasants against RHCP

DMC-ResNet 智能体作为地主对战 RHCP 农民的 ADP 如图3.15所示，DMC-ResNet 智能体作为农民对战 RHCP 地主的 ADP 如图3.16所示。

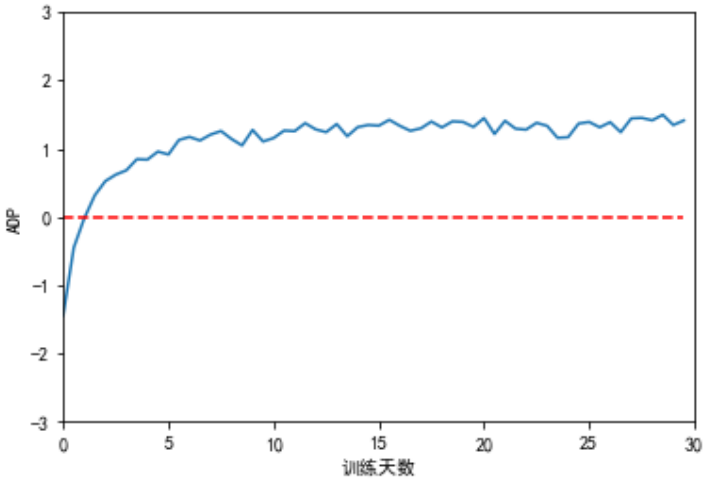

图 3.15 DMC-ResNet 地主对战 RHCP 算法农民的 ADP

Figure 3.15 ADP of DMC-ResNet Landlord against RHCP

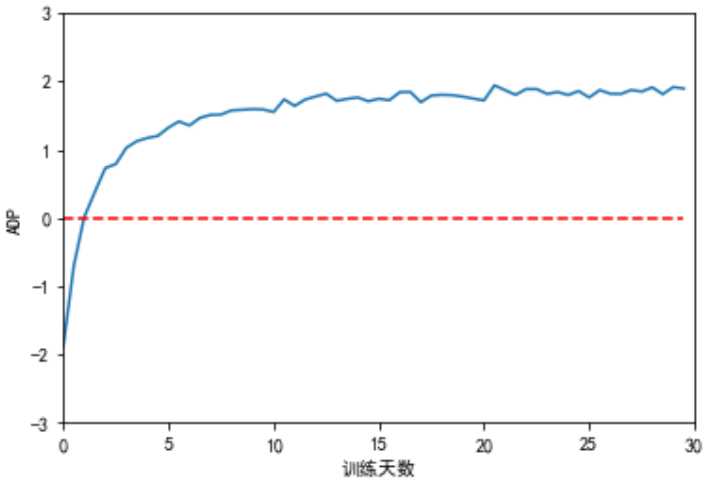

图 3.16 DMC-ResNet 农民对战 RHCP 算法地主的 ADP

Figure 3.16 ADP of DMC-ResNet Peasants against RHCP

和对战 RHCP 算法的胜率相同，在 ADP 上该模型的农民和地主角色都很快的战胜了 RHCP 算法。深度蒙特卡洛模型作为地主对战 RHCP 算法的农民最终 ADP 稳定在 2.009，作为农民对战 RHCP 算法的地主最终 ADP 是 1.441。

与胜率相比，可以看到虽然农民和地主角色的胜率差别不大，但是两者的 ADP 却有明显差别，其中农民的 ADP 大于地主的 ADP。这主要是因为本文研究均利用胜率作为最终的训练目标，这样的设置更符合实际生活中的游戏目标。

3.3.2.3　与 CQL 算法比较

You 等 (2020) 所提出的 CQL 算法是基于手牌拆分的斗地主算法，对手牌采样组合学习的方式，启发式的模仿人类玩家的决策过程，利用两段式的网络结构来获得最终的动作：手牌的组合建议网络和最终的动作决策网络。CQL 算法可以极大的降低动作空间的规模，并且因为网络中的池化层可以很好的保留了动作特征间的关联。该算法在与简单的深度强化学习如 DQL、A3C 等方法的对局中取得了很好的成绩。

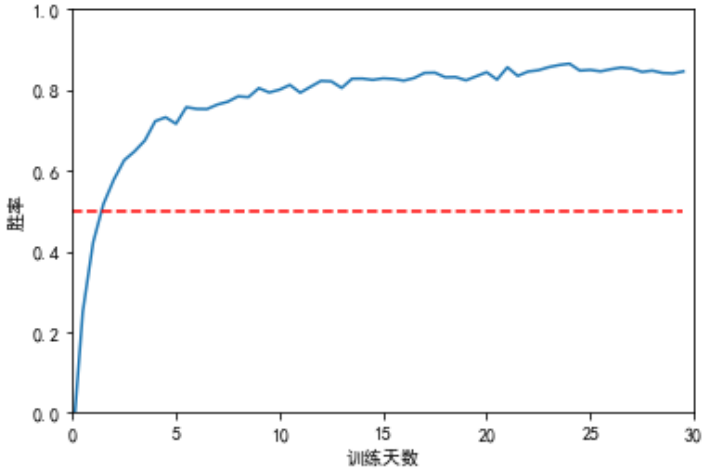

图 3.17 DMC-ResNet 地主对战 CQL 算法农民的胜率

Figure 3.17 WP of DMC-ResNet Landlord against CQL

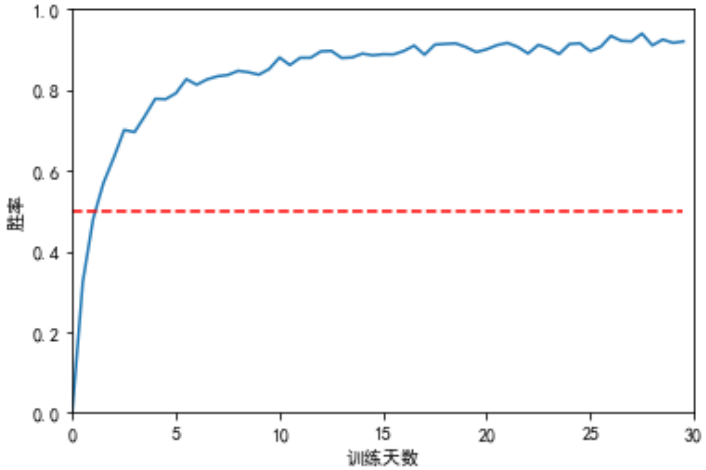

图 3.18 DMC-ResNet 农民对战 CQL 算法地主的胜率

Figure 3.18 WP of DMC-ResNet Peasants against CQL

深度蒙特卡洛方法作为地主对战 CQL 算法农民的胜率如图3.17所示，深度

蒙特卡洛方法作为农民对战 CQL 算法地主的胜率如图3.18所示。

从上图可以看到，深度蒙特卡洛模型的胜率在三天内就超过了 CQL 算法，并在一周内达到了稳定，其中农民角色的胜率稳定在 0.863，而地主角色的胜率不如农民角色稳定且出现在 0.885 附近。可以看到深度蒙特卡洛算法在地主上的表现依旧不如农民角色，整体对战 CQL 算法的胜率为 0.874。

深度蒙特卡洛方法作为地主对战 CQL 算法农民的 ADP 如图3.19所示，深度蒙特卡洛方法作为农民对战 CQL 算法地主的 ADP 如图3.20所示。

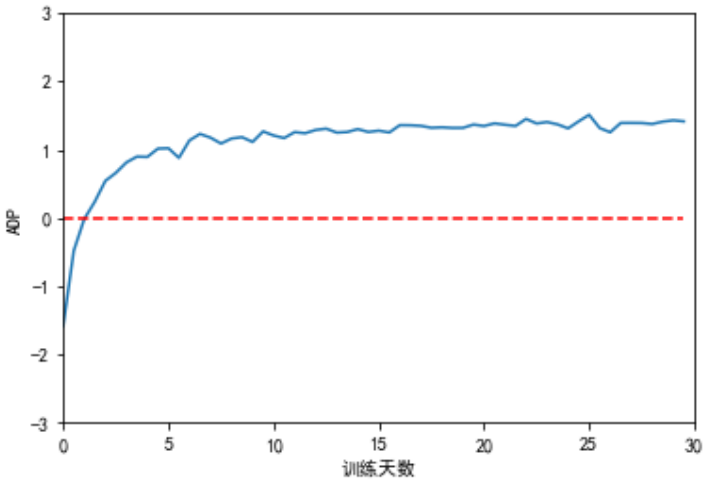

图 3.19 DMC-ResNet 地主对战 CQL 算法农民的 ADP

Figure 3.19 ADP of DMC-ResNet Landlord against CQL

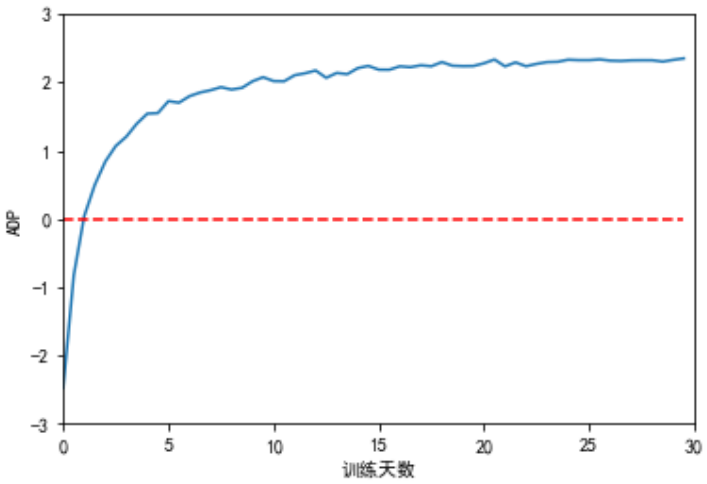

图 3.20 DMC-ResNet 农民对战 CQL 算法地主的 ADP

Figure 3.20 ADP of DMC-ResNet Peasants against CQL

在 ADP 方面，DMC-ResNet 智能体在地主和农民上经过两天左右的训练就

可以战胜 CQL 算法，并且在五天内 ADP 保持了稳定。其中 DMC-ResNet 智能体的地主 ADP 经过三十天的训练达到 1.368，而农民角色的 ADP 最终为 2.001，农民角色的在 ADP 上的表现也明显好于地主角色。

3.3.2.4　与 DouZero 算法比较

Zha 等 (2021) 提出的 DouZero 算法是目前在斗地主游戏上领先的智能算法，该方法提出了基于蒙特卡洛算法框架对样本进行采样学习。同时该方法在知名机器博弈游戏网站 Botzone的天梯积分排行榜上取得了优秀的成绩，并且在和相关的基础算法比较中也保持着领先的地位，比如 DeltaDou 系统、RHCP 系统和 CQL 系统等。

为了更好的和 DouZero 系统做比较，列出了 DMC-ResNet 和 DouZero 在对战不同算法的胜率和 ADP。其中为了保持对比实验的有效性，均选择训练时间为 30 天的模型。为了实验的精确性，利用最终的训练模型增加对战的局数。每次对战实验都进行 100,000 局的比赛，之后统计 DMC-ResNet 智能体胜率和 ADP，其中 50,000 局 DMC-ResNet 作为地主，另外 50,000 局 DMC-ResNet 作为农民。其中的 DouZero 的胜率和 ADP 均为原有文章中的实验数据 (Zha 等, 2021)。表格中表示横轴的算法和纵轴的算法进行实战时的 WP 和 ADP，表格中 P 代表农民（Peasants），L 代表地主（Landlord），P 后的数字表示横轴算法为农民纵轴算法为地主时的数据，L 后的数字表示横轴算法为地主纵轴算法为农民时的数据。

DMC-ResNet 与 DouZero 对战的胜率比较如3.5所示，从表 3.5可以看到，首先在与 DouZero 的直接对局中，DMC-ResNet 的综合胜率为 0.544 大于 50%，整体在胜率上超过了 DouZero 的对战水平。其中作为农民角色的胜率为 0.633，这说明了当 DMC-ResNet 为农民时的智能水平是高于 DouZero 的地主的。而当 DMC-ResNet 智能体为地主时的胜率为 0.454，这说明此时 DMC-ResNet 地主的实战水平是低于 DouZero 中的农民的。这主要是因为在斗地主游戏中地主角色的玩家其实为弱势的一方，他的出牌选择需要更加保守。但是在获胜后一般得到的奖励为两个农民失去的奖励，其风险与收益是对等。在对阵其他基础算法时，虽然在对阵随机算法时的胜率稍微存在了下降，但是与其他算法对战时可以看到胜率都有了一定的提升，其中在对战 RHCP 算法时的效果最为明显，直接从 0.764 提升到 0.899。此外可以看到在农民角色上的提升效果要优于地主角色。

与 DouZero 对战的 ADP 比较见表 3.6。从 ADP 的对比表中可以看到，在对

表 3.5 与 DouZero 比较胜率

Table 3.5 Winning Percentage comparing with DouZero

胜率	DouZero	CQL	RHCP	随机
DouZero	0.500 (P0.584/L0.416)	0.810 (P0.851/L0.769)	0.764 (P0.803/L0.725)	0.989 (P0.992/L0.986)
DMC-ResNet	0.544 (P0.633/L0.454)	0.874 (P0.863/L0.885)	0.899 (P0.911/L0.887)	0.983 (P0.988/L0.978)

表 3.6 与 DouZero 比较 ADP

Table 3.6 Average Difference in Points comparing with DouZero

ADP	DouZero	CQL	RHCP	随机
DouZero	–	1.685 (P2.001/L1.368)	1.671 (P1.850/L1.492)	3.036 (P2.818/L3.254)
DMC-ResNet	-0.393 (P0.122/L-0.908)	2.112 (P2.881/L1.342)	1.725 (P2.009/L1.441)	2.968 (P2.941/L2.995)

阵其他基础算法时，虽然在对阵随机算法时的 ADP 出现了下降情况，这主要是因为随机算法不能很好的预测和利用玩家的出牌方式，使得 DMC-ResNet 智能体无法有效利用得分情况。但是在其他算法上可以看到胜率都有了一定的提升，其中在对战 CQL 算法时的效果最为明显，直接从 1.685 提升到 2.112。此外可以看到在农民角色上的 ADP 提升效果也同样要优于地主角色，这主要是因为在农民角色上算法加入了残局规则的对战方法，这使得农民学习的样本信息量要优于 DouZero 系统的参数靠共享缓存区通信以实时更新来直接对战。

但是实验发现如果使用 ADP 作为奖励，算法将倾向于非常活跃地出炸弹，策略具有更强的攻击性。这是因为玩炸弹有直接的得分激励，但是就不能保证游戏最终可以获胜。而在本次实验中以胜率为目标，智能体会更谨慎地出炸弹，因为炸弹会影响胜率，在做决定时只需要考虑胜率情况。直观来看，利用 ADP 作为训练的目标的智能体在 ADP 方面的表现更好，反之如同本次与 RHCP 对战实验所展示的一样，基于胜率作为训练的目标的智能体在胜率方面的表现比以 ADP 为目标训练的更好。

在与 DouZero 的直接对局中，DMC-ResNet 的综合 ADP 为-0.393 小于 0，整体在 ADP 上不如 DouZero 的对战水平。主要是因为其中作为地主角色的 ADP 为-0.908，这说明了当 DMC-ResNet 为地主时的智能水平是低于 DouZero 的地主的。而当 DMC-ResNet 智能体为农民时的 ADP 为 0.122，DMC-ResNet 地主的实战水平以微弱优势高于于 DouZero 中的农民的。反映了地主角色的策略提升在目前的深度蒙特卡洛框架下非常困难，需要对地主角色的智能性进行差异化的提升，对该问题的研究体现在第五章中。

3.3.2.5　性能稳定性比较

算法性能的稳定性可以通过比较收敛后的智能体的胜率方差和 ADP 方差进行分析。为了保持对比实验的有效性，均选择训练时间为三十天的模型。每次对战实验都进行 100,000 局的比赛，其中 50,000 局 DMC-ResNet 作为地主，另外 50,000 局 DMC-ResNet 作为农民，之后统计 DMC-ResNet 智能体胜率和 ADP。将上述实验分为十个对战轮次，计算这十个样本点在胜率和 ADP 上的方差，其中 DouZero 和 DMC-ResNet 算法与其他基准算法对战的胜率方差比较见表 3.7，DouZero 和 DMC-ResNet 算法与其他基准算法对战的 ADP 方差比较见表 3.8。

表 3.7 与 DouZero 比较胜率的方差

Table 3.7 Variance of WP comparing with DouZero

胜率的方差	DouZero	CQL	RHCP	随机
DouZero	0.0179	0.0387	0.0376	0.0667
DMC-ResNet	0.0145	0.0225	0.0281	0.0615
比较	18.99%	41.86%	25.26%	8.46%

表 3.8 与 DouZero 比较 ADP 的方差

Table 3.8 Variance of ADP comparing with DouZero

ADP 的方差	DouZero	CQL	RHCP	随机
DouZero	0.0013	0.0025	0.0024	0.0036
DMC-ResNet	0.0011	0.0021	0.0022	0.0042
比较	15.38%	16.00%	8.33%	−16.67%

从表3.7可以看到，DMC-ResNet 算法在对战其他基准算法的胜率方差相比 DouZero 均有一定的降低，其中与 CQL 算法对战时的提升效果最为明显，达到 41.86%。并且在与 DouZero 算法对战时，DMC-ResNet 算法极大的较低了胜率方差，平均胜率方差在整体上降低了 18.99%。从表3.8可以看到，DMC-ResNet 算法仅在与随机算法对战时的 ADP 方差产生了增大，这主要是应为该算法的目标是基于胜率进行训练，模型无法较好的兼顾胜率和 ADP。但是在与其他算法的对战中，ADP 方差均出现了一定程度的减少，其中在与 DouZero 算法的对战中，DMC-ResNet 算法极大的较低了方差，平均差异得分率的方差在整体上降低了 15.38%。基于上述实验结果，可以得到结合了残差网络的深度蒙特卡洛方法降低了与其他基准算法对战时的胜率和 ADP 方差，提升了算法性能的稳定性。

3.3.2.6 不同演员数量的比较

本章主要针对算法性能不稳定的问题，提出了结合残差网络的深度蒙特卡洛方法。为了加速深度蒙特卡洛方法中产生训练样本的速度，采用了多演员并行计算的方法快速生成大量的历史对局。但是与 DouZero 算法相比，由于计算资源的不同，在训练网络时所使用的样本数量不同，这是因为在本文中采用了 CPU 与 GPU 混合计算的方式，残差网络可以利用多个演员产生历史对局。演员数量的不同代表了计算资源的多少。

其中为了保持对比实验的有效性，最终的模型均选择训练时间为 30 天的模型。为了实验的精确性，利用最终的训练模型增加对战的局数。每次对战实验都进行 100,000 局的比赛，之后统计 DMC-ResNet 智能体胜率和 ADP，其中 50,000 局 DMC-ResNet 智能体作为地主，另外 50,000 局 DMC-ResNet 智能体作为农民。下面比较演员数量不同时该算法的实际性能，分别在对战中设置 15、30、45 个演员，使用上述相同的实验设置，比较算法的胜率和 ADP。

从表3.9可以看到，不同演员数量下的算法胜率相差不大，仅在演员数量为 15 时，与 DouZero 算法对战时的胜率出现了一定程度的下滑，计算资源的丰富性对斗地主算法最终的智能水平影响并不核心关键。从表3.10可以看到，不同演员数量下的算法 ADP 提升效果不明显，仅在演员数量为 15 时，与 RHCP 算法对战时出现了一定程度的下滑。基于上述实验结果，可以得到结合了残差网络的深度蒙特卡洛方法仅在初期可以一定程度的提升对战成绩，但是不同的演员数量中实验效果差别并不明显。

表 3.9 不同数量演员的胜率比较

Table 3.9 Comparing of WP with different numbers of Actor

胜率	DouZero	CQL	RHCP	随机
15	0.511	0.865	0.841	0.985
30	0.541	0.861	0.882	0.988
45	0.544	0.874	0.899	0.983

表 3.10 不同数量演员的 ADP 比较

Table 3.10 Comparing of ADP with different numbers of Actor

ADP	DouZero	CQL	RHCP	随机
15	-0.418	2.012	1.555	2.693
30	-0.409	2.001	1.618	2.881
45	-0.393	2.112	1.725	2.968

3.4　本章小结

本章的工作表明，对于斗地主算法出现的性能不稳定问题，在利用深度蒙特卡洛方法产生大量历史对局记录的条件下，利用基于残差网络的 Q 网络可以提取出更多层次的特征信息，提升模型的鲁棒性和泛化能力。与基准的斗地主算法的对战实验表明，结合残差网络的深度蒙特卡洛方法可以显著的降低算法性能的不稳定性，提高了斗地主算法整体的智能水平，并且以 0.544 的平均胜率上打败了当前最好的 DouZero 算法。

第 4 章　结合随机优先采样的深度蒙特卡洛方法

本章主要介绍结合随机优先采样的深度蒙特卡洛方法，先简单介绍了随机优先采样的基本思想，然后将该方法运用到基于深度蒙特卡洛方法的斗地主算法中，最后对实验结果进行了分析。

4.1　随机优先采样算法

深度强化学习中的稀疏奖励问题一直是学术界关注的关键问题。深度强化学习需要根据动作的奖励值来更新迭代策略，但是因为在策略的初始化时通常会选择随机初始化，因此强化学习算法在初始阶段很难获得正式的奖励结果，使得模型的学习训练很困难。并且在实际生活中的任务往往也存在稀疏奖励问题。以斗地主游戏为例，三位玩家的手牌共有五十四张，需要等到第一位玩家将手牌出完比赛才能结束，此时任务才能获得奖励，有相关的研究表明一局斗地主游戏的三个玩家平均需要一百次左右的动作决策才能生成奖励。因此研究稀疏奖励问题可以提高深度强化学习中训练速度，提高样本的利用率，为最终的策略优化提供便利。

Schaul 等 (2015) 提出了优先经验回放机制（Prioritized experience replay，PER），随后Horgan 等 (2018) 在此基础上也提出了分布式的优先经验回放方法（Distributed Prioritized experience replay，DPER）。在深度强化学习的训练过程中，例如 DQN 算法，需要对经验池中的样本进行采样以用于智能体的训练，这些样本是指智能体和环境交互生成的样本（transition），其中包括了当前状态（state），动作（action），获得的奖励（reward），以及下一步状态（next state）和最终状态（last state）等。但是因为稀疏奖励的问题，大部分样本的并没有获得奖励，所以包含的有利于模型训练的信息量很少，即可以认为 TD-error 很小。

优先经验回放机制的主要思想是：在采样时优先选取信息量较大的样本，这里对于信息量的衡量可以利用 TD-error。TD-error 值越大则表示 Q-target 与 Q-evaluator 之间的价值评估的差距越大，代表模型在该样本上的预测还有很多上升空间，那么这个样本就越需要被学习。

但是如果直接选择 TD-error 最大的样本无法使得模型收敛。因为首先，TD-

error 对于噪声很敏感，容易将源于噪声的估计误差加入。其次，如果每次采样前都需要更新整个经验池，计算量会非常的大，而如果只更新一个 batch 样本的 TD-error，那么初始产生的很小 TD-error 样本在存放到经验池后，会在很长时间都不会被回放采样。最后，容易出现过拟合的情况，因为基于贪心的采样方法会只专注于经验的一小部分，高 TD-error 的样本由于重放频繁会使得样本丧失多样性，导致出现“旱的旱死，涝的涝死”的情况。

因此，随机优先采样算法采样时还需要加入了随机性，样本被抽取的概率与优先级成正比，可以表示为：

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$$

其中 p_i 为样本的优先级，可以按照样本的 TD-error 直接定义：

$$p_i \triangleq \left|\delta_i\right| + \epsilon$$

其中 ϵ 为一个小的正常数，可以防止边缘样本无法被采样到。也可以按照排序进行定义：

$$p_i \triangleq \frac{1}{\text{rank}(i)}$$

其中 rank(i) 表示样本 i 根据 $\left|\delta_i\right|$ 大小排序的排名。这种采样方法不仅可以利用高质量样本，同时所有样本都有机会被采样，有利用增加样本的多样性。其中，按照第一种对样本优先级的定义方式设计优先经验回放机制，样本被抽取的概率可以写为：

$$P(i) = \frac{\left(\left|\delta_i\right| + \varepsilon\right)^\alpha}{\sum_k \left(\left|\delta_k\right| + \varepsilon\right)^\alpha}$$

其中 ε, α 均为约束的常数，而 δ_i 表示样本 i 的 TD-error，该样本的 TD-error 在 t 时刻的定义为：

$$\delta_t \quad \triangleq R_t + \gamma_t \max_a Q\left(S_t, a\right) - Q\left(S_{t-1}, A_{t-1}\right)$$

其中 R_t 为奖励，γ_t 为衰减系数，S_t, S_{t-1} 为 t 和 $t-1$ 时刻的状态，A_{t-1} 为 $t-1$ 时刻的动作。

此外，因为根据优先级对样本进行采样时，相对于均匀抽样改变了样本的实际分布，造成了偏差问题。优先经验回放算法可以在随机梯度下降时按照修正后的权重参数进行更新：

$$\theta \leftarrow \theta + \eta \cdot w_i \delta_t \cdot \nabla_\theta Q\big|_{S_{t-1}, A_{t-1}}$$

，其中的 θ 表示网络的参数，η 为学习率，w_i 为重要性采样权重，其定义为：

$$w_i \triangleq \left(\frac{1}{N} \cdot \frac{1}{P(i)}\right)^{\beta}$$

其中 N 表示样本集的大小，β 为约束常数，采用上述样本优先级定义时的实验中一般设定为 1。使用该重要性采样权重对偏差进行补偿后，梯度下降时就按照补偿后的梯度进行更新。

对偏差的样本权重进行补偿后，优先经验回放被称为随机优先采样。基于随机优先采样算法的深度 Q 学习可以降低智能体与环境的交互次数，提高智能体的最优策略水平，有效的解决了样本利用率低的问题。

4.2　基于随机优先采样的斗地主算法

一般的强化学习算法是利用在经验池中随机采样获得的批量样本更新目标网络，同时将和环境交互得到的新的样本存储到经验池中。而针对稀疏奖励问题，利用随机优先采样可以极大的提高样本的利用率。例如在 DQN 算法上采样随机优先采样算法的大致流程如图 4.1所示。在采样批量样本时需要按照优先级进行随机采样，Q 网络和目标网络再对批量样本的梯度进行更新，而由网络产生的动作会和环境交互而产生新的样本，这些样本需要根据优先级存储到经验池中。其中可以根据批量样本的 TD-error 来更新经验池中的样本优先级，损失函数在一个批量样本下利用随机梯度下降算法来完成模型的训练收敛。

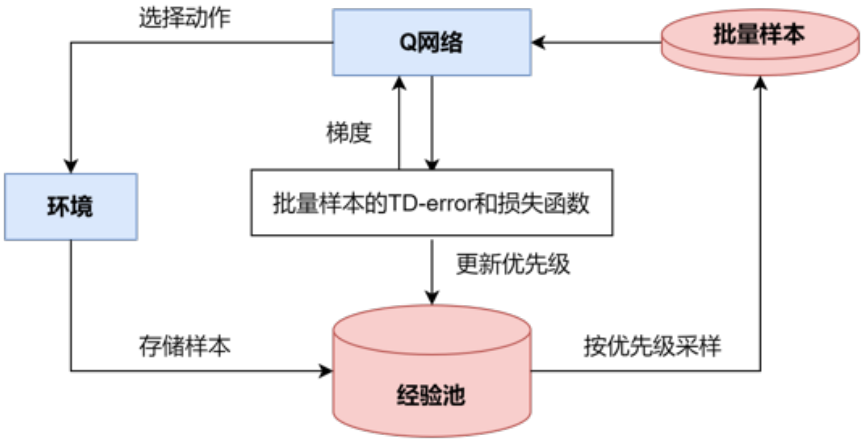

图 4.1 随机优先采样的流程图

Figure 4.1 Flowchart of randomly prioritized sampling

在实际实验中，为了方便存储经验池中的样本和对应的样本优先级，通常使用 SumTree 这样的二叉树结构来作为这种带优先级的经验回放池的存储数据结构。SumTree 是一种树的数据结构，所有的经验回放样本都在叶子节点上，每个树叶节点都存储着每个样本的优先级。内部的节点没有样本数据，每个父节点都有两个子节点，保存的是它们的优先级之和。

而在深度蒙特卡洛方法上使用随机优先采样方法依旧可以采用多演员并行计算的方式，通过利用多个演员（Actor）生成历史对局记录和一个学习者（Learner）训练模型参数。因为多个演员（演员）的网络策略的参数不相同，这就提高了样本的差异性，使学习者能能够充分利用计算资源。同时，演员（Actor）和学习者（Learner）都可以计算经验回放中的样本优先级，这使得经验池中的样本优先级可以得到及时的更新。随机优先采样算法在深度蒙特卡洛上的流程图如 4.2所示。

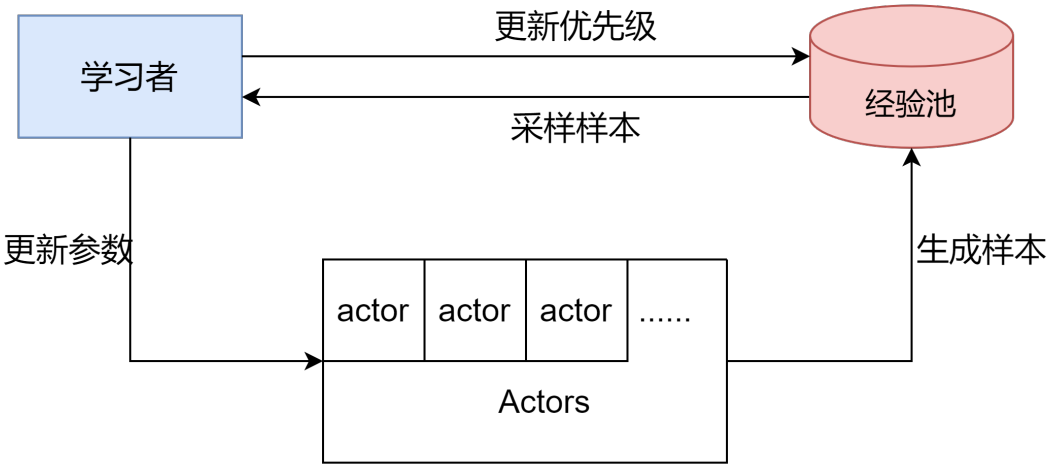

图 4.2 PER 在深度蒙特卡洛上的流程图

Figure 4.2 Flowchart of PER in Deep Monte Carlo

学习者通过对经验池随机优先采样生成的批量样本进行更新，并且可以对经验池中样本的优先级进行更新。多个演员的分布式对局可以快速的生成对战的历史记录作为样本存储到经验池中。学习者需要和演员间进行通信，演员需要及时更新自己的策略参数，并且在新的参数下继续对战生成样本。在实验设置中，学习者为三个斗地主玩家维护三个全局的 Q 网络，并将多个演员在对战中产生的历史记录作为样本，以更新全局网络，损失函数为 MSE。随后根据样本新的 TD-error 来更新样本优先值。每个演员都有三个本地 Q 网络，这些网络定

期与全局网络同步，学习者和演员通过三个共享缓冲区以实现通信。

本文继续利用上一个章所介绍的结合残差网络的深度蒙特卡洛模型作为基础模型，在基于深度蒙特卡洛方法的框架下使用随机优先采样方法对经验池进行采样。在本章中为了方便讲述，将未使用随机优先采样方法的智能体称为 DMC，采用随机优先采样方法的智能体称为 DMC-PER。

经验池是共享的计算缓存区，但是演员不是每生成一条样本就立即更新到经验池中，每个演员都在本地有个缓存区，最后通过批量更新至经验池，否则经验池在计算中会有很大的请求压力。演员将样本添加到经验池后，学习者再从经验池中按照随机优先机制采样数据，并进行学习，演员也会和学习者同步参数。

基于经验回放机制可以显著的提升模型的训练性能，通过利用多个演员并行计算以产生不同的经验样本，计算效率也得到了很好的提升，因为每个演员都不同，在利用 $\epsilon - greedy$ 的方式进行探索利用时可以选择不同的贪心概率参数。这样可以更好的实现探索与利用机制，并且更全面的寻找优先级最高的样本来进行训练。

而整个算法框架可以分为学习者（Learner）和演员（Actor），两者可以分离的分布式运行，本文实验一般在 CPU 上运行演员计算模块，在 GPU 上运行学习者模块。通过多演员进行采样并收集斗地主游戏的历史记录数据，同时当前演员的网络可以直接生成初始样本的优先级。演员的数量肯定会对模型最终的实验效果造成影响，一般而言，演员越多，采集到的斗地主对局记录数据也就越丰富，尤其当不同演员采用不同的策略时，能显著提高系统整体的探索能力。再通过随机优先采样方法从丰富的历史对局数据中采样出更有利于学习的样本，从而提高学习效率和模型性能。同样经验池的规模大小，也会影响系统性能的提升。一般规模越大，优先级较高的样本停留的时间越长，对学习者训练的贡献也越大。而且提高的批样本集的大小也会有利于提高系统的学习效率。实验中具体的参数设置见介绍实验环境的 4.3.1 章节。

需要注意的是，传统的优先级经验回放机制在进行优先级初始化时，需要将最新的样本的优先级直接设置为最大的，从而让新的样本能快速被采样到，然后经过网络训练后，再根据该样本的 TD-error 调整优先级。但是在分布式的随机优先采样下，每一个演员都会产生样本，如果直接初始化都为最大优先级，就导致在学习者选取样本时大部分是最近产生的样本，而过去的样本无法被采样到。

因此，每个演员可以利用自己的策略计算出样本的优先级，再将样本放入经验池中。结合了随机优先采样算法后，此时的演员的运行算法见算法 5，学习者的运行算法见算法 **??**。

算法 5 演员算法

输入: 共享缓存区 $\mathcal{B}_L, \mathcal{B}_U$ 和 $\mathcal{B}_D$ ，容量均为 $L = B \times S$ ，探索利用参数 ϵ，损失系数 γ

初始化本地网络 $Q_{\mathrm{L}}, Q_{\mathrm{U}}$ 和 Q_{D}，以及本地缓存区 $\mathcal{D}_L, \mathcal{D}_U$ 和 $\mathcal{D}_D$
procedure ACTOR(B, T) ▷ 在环境中运行智能体并存储经验
    $\theta_0 \leftarrow$ LEARNER.PARAMETERS() ▷ 远程调用获取最新网络参数
    $s_0 \leftarrow$ ENVIRONMENT.INITIALIZE() ▷ 从环境中获得初始状态
    for $i \leftarrow 1, T$ **do**
        $a_{t-1} \leftarrow \pi_{\theta_{t-1}}\left(s_{t-1}\right)$ ▷ 使用当前策略生成行动
        $\left(r_t, \gamma_t, s_t\right) \leftarrow$ ENVIRONMENT.STEP $\left(a_{t-1}\right)$ ▷ 与环境交互生成样本
        LOCALBUFFER.ADD $\left(\left(s_{t-1}, a_{t-1}, r_t, \gamma_t\right)\right)$ ▷ 增加样本到缓存区中
        if LOCALBUFFER.SIZE( ) $\geq B$ **then** ▷ 定期从经验池中采样
            $\tau \leftarrow$ LOCALBUFFER.GET (B) ▷ 获得缓存样本
            $p \leftarrow$ COMPUTEPRIORITIES (τ) ▷ 计算样本优先级
            REPLAY.ADD (τ, p) ▷ 增加样本到经验池中
        end if
        PERIODICALLY $\left(\theta_t \leftarrow\right.$ LEARNER.PARAMETERS ()) ▷ 获得最新的网络参数
    end for
end procedure

上面主要介绍了结合随机优先采样的深度蒙特卡洛方法的实现原理，下面简要分析使用该方法的优点。基于随机优先采样的深度蒙特卡洛方法是一种 off-policy 学习的数据采样机制，通过提高系统吞吐量来提高学习效率。和以前的方法相比，利用多演员来采集数据可以扩大经验池的规模大小，充分利用了计算资源，可以大幅度加快训练速度。而通过不同的演员得到不同样本优先级的回收，也大幅度提升了模型探索的能力，防止出现过拟合现象。

另外，相比于分布式的求网络梯度，分布式的经验回放可以提升训练效率。这主要是因为梯度信息的时效性比较差，信息量低的演员提供的梯度信息相比于信息量高的演员会在训练后期产生失去价值的训练样本。在初始化多个演员后，由于梯度信息代表的了当前网络更新参数的方向，即使初始化所有演员都采用相同网络，但是如果有的演员训练比较快，那它也会优先于其它演员网络，梯

算法 6 学习者算法

输入： 共享缓存区 $\mathcal{B}_L, \mathcal{B}_U$ 和 $\mathcal{B}_D$ ，容量均为 $L = B \times S$ ，探索利用参数 ϵ，损失系数 γ

初始化本地网络 $Q_{\mathrm{L}}, Q_{\mathrm{U}}$ 和 Q_{D}，以及本地缓存区 $\mathcal{D}_L, \mathcal{D}_U$ 和 $\mathcal{D}_D$
procedure LEARNER(T)　▷ 利用从经验池中采样的经验样本更新网络
    $\theta_0 \leftarrow$ INITIALIZE.NETWORK( )
    for $i \leftarrow 1, T$ **do**
        $id, \tau \leftarrow$ REPLAY.SAMPLE( )　▷ 随机优先采样一个 batch 的样本
        $l_t \leftarrow$ COMPUTE.LOSS($\tau;\theta$)　▷ 运用 DQN 学习算法计算损失函数
        $\theta_{t+1} \leftarrow$ UPDATEPARAMETERS(l_t, θ_t)　▷ 更新网络参数
        $p \leftarrow$COMPUTE.PRIORITIES( )　▷ 利用 TD-error 计算样本优先级
        REPLAY.SETPRIORITY(id, p)　▷ 更新样本优先级
    end for
end procedure

度信息也会导致更新参数的方向存在较大差异。但是基于分布式的经验回放机制可以避免上述问题，经验存储在样本池中，如果一个较差的演员产生了一个信息量高的样本也可以被优先采样到。如果产生了没有价值的经验，该样本的优先级就低，选取该经验的概率就小，即使被选取了用于训练网络，也不会对网络产生错误的影响。

4.3 实验结果与分析

4.3.1 实验环境设置

本次研究的所有实验均在 RLCard 框架的斗地主环境中进行，主要将结合了随机优先采样的深度蒙特卡洛方法的斗地主算法，与依据其他算法进行逐对的实验对战，包括随机出牌的算法、基于规则的 RHCP 算法、基于手牌拆分的 CQL 和最后的 DouZero 算法。将 DMC-PER 算法分别作为地主和农民角色，与其他基础算法进行对战。实验分别选取训练过程中的三十天内的算法智能体，每隔半天就对 DMC-PER 的智能体进行保存并与其他基础算法进行对战实验。对 54 张手牌随机初始化发牌后，首先指定 DMC-PER 智能体为地主角色并分发余下的三张公共牌，其中此时的 DMC-ER 算法为训练中地主角色的智能体，对战的基准算法作为两个农民角色的决策算法。随后保持初始的发牌记录，指定 DMC-PER 算法为农民角色，其中此时的 DMC-PER 算法为训练中两个农民角色的智能体，对

战的基准算法此时则作为地主角色的决策算法，在按照规则进行博弈到结束。每次对战实验都进行 10,000 局的比赛后统计 DMC-PER 算法的胜率和 ADP，其中 5,000 局 DMC-PER 算法作为地主，另外 5,000 局 DMC-PER 算法作为农民。计算 DMC-PER 算法平均胜率和平均差异得分率包括了所有的历史成绩，共为 10,000 局。

为了和上一章节的实验设置做对比，在利用随机优先采样方法时的参数设置为，共享缓存区的大小 $B = 100$，样本集大小为 $S = 100$，batch 大小为 $M = 32$ 并且贪心探索的概率参数为 0.01。设置损失系数 $\gamma = 1$，并且采用 RMSprop 优化器，其中的学习率为 $\psi = 0.0001$ 以及误差参数为 10^{-5}。此外，在随机优先采样中选择利用 TD-error 的结构定义样本的优先级，其中的优先级次幂参数 $\alpha = 1$，即样本优先级与被抽取的概率成正比，边缘样本参数为 0.1。

4.3.2 对战实验结果与分析

4.3.2.1 与随机算法对战

DMC-PER 智能体对战随机算法的胜率比较如图 4.3所示，DMC-PER 智能体对战随机算法的 ADP 如图 4.4所示。

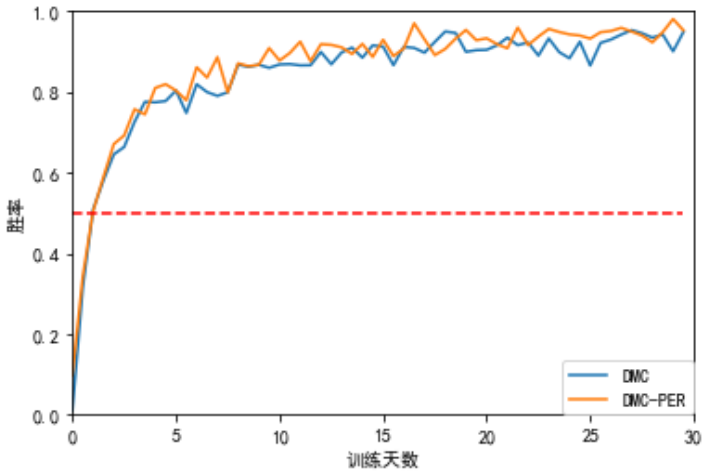

图 4.3 对战随机算法的胜率比较

Figure 4.3 Winning Percentage of DMC-PER and DMC against Random

从图中可以看出 DMC-PER 模型在训练一天就可以在胜率上打败随机算法，并且在五天内胜率达到稳定。与未使用随机优先采样的深度蒙特卡洛方法比较，前期的收敛速度基本一致。在训练三十天后，DMC-PER 模型对战随机出牌算法的胜率达到了 0.987，与 DMC 模型的胜率 0.983 基本相同。这主要是因为随机

出牌算法的智能性很低，在简单的随机采样下，DMC 模型就具有了很高的胜率水平，即使加入随机优先采样算法，胜率的实际提升效果也不明显。而这两个模型在 WP 稳定后的方差都较大，这主要是因为随机算法作为对手时，它的出牌随机性会直接影响到深度蒙特卡洛方法的发挥稳定，但是由于随机算法过于简单，所以整体的胜率和 ADP 都较高。

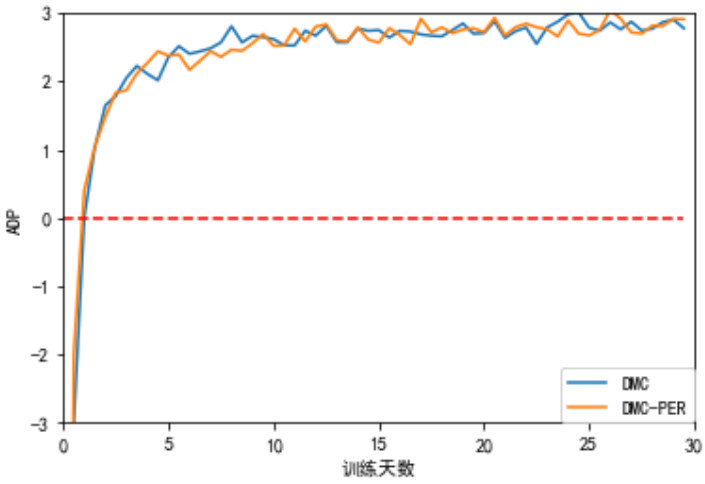

图 4.4 对战随机算法的 ADP 比较

Figure 4.4 Average Difference in Points of DMC-PER and DMC against Random

和上面的胜率部分相似，DMC-PER 模型经过一天左右的训练就可以在 ADP 方面打败随机出牌算法，训练经过一周后，DMC 模型和 DMC-PER 模型的 ADP 都基本稳定。但是可以看到，DMC-PER 模型的 ADP 要稍微高于 DMC 模型，实验数据表明经过三十天的训练后，DMC-PER 智能体的 ADP 为 3.001，这稍微大于 DMC 模型的 2.968。

同时，将 ADP 曲线和胜率曲线比较可以看到，在对战随机算法时，两个深度蒙特卡洛在达到稳定后的 ADP 方差较小，不会出现胜率情况中较为不稳定的情况。这主要是因为将随机算法作为对手时，模型可以较为好的取得分数上的胜利，模型在利用对手失误上较为成功，但是又因为出牌过于随机，在某些不利局面下往往无法完全取得牌局的胜利，使得胜率出现方差较大的情况。

4.3.2.2　与 RHCP 算法对战

和上述实验设置相同，分别将 DMC-PER 算法和 RHCP 算法作为地主和农民对战 5,000 局，再保持相同的手牌状态将角色互换后，继续对局 5,000 局。实验后的综合胜率与 ADP 将上述对战实验取平均所得。DMC-PER 智能体对战 RHCP

算法的胜率如图 4.5所示，DMC-PER 智能体对战 RHCP 算法的 ADP 如图 4.6所示。

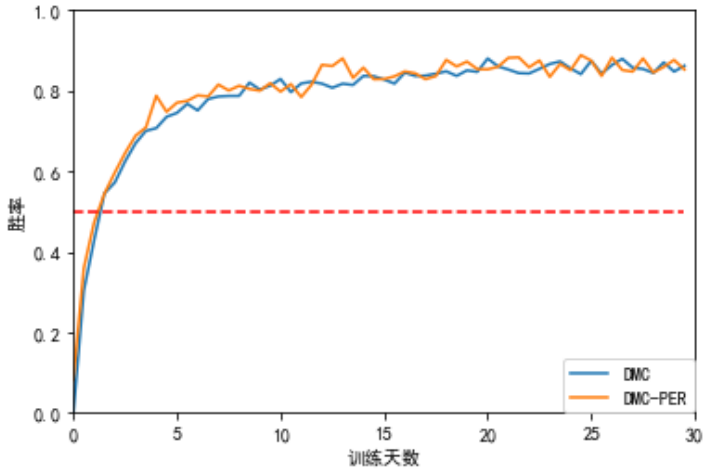

图 4.5 对战 RHCP 算法的胜率比较

Figure 4.5 Winning Percentage of DMC-PER and DMC against RHCP

从图 4.5可以发现，在训练初始的提升阶段，DMC-PER 的胜率要稍微好于 DMC 模型，而在经过 10 天左右的训练后模型趋于稳定，DMC 模型和 DMC-PER 算法的胜率水平也大致相同。实验结果表明在对战基于经验的 RHCP 时，经过三十天的训练后 DMC-PER 智能体的胜率为 0.856，和 DMC 模型的 0.899 基本相同。比较对战 RHCP 时的胜率，随机优先采样算法可以在训练初期提升模型的效果，但是无法提升最终模型的智能水平。

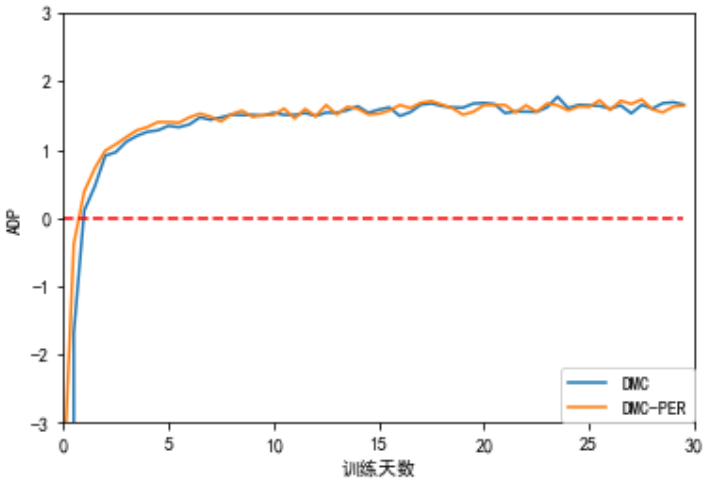

图 4.6 对战 RHCP 算法的 ADP 比较

Figure 4.6 Average Difference in Points of DMC-PER and DMC against RHCP

上图表明在 ADP 方面和 WP 的情况相似，PER 算法都可以在训练提升阶段改善 DMC-PER 算法的智能性，这主要是因为在训练初期时，随机优先采样方法可以将信息量更高的样本优先提供给学习者网络，从而提升了实际对战中的成绩。在训练稳定后，因为经验样本基本同水平，模型的能力受到限制，此时 DMC 和 DMC-PER 的实际对战水平基本相同。

4.3.2.3　与 CQL 算法对战

将 DMC-PER 算法和 CQL 算法作为地主和农民对战 5,000 局，再保持相同的手牌状态将角色互换后，继续对局 5,000 局。实验后的综合胜率与 ADP 将上述对战实验取平均所得。DMC-PER 智能体对战 CQL 算法的胜率如图 4.7所示，DMC-PER 智能体对战 CQL 算法的 ADP 如图 4.8所示。

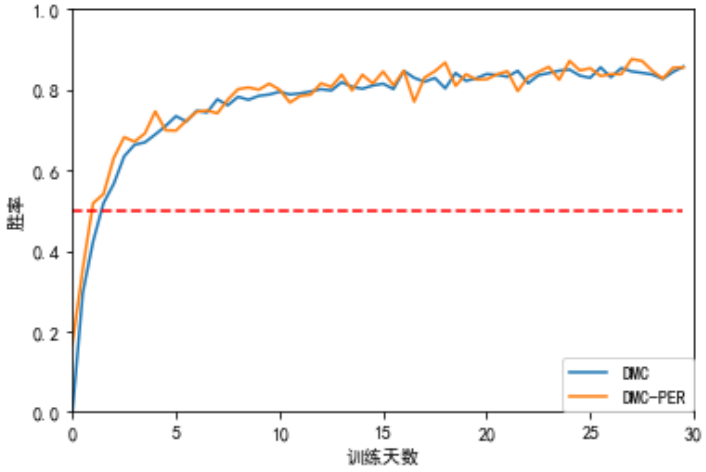

图 4.7 对战 CQL 算法的胜率比较

Figure 4.7 Winning Percentage of DMC-PER and DMC against CQL

从图 4.7可以看到，在训练初始的提升阶段，DMC-PER 的胜率要明显好于 DMC 模型，而在经过十天左右的训练后模型趋于稳定，DMC 模型和 DMC-PER 算法在 WP 水平上基本一致。三十天后，在对战基于 DQN 算法的 CQL 时，DMC-PER 智能体的胜率为 0.772，小于 DMC 模型的 0.874。比较对战 CQL 时的胜率，PER 算法可以在训练初期提升深度蒙特卡洛模型采样的样本质量，但是无法提升模型的最终智能水平。

从图 4.8中可以看到，DMC-PER 的 ADP 在训练初始阶段要好于 DMC 模型，DMC-PER 智能体也先于 DMC 模型在 ADP 上战胜 CQL 算法，而在经过十天左右的训练后模型趋于稳定。不同于 WP 情况的是，在对战基于 DQN 算法的 CQL

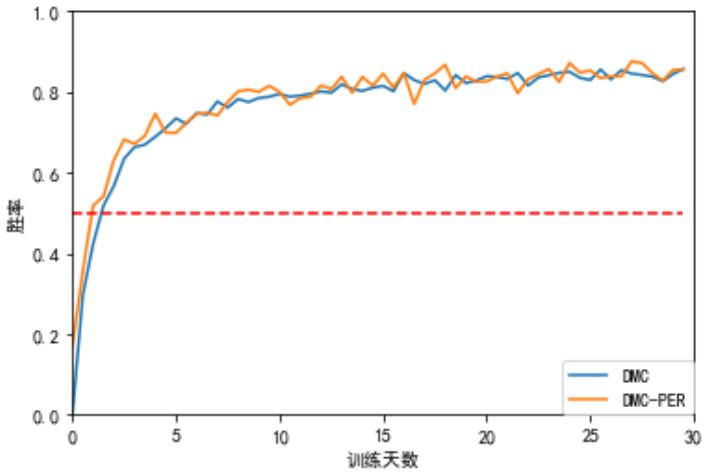

图 4.8 对战 CQL 算法的 ADP 比较

Figure 4.8 Average Difference in Points of DMC-PER and DMC against CQL

时，DMC-PER 智能体的 ADP 为 2.609 大于 DMC 模型的 2.117。在对战 CQL 时的 ADP 情况下，PER 算法可以在训练初期提升深度蒙特卡洛模型采样的样本质量，也提升了斗地主算法的实际对战效果。

4.3.2.4 与 DouZero 算法对战

DouZero 算法同样基于深度蒙特卡洛方法，将 DMC-PER 算法和 DouZero 算法对战 10,000 局，其中保持相同的手牌状态互换位置，分别作为地主和农民各 5,000 局。DMC-PER 对战 DouZero 算法的胜率比较如图4.9所示，DMC-PER 对战 DouZero 算法的 ADP 比较如图4.10所示。

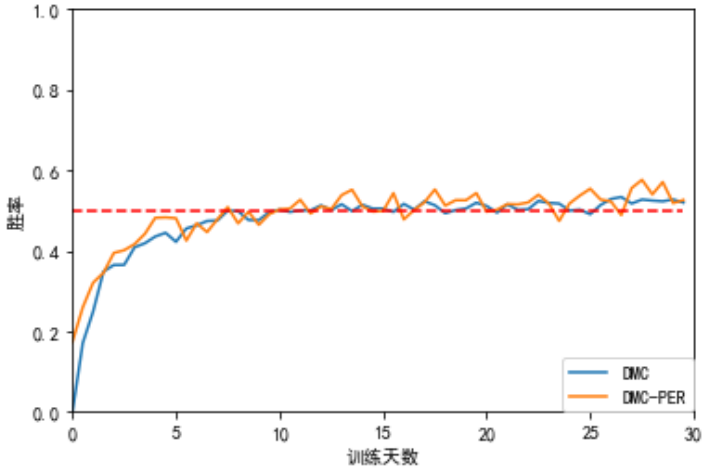

图 4.9 对战 DouZero 算法的胜率比较

Figure 4.9 Winning Percentage of DMC-PER and DMC against DouZero

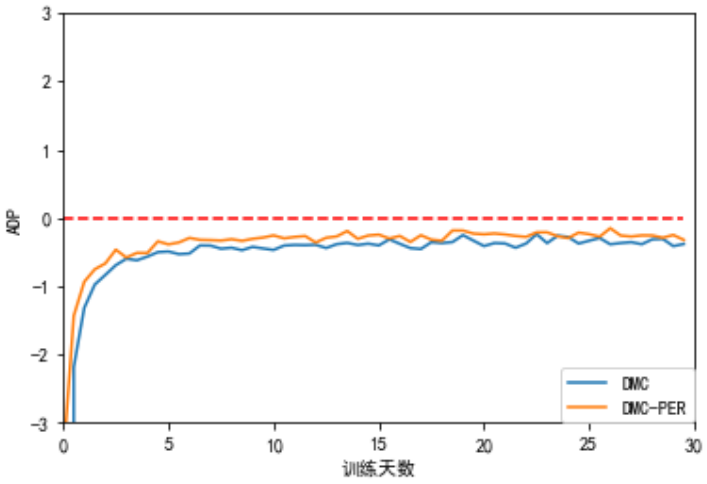

图 4.10 对战 DouZero 算法的 ADP 比较

Figure 4.10 Average Difference in Points of DMC-PER and DMC against DouZero

从图4.9可以看出，在训练开始阶段，DMC-PER 的胜率要明显好于 DMC 模型，而在经过十天左右的训练后，模型的 WP 水平趋于稳定，DMC-PER 算法要稍微好于 DMC。最终的模型，在对战 DouZero 算法时，DMC-PER 智能体的胜率为 0.541，和 DMC 模型的 0.544 基本相同。DMC-PER 模型在对战 DouZero 时的胜率可以在训练初期好于 DMC 的随机采样，但是无法提升模型的最终智能水平。

从图4.10中可以看出上一章中提到的问题，在 ADP 上 DMC 算法无法战胜 DouZero 算法，主要是因为以胜率为训练目标的 DMC 算法，在地主角色上的 ADP 表现较差，无法完全战胜 DouZero 算法中的农民。但是可以看到，在利用随机优先采样算法后，DMC-PER 模型的 ADP 水平有了较为明显的提升。

4.3.2.5　收敛时间与算法性能比较

将基于深度蒙特卡洛方法的斗地主算法做农民和地主角色的 WP 和 ADP 综合比较，见下表 4.1。其中的 DouZero 的数据来源于Zha 等 (2021) 中的实验数据。表格表示横轴的算法 A 和纵轴的算法 B 进行比较实验时的综合胜率和 ADP，P 为农民（Peasants），L 为地主（Landlord），P 后的数字表示算法 A 为农民，算法 B 为地主时的胜率或 ADP，L 后的数字表示算法 A 为地主，算法 B 为农民时的胜率或 ADP。DouZero-PER 表示采用了随机优先采样的 DouZero 算法，收敛时间定义为胜率达到 0.5 或者 ADP 达到 0 所需要的训练时间。收敛时间的比较定义为 DMC-PER 算法相比于 DMC 算法在收敛时间上减少的比例，若为正数则代

表训练的收敛时间出现了下降，若为负数则代表 PER 算法提高了训练的收敛时间。

为了实验的精确性，每次比较实验的局数扩大到上述实验的十倍，每次对战实验都进行 100,000 局的比赛后统计智能体的胜率和 ADP，其中 50,000 局 DMC-PER 智能体作为地主，另外 50,000 局 DMC-PER 智能体作为农民。表格中的收敛时间是指胜率达到 50% 以及 ADP 达到 0 时，模型训练所需要的时间。收敛时间比较是指 DMC-PER 在对战不同算法时达到上述要求所需要的时间，相比于 DMC 模型所需要的时间缩减了多少比例。

从表 4.1中可以发现，通过比较不同采样形式下最终训练出的智能体，在胜率上的提升并不明显，甚至在对阵 RHCP 算法和 CQL 算法时出现了小幅下降的情况。反而在 ADP 上，采用随机优先采样的方法可以得到一定的效果提升，只有在对阵 CQL 算法时的 ADP 效果出现了下降。DMC 算法相比 DouZero-PER 算法的实战性能较弱，说明在使用随机优先采样下，残差网络可以提升算法性能。在 ADP 上，虽然在胜率上会出现部分对战实验的结果出现了一定程度上的下降，但是 DMC-PER 相比 DMC 有了比较明显的提升，并且模型收敛到打败对战算法的时间相对减少。

这主要是因为不论随机优先采样方法还是随机采样方法，学习者经过长时间的训练基本都可以遍历到经验池中的全局样本情况，差别仅在采样时的顺序不同。在采用 PER 算法时，训练前期可以优先选出质量较高的样本提供给学习者网络，这就可以提升学习者的智能水平。但是经过长时间的训练，多演员所产生的样本都已经存储到了经验池中，网络经过采样也基本学习完了所有的样本，由 WP 为目标训练的模型在最终胜率上的提升并不明显。但是采样顺序的不同可以影响智能体在得分能力上的表现，利用 PER 算法的斗地主算法如 DouZero-PER 和 DMC-PER 算法，可以更好的利用当前牌局的状态，最终在 ADP 上有了理想的提升。

在收敛时间的比较上，仅在与随机算法的对战中，收敛时间出现了小幅的提升。但是，在与如 DouZero 算法、CQL 算法、RHCP 算法等对战时，在具体的实际收敛时间上，采用随机优先采样的斗地主算法在胜率达到 0.5 的收敛时间上大约可以减少四到五个小时左右，算法最终在五天内到达稳定，相比于 DMC 算法则提前一到两天。在与其余的基准算法对战时，DMC-PER 算法相比于 DMC

表 4.1 收敛时间与算法性能比较

Table 4.1 Comparison of Convergence Time and Agent Performance

胜率	DMC	DouZero	CQL	RHCP	随机
DouZero-PER	——	0.511 (P0.611/L0.410)	0.810 (P0.769/L0.851)	0.764 (P0.725/L0.803)	0.989 (P0.986/0.992)
DMC	——	0.544 (P0.633/L0.454)	0.874 (P0.885/L0.863)	0.899 (P0.911/L0.887)	0.983 (P0.988/L0.978)
DMC-PER	0.502 (P0.510/L0.494)	0.541 (P0.641/L0.441)	0.772 (P0.791/L0.753)	0.856 (P0.888/L0.824)	0.987 (P0.989/L0.985)
收敛时间比较	——	10.55%	12.10%	14.75%	-0.51%
ADP	DMC	DouZero	CQL	RHCP	随机
DouZero-PER	——	-0.177 (P0.003/L-0.357)	1.644 (P2.033/L1.411)	1.670 (P1.800/L1.540)	3.036 (P2.818/L3.254)
DMC	——	-0.303 (P-0.297/L-0.308)	2.112 (P2.881/L1.342)	1.725 (P2.009/L1.441)	2.968 (P2.941/L2.995)
DMC-PER	-0.110 (P0.140/L-0.360)	−0.121 (P0.091/L−0.333)	1.685 (P2.001/L1.368)	1.671 (P1.850/L1.492)	3.010 (P2.110/L3.910)
收敛时间比较	——	10.01%	23.53%	24.75%	-1.31%

算法的收敛时间均出现了减少的现象，其中胜率的收敛时间均减少了 10% 以上，而 ADP 的收敛时间则较少的更为明显，其中在与 RHCP 算法的对战中，收敛时间减少了 24.75%。

4.4 本章小结

本章主要介绍了结合了随机优先采样的深度蒙特卡洛方法，本文工作表明对于样本利用率低的问题，可以在采样中加入随机优先采样方法，通过 TD-error 来定义采样环节中样本的优先级。同时，由于随机优先采样在采样时也具有随机性，所有样本都有机会被采样，也增加了样本的多样性。结合了随机优先采样的深度蒙特卡洛方法，在保持智能体的对战性能基本不变的情况下，显著地减少了模型的收敛时间，算法的收敛时间减少了 10% 以上，提升了样本的利用率。

第 5 章　结合多体学习的深度蒙特卡洛方法

本章主要介绍了结合多体学习的深度蒙特卡洛方法。针对斗地主游戏中地主角色智能水平较低的问题，采用多体强化学习算法对地主智能体进行单独训练，提升了地主决策时的智能水平，最后分析了地主智能体的对战结果。

5.1　多智能体强化学习

深度强化学习技术最近在机器博弈研究领域取得了非常理想的成绩，目前的研究前沿已经从单智能体系统延伸到多智能体系统（Multi-Agent System，MAS）。在实际任务中，环境通常需要和多个智能体进行交互，这需要研究包含多智能体的复杂系统构建，以及每个个体独立行为的相互协调机制。多智能体强化学习是目前人工智能领域最为重要和活跃的研究领域之一。

多智能体系统在环境中具有自身的行动过程，每个智能体有自己的目标、感知方法、动作集等。多智能体与环境交互时包括了获取观测、做出动作、改变环境状态、获取奖励的四个步骤，并且在系统进入下一个状态后继续重复上述过程。最终使得多智能体系统可以通过动作协调完成系统设置的目标。其中环境的定义包括了状态空间、动作空间、状态上的动作效果以及可观测信息集等。

强化学习的核心思想是不断“试错”，智能体通过与环境交互，根据获得的反馈信息迭代优化。在强化学习中，待解决的问题通常被描述为马尔科夫决策过程（Markov Decision Process，MDP）。当同时存在多个智能体与环境交互时，整个系统就变成一个多智能体系统。每个智能体仍然是遵循着强化学习的目标，也就是是最大化能够获得的累积回报，而此时环境全局状态的改变就和所有智能体的联合动作相关了。因此在多智能体强化学习的过程中，需要考虑联合动作的影响。

一个多智能体系统（Multi-Agent System，MAS）的形式化建模还可以概括为：n 个智能体 $G=\{g_1,g_2,\ldots,g_n\}$，智能体的行为过程使用 MDP 表示 $\{(S_t,r_t,a_t)\}_{t=1}^{T}$，使用神经网络学习决策过程，网络参数为 $\Theta=\{\theta_1,\theta_2,\ldots,\theta_n\}$。

单体智能学习的过程可以概括为以下步骤:

(1) 智能体 g_1 独立学习，目标是最大化自身奖励；

(2) 网络参数更新，使得智能体 g_1 获得最大奖励；

(3) 依次对所有智能体 $G=\{g_1,g_2,\ldots,g_n\}$ 重复上述步骤，直到多智能系统最终收敛到次优状态；

多体智能学习的过程可以概括为以下步骤:

(1) 多个智能体构建群体博弈：对多个智能体进行分组，每个智能体的目标是最大化团队奖励；

(2) 求解纳什均衡，智能体 g_1 得到团队目标；

(3) 智能体 g_1 开始学习，目标是最大化团队目标；

(4) 网络参数更新，智能体 g_1 取最大团队奖励。

(5) 依次对所有智能体 $G=\{g_1,g_2,\ldots,g_n\}$ 重复上述步骤，直到多智能系统最终收敛到相对较优状态；

利用深度 Q 学习作为网络的训练算法，多体学习的训练过程见算法 7。

算法 7 多体智能学习

输入: 多智能体 $G=\{g_1,g_2,\ldots,g_n\}$，$\mathrm{MDP}\{(S_t,r_t,a_t)\}_{t=1}^{T}$，网络初始参数 Θ

输出: 网络参数 Θ^*。

1: **for** $t=1,\ldots,T$ **do**

2: $\{\tilde{a}_1,\ldots,\tilde{a}_n\}=\mathrm{Nash}(S_t,g_1,\ldots,g_n)$; ▷ 求解纳什均衡

3: **for** $i=1,\ldots,n$ **do** ▷ 在纳什均衡条件下更新网络

4:

$$Q\left(S_t,a_i\mid\{\tilde{a}_j\}_{j\neq i}\right)=E_{P\left(S_{t+1}\mid S_t,a_i,\{\tilde{a}_j\}_{j\neq i}\right)}r\left(S_{t+1}\mid S_t\right)+\max_a Q^{\theta}\left(S_{t+1},a\right)$$

5: 更新网络参数: $\theta^*=\underset{\theta}{\mathrm{argmin}}\left(\mathrm{Q}\left(S_t,a_i\mid\{\tilde{a}_j\}_{j\neq i}\right)-Q^{\theta}\left(S_t,a_i\right)\right)^2$

6: 选择动作: $a_i^*=\underset{a_i}{\mathrm{argmax}}Q\left(S_t,a_i\mid\{\tilde{a}_j\}_{j\neq i}\right)$

7: **end for**

8: $S_t\times\{a_1^*,\ldots,a_n^*\}\rightarrow S_{t+1}$

9: **end for**

多智能体强化学习的发展过程主要是从单体智能学习发展到多体智能学习。单体智能学习是指多智能体系统中的每一个体都仅从自身目标出发，独立学习使得自身奖励最大化，并且在学习过程中与其他智能体没有交互。这种方法适

用于简单且目标简单的非合作任务，训练的过程中只需要关注单独个体的学习，易于模型收敛。但是单体学习只能完成有限任务，算法无法达到全局最优。多体学习则是通过交互学习完成较复杂任务，多智能体系统中的每一个智能体在学习中和其他智能体存在信息交互，每个智能体在学习时需要考虑其他智能体的当前动作。多体学习通过中心态势分配交互信息来实现多智能体间的协同学习，训练过程中信息有交互机制，最终通过多智能体的合作实现多体协同。

依据多体学习的思想，在两个玩家的零和博弈中，可以得到极大极小 Q 学习（Minmax Q-Learning，MinmaxQ）(Littman, 1994)，其中的 Minmax 指的是使用极大极小方法来求解纳什均衡策略，每个特定状态下的阶段博弈都是两人零和博弈，可以通过线性规划来求解。Q 学习指的是使用 Q-learning 来迭代学习状态值函数或动作-状态对的值函数。两个智能体 i 的状态值函数定义为：

$$V_i^*(s) = \max_{\pi_i(s,\cdot)} \min_{a_{-i} \in A_{-i}} \sum_{a_i \in A_i} Q_i^* \left(s, a_i, a_{-i}\right) \pi_i \left(s, a_i\right), i = 1, 2$$

其中 $-i$ 表示智能体 i 的对手，$Q_i^* \left(s, a_i, a_{-i}\right)$ 为联结动作状态值函数，需要使用 Q-learning 中的时间差分来更新逼近真实的 $Q_i \left(s, a_i, a_{-i}\right)$。在线性规划求解的纳什均衡状态下，更新智能体 i 的 Q 值时采用：

$$Q_i \left(s, a_i\right) \leftarrow Q_i \left(s, a_i\right) + \alpha \left[\mathrm{r}_i + \gamma V_i \left(\mathrm{s}'\right) - Q_i \left(s, a_i\right)\right]$$

将上述方法从两人零和博弈拓展到多人一般和博弈，多体学习算法也可以叫做纳什 Q 学习算法（Nash Q-learning，NashQ）(Hu 等, 2003)。在极大极小 Q 学习算法中需要通过线性规划来求解两人零和博弈的纳什均衡，扩展到纳什 Q 学习算法就是使用二次规划来求解多人的纳什均衡。并且研究人员已经证明纳什 Q 学习算法具有收敛性，可以在合作性博弈或对抗性博弈中收敛到纳什均衡解。不同于极大极小 Q 学习，纳什 Q 学习算法在训练中需要求解二次规划 (Greenwald 等, 2003)，这会使得计算过程非常耗时，降低了学习速度。

在多智能体强化学习中，衡量算法有两个主要的技术指标：合理性与收敛性。合理性是指在针对当前对手的既定策略，算法能够学习并收敛到一个最大化剥削对手的策略。收敛性是指在算法能够在理论上保证可以学习并收敛到一个稳定的策略，通常情况下，该稳定的策略为纳什均衡策略。极大极小 Q 算法和纳什 Q 算法都只能在理论上保证可以收敛到纳什均衡策略，但不具有合理性。

5.2 基于多体学习的斗地主算法

斗地主游戏是属于多人的一般性混合博弈，玩家间同时存在合作和竞争关系，并且每个玩家是自私的，最终拥有不同的奖励。但是三个玩家可以分为两组：地主和两个农民，这两组玩家可以构成零和博弈。由此可以利用上述的极大极小 Q 算法对斗地主中的农民和地主玩家分别进行训练。该方法具有以下优点：

(一) 如果直接对三个玩家使用 Nash Q-Learning 算法，需要不断求解二次规划，这会增加计算时间。而利用分组后极大极小 Q 算法则只需要求解线性规划，提高了计算速度。

(二) 在运行算法时，地主智能体需要知道所有农民智能体的动作空间。基于深度蒙特卡洛方法的斗地主算法在训练时刚好可以分为 Learner 和 Actor 两个部分，三个玩家的智能体在学习时都在 Learner 上，可以直接相互通信。

(三) 极大极小 Q 算法具有收敛性。极大极小 Q 学习算法是对手的独立算法，不论对手的策略如何，都可以收敛到当前状态下阶段博弈的纳什均衡解上。

但是极大极小 Q 算法只能理论上保证在两人零和博弈时收敛到纳什均衡策略，但不能根据其他智能体的策略来优化完善自身的策略。而在本文中，将地主与农民分组形成两组零和博弈，但是两个农民的策略为联合策略，无法达到纳什均衡，只有地主可以达到纳什均衡。

并且极大极小 Q 学习算法不具有合理利用对手的性质，假设对手不按照理性的纳什均衡策略，而是一个智能水平较差的策略，则当前玩家并不能根据对手的策略最大化剥削对手。根据之前的实验，地主和农民由于地位不对等会导致智能水平存在差异，斗地主算法中农民角色的胜率明显高于地主角色。在斗地主游戏中地主处于保守劣势的一方，而农民获得胜利的可能性更高。

这就说明在上述研究的基础上使用极大极小 Q 算法，只能通过胜率水平更高的农民角色来提升地主的水平，反之极大极小 Q 算法在理论上无法利用地主智能体对农民进行提升。因此，为了更好的提升斗地主模型的智能水平，就需要单独提升地主的智能水平。同时在基于深度蒙特卡洛方法训练斗地主算法时，为了降低多智能体训练的难度，本章也只考虑如何继续提升结合了残差网络和随机优先采样的斗地主算法中地主角色的智能水平。设置上一章实验中的最终智能体作为地主和农民的预训练模型，并基于极大极小 Q 学习算法，针对性地提升地主的实际对战水平。

在基于深度蒙特卡洛方法的框架下，继续使用多演员并行计算以产生对战历史作为训练样本，其中农民角色的网络固定使用上文中的 DMC-PER 智能体，以下为了更好地比较就简称为 DMC。结合了多体学习算法后，此时 Actor 的运行算法见算法 8，同时网络的学习则需要利用极大极小 Q 算法，Learner 的运行算法见算法 9，其中在更新 Q 网络参数时的方式使用算法7。对于结合多体学习的斗地主网络，仅选择使用地主网络，对两个联合的农民网络进行舍弃，而使用上一章节的 DMC-PER 算法中的两个独立的农民。

算法 8 Actor 算法

输入： 共享缓存区 $\mathcal{B}_L, \mathcal{B}_U$ 和 $\mathcal{B}_D$，容量均为 $L = B \times S$，探索利用参数 ϵ，损失系数 γ

初始化本地地主网络 Q_{L} 以及本地缓存区 $\mathcal{D}_L, \mathcal{D}_U$ 和 $\mathcal{D}_D$，将 DMC 生成的农民模型固定为 Q_{U} 和 Q_{D}
for $iteration = 1, 2, \ldots$ 到收敛 **do**
    同步地主网络参数 Q_{L}
    for $t = 1, 2, \ldots, T$ **do**
        $Q \leftarrow Q_{\mathrm{L}}, Q_{\mathrm{U}}, Q_{\mathrm{D}}$ ▷ 生成 episode
        生成动作 $a_t \leftarrow \begin{cases} \arg\max_a Q\left(s_t, a\right) & \text{概率为}(1-\epsilon) \\ \text{随机选取行动} & \text{概率为}\epsilon \end{cases}$
        生成下一步状态 s_{t+1}，奖励 r_t
        在 $\mathcal{D}_L, \mathcal{D}_U$ 以及 $\mathcal{D}_D$ 中储存 $\left\{s_t, a_t, r_t\right\}$
    end for
    for $t = T-1, T-2, \ldots, 1$ **do** ▷ 获得累积奖励
        $r_t \leftarrow r_t + \gamma r_{t+1}$ 并更新 $\mathcal{D}_L, \mathcal{D}_U$ 和 $\mathcal{D}_D$ 中的 r_t
    end for
    for $p \in \{L, U, D\}$ **do** ▷ 多进程优化
        if $\mathcal{D}_p$.length $\geq L$ **then**
            请求并等待一个空的缓存区 $\mathcal{B}_p$;
            从 $\mathcal{D}_p$ 移动到 $\mathcal{B}_p$ 大小为 L 个的 $\left\{s_t, a_t, r_t\right\}$;
        end if
    end for
end for

算法 9 Learner 算法

输入： 共享缓存区 $\mathcal{B}_L, \mathcal{B}_U$ 和 $\mathcal{B}_D$ ，每个容量为 $L = B \times S$、Batch 大小 M、学习率 ψ

初始化本地网络 Q_{L}^{g}，对于 Q_{U}^{g} 和 Q_{D}^{g} 固定使用 DMC 训练好的农民网络
for *iteration* $= 1, 2, \ldots$ 到收敛 **do**
    $\left\{\tilde{a}_L, \tilde{a}_U, \tilde{a}_D\right\} = \mathrm{Nash}\left(S_t, Q_{\mathrm{L}}^{g}, (Q_{\mathrm{U}}^{g}, Q_{\mathrm{D}}^{g})\right)$; ▷ 求解当前局势下两组零和博弈的纳什均衡
    for $p \in \{L, (U, D)\}$ **do**
        if $\mathcal{B}_p \geq L$ **then**
            从 $\mathcal{B}_p$ 中采样一个 batch 样本 $\left\{s_t, a_t, r_t\right\}$，大小为 $L = B \times S$
            利用 MSE 损失函数和学习率 ψ 来更新参数 Q_{p}^{g}
        end if
    end for
end for

5.3 实验结果与分析

5.3.1 实验环境设置

本次研究主要将结合了多体学习算法和深度蒙特卡洛方法的斗地主算法，与依据其他算法进行逐对的实验对战，包括随机出牌的算法、基于规则的 RHCP 算法、基于手牌拆分的 CQL 和 DMC 算法。实验分别选取训练过程中的三十天内的算法智能体，每隔半天就对 DMC-Nash 的智能体进行保存并与其他基础算法进行对战实验。在与基准算法对战时，先指定 DMC-Nash 智能体为地主并分发余下的三张公共牌，随后保持初始的发牌记录，再指定基准算法也作为地主。每次对战实验都进行 10,000 局的比赛，统计 DMC-Nash 算法作为地主。然后在原有发牌的状态下再进行 10,000 局对战，此时基准算法作为地主。

为了和上一章节的实验设置做对比，在利用多体学习方法时的参数设置为，共享缓存区的 $B = 100$，样本集大小为 $S = 100$，batch 大小 $M = 32$ 并且探索与利用参数为 0.01。设置损失系数 $\gamma = 1$，在求解纳什均衡时使用的是线性规划求解器 SciPy Optimization and Root Finding，在训练网络时采用 RMSprop 优化器，学习率为 $\psi = 0.0001$，误差参数为 10^{-5}。

本文实验中采用 CPU 和 GPU 混合计算，其中演员对战部分使用 CPU 进行计算，具体设备为英特尔至强金牌 5222（3.9GHz Turbo,10.4GT/s 2UPI16.5MB 缓存）。而 Learner 学习部分使用 GPU 进行计算，具体设备为 RTX 3090。

5.3.2　对战实验结果与分析

5.3.2.1　与随机算法对战

多体学习训练下对战随机算法的地主胜率比较如图5.1所示，多智能体强化学习训练下对战随机算法的地主 ADP 如图5.2所示。

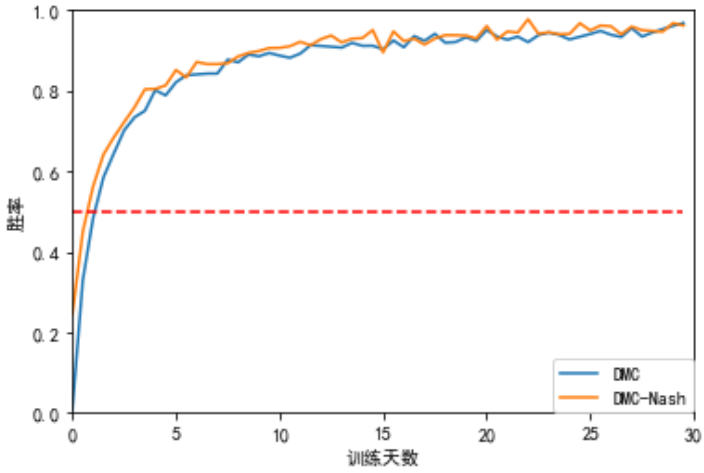

图 5.1 对战随机算法的地主胜率比较

Figure 5.1 Comparison of the Landlord agent's WP against Random

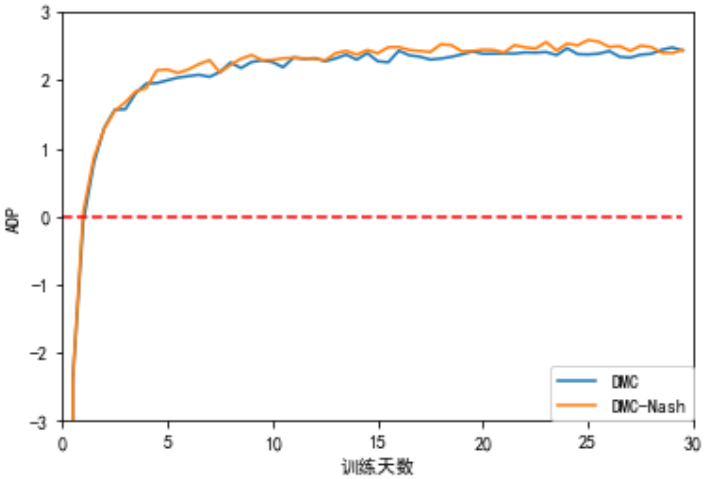

图 5.2 对战随机算法的地主 ADP 比较

Figure 5.2 Comparison of the Landlord agent's ADP against Random

从图5.1可以看到，在对战随机算法时，通过基于合作的多体智能算法所学习的 DMC-Nash，在作为地主智能体时的胜率初始变化不大，只有在稳定期稍微高于单体学习的斗地主算法，这主要是因为在取得较高胜率时，多体学习算法的提升效果不明显。

从图5.2可以看出，与胜率情形相似，多体学习训练下的地主智能体在对战随机算法时 ADP 的提升效果也并不明显，说明地主智能体在面对随机算法时具有压倒性的优势，无法明显地提升智能水平。

5.3.2.2 与 RHCP 算法对战

与上述实验设置相似，每次对战实验先进行 10,000 局的比赛后统计 DMC-Nash 算法作为地主的对战记录，然后在原有发牌的状态下再进行 10,000 局，其中 RHCP 算法作为地主。多体学习训练下对战 RHCP 算法的地主胜率如图所示5.3，多体学习训练下对战 RHCP 算法的地主 ADP 如图所示5.4。

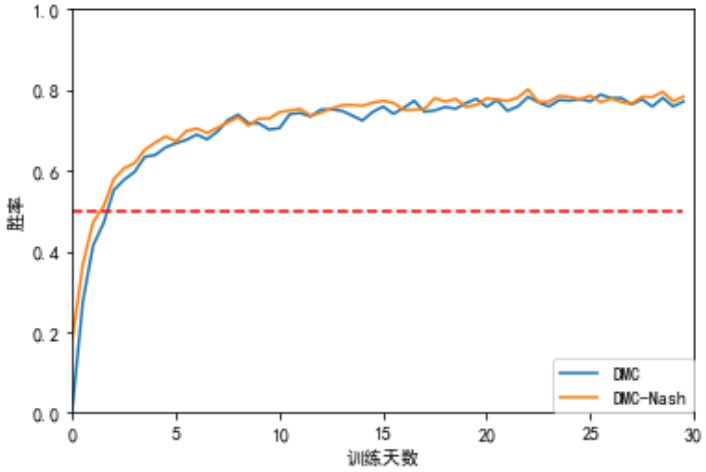

图 5.3 对战 RHCP 算法的地主胜率比较

Figure 5.3 Comparison of the Landlord agent's WP against RHCP

从图中可以看到，在对战 RHCP 算法时，通过基于多体学习的 DMC-Nash 作为地主智能体时的胜率和 ADP 都有一定的提升，实验数据表明在训练稳定后，DMC-Nash 作为地主的胜率从 0.887 提升为 0.892。在 ADP 方面，DMC-Nash 作为地主的 ADP 从 1.441 提升到了 1.933。

从图中还可以看到，基于合作的多体智能算法训练下，地主智能体的 ADP 相比于单体训练的提升速度更快，实验数据表明在对战 RHCP 时，多体学习训练下的地主智能体的 ADP 提前一天就可以大于零，并且在五天内性能达到稳定的状态，但是在胜率上获得稳定所需要的时间要更久一点，这主要是因为本次实验所选取的训练目标为胜率，智能体需要对游戏的胜负做更多的采样训练以提升胜率。

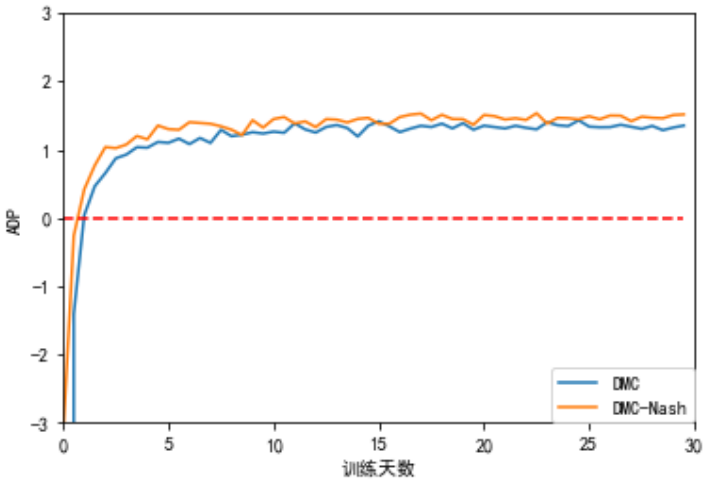

图 5.4 对战 RHCP 算法的地主 ADP 比较

Figure 5.4 Comparison of the Landlord agent's ADP against RHCP

5.3.2.3　与 CQL 算法对战

多体学习训练下对战 CQL 算法的地主胜率率如图5.5所示，多体学习训练下对战 CQL 算法的地主 ADP 如图5.6所示。

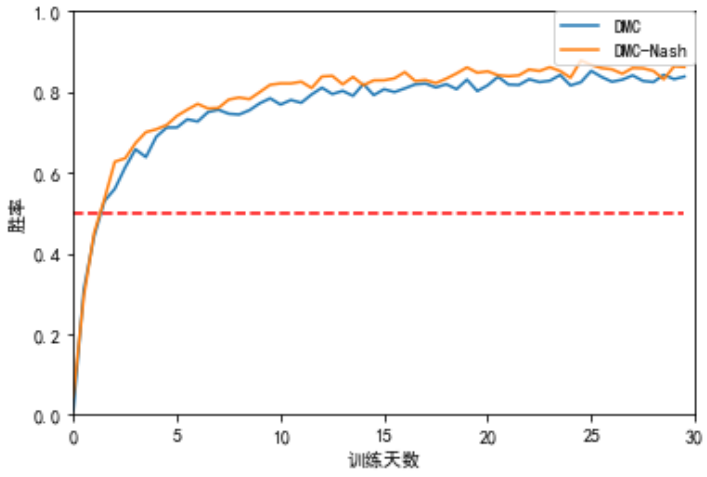

图 5.5 对战 CQL 算法的地主胜率比较

Figure 5.5 Comparison of the Landlord agent's WP against CQL

从图中可以看到，在对战 CQL 算法时，DMC-Nash 模块作为地主时的胜率基本相同，实验数据表明在训练稳定后，DMC-Nash 作为地主的胜率从 0.863 提升为 0.870。然而在 ADP 方面，DMC-Nash 的提升则更加明显，从 1.342 提升到了 1.490。

根据实验数据可以得到，多体学习训练下的地主智能体对战时可在两天内

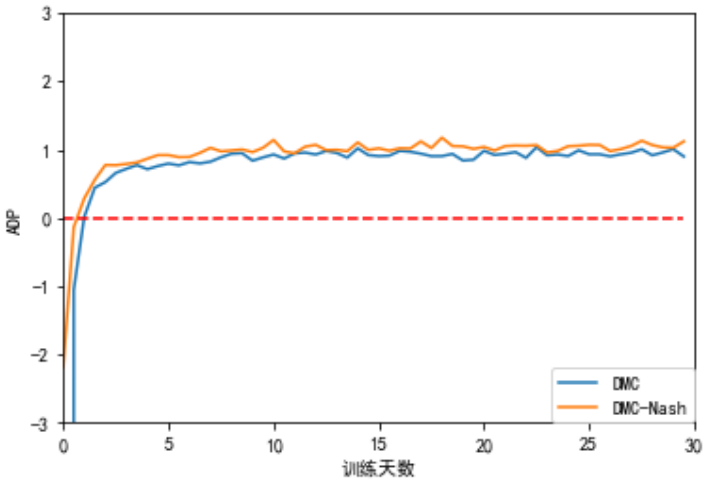

图 5.6 对战 CQL 算法的地主 ADP 比较

Figure 5.6 Comparison of the Landlord agent's ADP against CQL

在胜率上打败 CQL 算法，胜率在十天时达到稳定，这相比于单体学习的 DMC 智能体更慢，这主要是由于在训练中需要对当前的状态求纳什均衡，虽然是求解线性规划但也需要一定的时间。

5.3.2.4 与 DouZero 算法对战

多体学习训练下对战 DouZero 算法的地主胜率如图 5.7所示，对战 DouZero 算法的地主 ADP 如图 5.8所示。

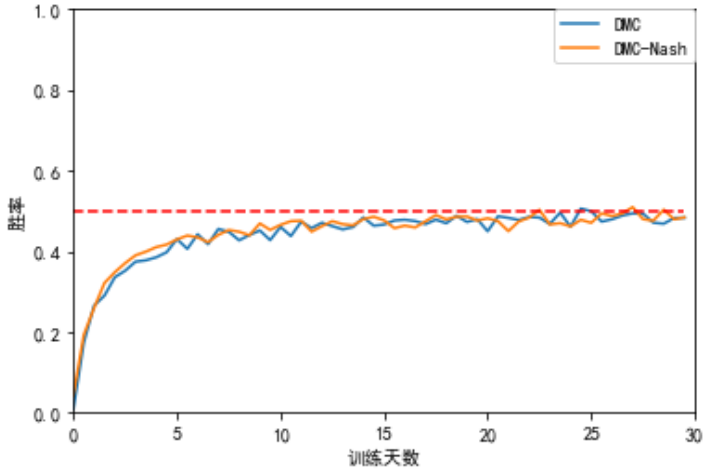

图 5.7 对战 DouZero 算法的地主胜率比较

Figure 5.7 Comparison of the Landlord agent's WP against DouZero

从图中可以看到，在对战 DouZero 算法时，DMC-Nash 算法作为地主时的胜率有一定的提升，实验数据表明在训练稳定后，DMC-Nash 作为地主的胜率从

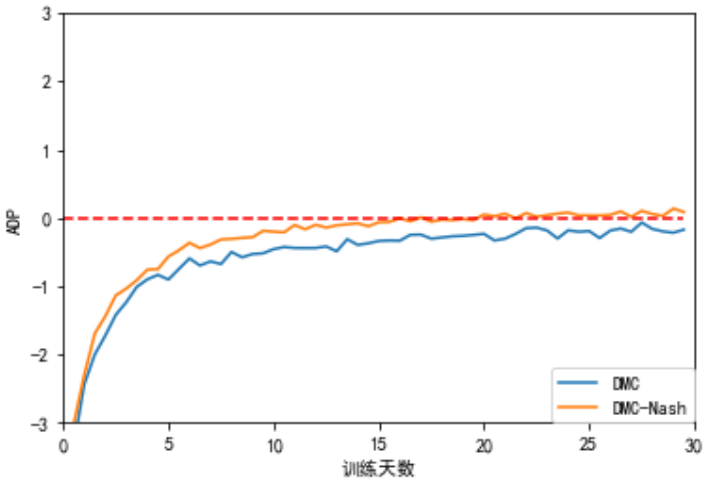

图 5.8 对战 DouZero 算法的地主 ADP 比较

Figure 5.8 Comparison of the Landlord agent's ADP against DouZero

0.454 提升为 0.483。虽然还无法在胜率上直接打败 DouZero 的农民智能体，但是相比于原先的 DouZero 地主智能体在对战农民时的胜率有了一定的提升。在 ADP 方面，DMC-Nash 的提升则更加明显，从-0.308 提升到了 0.081，首次实现了在对战 DouZero 算法时，地主角色在 ADP 方面的胜利。

5.3.2.5　地主的智能水平比较

将训练三十天的 DMC-Nash 和 DMC 分别作为地主智能体，与 DouZero 算法、CQL 算法、RHCP 算法和随机算法的农民智能体对战 200,000 局，其中 100,000 局是 DMC 作为地主，所对战的算法作为两个农民智能体。再保持相同的发牌状态，指定 DMC-Nash 作为地主，并且依旧保证所对战的算法作为里两个农民角色的智能体。多体学习训练下地主的胜率比较见表 5.1，多体学习训练下地主的 ADP 比较见表 5.2。

基于多智能体强化学习对地主角色的提升较为明显，除了对战随机出牌方法下地主智能体的胜率和 ADP 稍微下降外，在对战其他算法时的地主智能水平都得到了一定的提升。其中在对战当前领先的斗地主算法 DouZero 时，DMC-Nash 地主智能体的胜率可以达到 0.483。并且 DMC-Nash 地主智能体首次在 ADP 上获得了胜利，达到了 0.081。DMC-Nash 算法在整体上以 0.086 的平均差异得分率打败目前最好的 DouZero 算法。

表 5.1 多体学习下地主胜率的比较

Table 5.1 Comparison of the landlord's WP with Nash Q-learning

WP	DouZero	CQL	RHCP	随机
DMC	0.454	0.863	0.887	0.978
DMC-Nash	0.483	0.870	0.892	0.972

表 5.2 多体学习下地主 ADP 的比较

Table 5.2 Comparison of the landlord's ADP with Nash Q-learning

ADP	DouZero	CQL	RHCP	随机
DMC	-0.333	1.342	1.441	2.995
DMC-Nash	0.081	1.490	1.933	2.897

5.3.2.6 消融实验

将训练三十天的 DMC-Nash、DouZero-Nash 和 DouZero 分别作为地主智能体，与 DouZero 算法、CQL 算法、RHCP 算法和随机算法的农民智能体对战 10,000 局，其中 5,000 局是比较的算法作为地主，所对战的基准算法作为两个农民智能体。再保持相同的发牌状态，指定 DMC-Nash、DouZero-Nash 和 DouZero 作为地主，并且依旧保证所对战的基准算法作为里两个农民角色的智能体。地主的胜率比较见表 5.3，地主的 ADP 比较见表 5.4。

表 5.3 地主的胜率比较

Table 5.3 Comparison of the landlord's WP

WP	DouZero	CQL	RHCP	随机
DouZero	0.416	0.725	0.769	0.986
DouZero-Nash	0.453	0.770	0.888	0.971
DMC-Nash	0.483	0.870	0.892	0.972

从表5.3中可以看到，基于多体学习对地主角色的提升较为明显，除了对战随机出牌方法下地主智能体的胜率和 ADP 稍微下降外，在对战其他算法时的地主智能水平都得到了一定的提升。其中在对战当前领先的斗地主算法 DouZero 时，DMC-Nash 地主智能体的胜率可以达到 0.483。并且 DMC-Nash 地主智能体首次

表 5.4 地主的 ADP 比较

Table 5.4 Comparison of the landlord's ADP

ADP	DouZero	CQL	RHCP	随机
DouZero	**-0.435**	**1.368**	**1.492**	**2.254**
DouZero-Nash	**0.007**	**1.432**	**1.441**	**2.995**
DMC-Nash	**0.081**	**1.490**	**1.933**	**2.897**

在 ADP 上获得了胜利，达到了 0.081。DMC-Nash 算法在整体上以 0.086 的平均差异得分率打败目前最好的 DouZero 算法。采用了多体学习的 DouZero 算法相比于随机抽取样本的方式可以提升地主的胜率和 ADP，这说明结合多体学习的深度蒙特卡洛方法可以针对性的提升地主的智能水平。并且对比 DouZero-Nash 和 DMC-Nash 算法可以看到，结合了残差网络和随机优先采样的斗地主算法可以进一步的提升地主的智能水平。

5.4　本章小结

本章主要介绍了结合多体学习的深度蒙特卡洛方法，针对地主智能性较低的问题，通过将斗地主中的玩家分为农民和地主两组，构成两组零和博弈并求解纳什均衡，再利用极大极小 Q 算法对地主智能体进行单独训练，地主与农民间交互学习当前状态下的决策任务，实现了多体协同机制。与现有的斗地主算法的对战实验表明，多体学习的训练方式明显地增强了地主的对战能力，提高了地主的智能水平。

第 6 章　总结与展望

本章首先总结了基于深度蒙特卡洛方法的斗地主算法在网络结构设计、样本采样方法和智能体训练方式等三个方面做出的改进工作，并且通过对本文研究内容的分析，引出了对未来斗地主算法的展望。

6.1　研究总结

机器博弈是目前人工智能的核心问题，可以作为人工智能技术绝佳的研究和实验平台，其研究能够推动人工智能技术由感知智能迈向决策智能。其中的不完美信息博弈是对许多现实问题的抽象和模拟，具有博弈信息私有、游戏环境随机、信息动态变化以及状态空间复杂等特点，使其研究和解决存在巨大的挑战。斗地主游戏作为一种典型的不完美信息博弈，具有合作竞争并存、固定牌型复杂以及决策序列较长等特点，使其研究和解决形成了特有的挑战。近年来，以 DouZero 为代表的斗地主算法采用深度蒙特卡洛方法，在实战中展现了较强的智能水平。但是，已有的斗地主算法存在以下三个问题：

（一）**性能不稳定：**在其与随机出牌、CQN 算法和 RHCP 算法等的实际对战中，深度蒙特卡洛模型的胜绩不够稳定，受玩家初始手牌的影响较大，并且在游戏过程中玩家的手牌存在动态变化。这说明模型的泛化能力较差，不能很好的应对“运气”所带来的影响，算法的胜率和等分率方差大。

（二）**样本利用率低：**结合残差网络的深度蒙特卡洛方法需要大量的对局记录作为训练样本，虽然模型的训练时间与 CQL 等算法相比已经有了极大的减少，但是在实际中样本的利用率低，训练周期较长，模型学习的时间在一个月左右。算法在采样样本时随机选择经验池中的样本，而样本池中的样本大多为稀疏奖励，很多样本的实际信息含量低，这使得模型在训练中的样本利用效率较低，网络的训练周期长。

（三）**地主智能水平较低：**虽然深度蒙特卡洛模型可以分别为三个玩家训练了不同的智能体，但斗地主游戏本身就具有角色的不对等性，地主相比农民需要更保守的策略。然而，算法在不同角色上的智能水平存在差异，农民角色的胜率和得分率明显高于地主，说明地主的智能水平较弱。

为解决上述问题，本文从网络结构设计、样本采样方法和智能体训练方式三个方面开展一系列工作，并取得如下结果：

（一） 针对斗地主算法性能不稳定的问题，在利用多 Actor 并行计算产生大量历史对局记录的条件下，本文引入残差网络模型设计深层的强化学习 Q 网络，既提高了 Q 网络的泛化能力，又可以利用浅层网络的优势来避免退化。利用基于残差网络的 Q 网络和基于长短时记忆网络的状态网络，提取出斗地主中更深层次的局面信息特征和历史记录特征。结合了残差网络的深度蒙特卡洛方法，降低了训练后智能体性能的不稳定性，提高了斗地主算法整体的智能水平，并且以平均胜率为 0.544 的成绩打败了当前领先的 DouZero 算法。

（二） 针对样本利用率低的问题，本文基于随机优先采样方法，在随机采样的基础上引入样本优先级，采样时样本被抽取的概率与优先级成正比，使得模型可以优先使用经验池中具有较大 TD-error 的样本，改进了原有的多 actor 并行算法流程。同时，由于随机优先采样在采样时也具有随机性，所有样本都有机会被采样，也增加了样本的多样性。结合了随机优先采样的深度蒙特卡洛方法，显著地减少了模型的收敛时间，提升了模型中样本的利用率。

（三） 针对地主智能水平较低的问题，本文基于多体学习算法，差异化地训练不同智能体来应对游戏中的不同角色，通过将斗地主中的玩家分为农民和地主两组，可以构成两组零和博弈并求解纳什均衡，再利用 Minmax-Q 算法对地主智能体进行单独训练，地主与农民间交互学习当前状态下的决策任务，实现了多体协同机制。结合了多智能体强化学习的深度蒙特卡洛方法，明显地提高了地主的平均胜率和平均得分率，增强了地主的智能水平，算法以 0.086 的平均差异得分率打败目前最好的 DouZero 算法；

本文研究了基于深度蒙特卡洛方法的斗地主算法，改进了目前算法在对战性能稳定性、样本利用率和地主智能水平等方面存在的问题，整体上提高了斗地主算法的智能性，在平均胜率和平均差异得分率上打败了当前最好的斗地主算法 DouZero，最终的平均胜率为 0.558，平均差异得分率为 0.086。

斗地主智能算法的研究，可以为其他的多人一般和博弈游戏提供解决范例。斗地主算法中针对大规模状态信息集、合作竞争并存等游戏特点的设计，也给一般性的不完美信息博弈提供了坚实的研究基础。

6.2　未来展望

对于斗地主算法，目前本文已经提高了性能的稳定性、样本的利用率和地主的智能水平，一定程度上弥补了当前斗地主算法的弱势。但是，目前依旧存在农民与地主间的交互学习不完善、无法应对多变的对手策略、无法最大化地剥削对手等问题，下面列出斗地主算法还可以继续研究的技术路径。

（一）**更加合理的交互学习：**在利用交互的多智能体强化学习算法时，Littman 等 (2001) 提出的分组 Q 算法（Friend-or-Foe Q-Learning，FFQ）也可以用于斗地主游戏。该方法是从 Minmax-Q 算法拓展而来，对于每一个玩家的智能体，将其他所有玩家分为两组，一组为该玩家的朋友帮助其一起追求奖励的最大化，另一组为该玩家的对手，完全对抗并降低该玩家的奖励回报，因此对每个玩家而言都有两组其他玩家。在斗地主游戏中，对于农民角色，其他玩家可以分为一个朋友和一个对手玩家，而对于地主角色，其他玩家都分为对手玩家。文章中采用的极大极小 Q 学习来实现两队零和博弈，这样每一组有一个领导者控制这一队玩家的所有策略，获取的奖励值也是这一个小组的整体奖励值。不同的是，FFQ 算法没有领导者和联合策略，每个玩家都通过学习自己的策略获得个人奖励值。但是为了更新值，每个智能体需要在每一步观测其他所有朋友与对手的动作。

（二）**更加准确的对手建模：**本文虽然利用了最近五轮的历史记录作为对手特征，但是还不能很好地建模不同对手的策略，需要利用更加复杂的特征来表征对手的策略空间。同时在对战的采样中需要加入更多种类的对手策略进行训练，从而可以得到更多的局面和对手策略信息，以增强当前策略的全面性。

（三）**在线博弈时的策略更新：**本文主要针对的是离线训练阶段斗地主算法出现的问题，下一步研究的方向可以是在线博弈阶段对未知对手的最大剥削问题。算法可以在对未知对手的策略进行风格度量后，能够根据对手的具体博弈风格及时更新自身的博弈策略。同时，在对手建模的置信区间较低时，智能体能够使用较为安全的动作策略，从而避免重大的决策失误。

参考文献

彭文. 基于蒙特卡洛树搜索的“斗地主”研究 [J]. 2020.

沈恒恒. 基于 UCT 算法和深度强化学习的非完备博弈策略研究与应用 [D]. 安徽大学, 2021.

王骄, 王涛, 罗艳红, 等. 中国象棋计算机博弈系统评估函数的自适应遗传算法实现 [D]. 2005.

马骁, 王轩, 王晓龙. 一类非完备信息博弈的信息模型 [J]. 计算机研究与发展, 2010, 47(12).

王骄, 孙英龙, 吕辉展, 等. 中国象棋中亚洲棋规循环问题的解决 [J]. 东北大学学报 (自然科学版), 2012, 33(4): 476.

徐长明. 基于连珠模式的六子棋机器博弈关键技术研究 [D]. 沈阳: 东北大学, 2010.

张小川, 陈光年, 张世强, 等. 六子棋博弈的评估函数 [J]. 重庆理工大学学报: 自然科学, 2010 (2): 64-68.

Bampton H J. Solving imperfect information games using the monte carlo heuristic [D]. Citeseer, 1994.

Bench-Capon T J, Dunne P E. Argumentation in artificial intelligence [J]. Artificial intelligence, 2007, 171(10-15): 619-641.

Berlinski D. The advent of the algorithm: the 300-year journey from an idea to the computer [M]. Houghton Mifflin Harcourt, 2001.

Billings D, Burch N, Davidson A, et al. Approximating game-theoretic optimal strategies for full-scale poker [C]//IJCAI: volume 3. 2003: 661.

Brown N. Equilibrium finding for large adversarial imperfect-information games [D]. US army, 2020.

Brown N, Sandholm T. Superhuman ai for heads-up no-limit poker: Libratus beats top professionals [J]. Science, 2018, 359(6374): 418-424.

Brown N, Sandholm T. Superhuman ai for multiplayer poker [J]. Science, 2019, 365(6456): 885-890.

Buchanan B G. A (very) brief history of artificial intelligence [J]. Ai Magazine, 2005, 26(4): 53-53.

Chen J X. The evolution of computing: Alphago [J]. Computing in Science & Engineering, 2016, 18(4): 4-7.

Dahl F A. A reinforcement learning algorithm applied to simplified two-player texas hold’ em poker [C]//European Conference on Machine Learning. Springer, 2001: 85-96.

Du W, Ding S. A survey on multi-agent deep reinforcement learning: from the perspective of challenges and applications [J]. Artificial Intelligence Review, 2021, 54(5): 3215-3238.

Egorov M. Multi-agent deep reinforcement learning [J]. CS231n: convolutional neural networks for visual recognition, 2016: 1-8.

Fuller S H, Gaschnig J G, Gillogly J, et al. Analysis of the alpha-beta pruning algorithm [M]. Department of Computer Science, Carnegie-Mellon University, 1973.

Ginsberg M L. Multivalued logics: a uniform approach to reasoning in artificial intelligence [J]. Computational intelligence, 1988, 4(3): 265-316.

Goodfellow I, Bengio Y, Courville A. Deep learning [M]. MIT press, 2016.

Graves A. Long short-term memory [J]. Supervised sequence labelling with recurrent neural networks, 2012: 37-45.

Greenwald A, Hall K, Serrano R, et al. Correlated q-learning [C]//ICML: volume 3. 2003: 242-249.

Gronauer S, Diepold K. Multi-agent deep reinforcement learning: a survey [J]. Artificial Intelligence Review, 2022, 55(2): 895-943.

Gupta J K, Egorov M, Kochenderfer M. Cooperative multi-agent control using deep reinforcement learning [C]//International conference on autonomous agents and multiagent systems. Springer, 2017: 66-83.

Hart S, Mas-Colell A. Simple adaptive strategies: from regret-matching to uncoupled dynamics: volume 4 [M]. World Scientific, 2013.

He K, Zhang X, Ren S, et al. Deep residual learning for image recognition [C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.

Hochreiter S, Schmidhuber J. Long short-term memory [J]. Neural computation, 1997, 9(8): 1735-1780.

Horgan D, Quan J, Budden D, et al. Distributed prioritized experience replay [J]. arXiv preprint arXiv:1803.00933, 2018.

Hu J, Wellman M P. Nash q-learning for general-sum stochastic games [J]. Journal of machine learning research, 2003, 4(Nov): 1039-1069.

Jiang Q, Li K, Du B, et al. Deltadou: Expert-level doudizhu ai through self-play. [C]//IJCAI. 2019: 1265-1271.

Kaplan A, Haenlein M. Siri, siri, in my hand: Who’s the fairest in the land? on the interpretations, illustrations, and implications of artificial intelligence [J]. Business Horizons, 2019, 62(1): 15-25.

Kirzhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks [J]. Advances in neural information processing systems, 2012, 25: 1097-1105.

Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks [J]. Advances in neural information processing systems, 2012, 25.

Lan Q, Pan Y, Fyshe A, et al. Maxmin q-learning: Controlling the estimation bias of q-learning [J]. arXiv preprint arXiv:2002.06487, 2020.

LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition [J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324.

LeCun Y, Bengio Y, Hinton G. Deep learning [J]. nature, 2015, 521(7553): 436-444.

Lieberum J. An evaluation function for the game of amazons [J]. Theoretical computer science, 2005, 349(2): 230-244.

Littman M L. Markov games as a framework for multi-agent reinforcement learning [M]//Machine learning proceedings 1994. Elsevier, 1994: 157-163.

Littman M L, et al. Friend-or-foe q-learning in general-sum games [C]//ICML: volume 1. 2001: 322-328.

Lu H, Xia Z. Awt: Aspiration with timer search algorithm in siguo [C]//International Conference on Computers and Games. Springer, 2008: 264-274.

Marsland T A, Campbell M. Parallel search of strongly ordered game trees [J]. ACM Computing Surveys (CSUR), 1982, 14(4): 533-551.

McCorduck P, Cfe C. Machines who think: A personal inquiry into the history and prospects of artificial intelligence [M]. CRC Press, 2004.

McCorduck P, Minsky M, Selfridge O G, et al. History of artificial intelligence. [C]//IJCAI. 1977: 951-954.

Mnih V, Kavukcuoglu K, Silver D, et al. Playing atari with deep reinforcement learning [J]. arXiv preprint arXiv:1312.5602, 2013.

Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning [J]. nature, 2015, 518(7540): 529-533.

Moravčík M, Schmid M, Burch N, et al. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker [J]. Science, 2017, 356(6337): 508-513.

Nash Jr J F. Equilibrium points in n-person games [J]. Proceedings of the national academy of sciences, 1950, 36(1): 48-49.

OroojlooyJadid A, Hajinezhad D. A review of cooperative multi-agent deep reinforcement learning [J]. arXiv preprint arXiv:1908.03963, 2019.

Ortiz-Gomez F G, Tarchi D, Martínez R, et al. Cooperative multi-agent deep reinforcement learning for resource management in full flexible vhts systems [J]. IEEE Transactions on Cognitive Communications and Networking, 2021: 1-1.

Schaul T, Quan J, Antonoglou I, et al. Prioritized experience replay [J]. arXiv preprint arXiv:1511.05952, 2015.

Sigtia S, Boulanger-Lewandowski N, Dixon S. Audio chord recognition with a hybrid recurrent neural network. [C]//ISMIR. 2015: 127-133.

Sutton R S, McAllester D, Singh S, et al. Policy gradient methods for reinforcement learning with function approximation [J]. Advances in neural information processing systems, 1999, 12.

Takeuchi S, Kaneko T, Yamaguchi K. Evaluation of game tree search methods by game records [J]. IEEE Transactions on Computational Intelligence and AI in Games, 2010, 2(4): 288-302.

Thomas H C, Charles E L, Ronald L R, et al. Introduction to algorithms [Z]. 2016.

Watkins C J, Dayan P. Q-learning [J]. Machine learning, 1992, 8(3): 279-292.

Winston P H. Artificial intelligence [M]. Addison-Wesley Longman Publishing Co., Inc., 1992.

You Y, Li L, Guo B, et al. Combinatorial q-learning for dou di zhu [J/OL]. Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 2020, 16(1): 301-307. https://ojs.aaai.org/index.php/AIIDE/article/view/7445.

Zha D, Xie J, Ma W, et al. Douzero: Mastering doudizhu with self-play deep reinforcement learning [C/OL]//Meila M, Zhang T. Proceedings of Machine Learning Research: volume 139 Proceedings of the 38th International Conference on Machine Learning. PMLR, 2021: 12333-12344. https://proceedings.mlr.press/v139/zha21a.html.

Zhang C, Kuppannagari S R, Xiong C, et al. A cooperative multi-agent deep reinforcement learning framework for real-time residential load scheduling [C]//Proceedings of the International Conference on Internet of Things Design and Implementation. 2019: 59-69.

Zhang H, Yu T. Alphazero [M]//Deep Reinforcement Learning. Springer, 2020: 391-415.

Zinkevich M, Johanson M, Bowling M, et al. Regret minimization in games with incomplete information [J]. Advances in neural information processing systems, 2007, 20.

致 谢

逝者如斯夫，不舍昼夜，转眼间三年硕士生涯即将结束了。在老师和同学们的帮助下，我在计算所的学习和生活都很顺利。值此论文即将付梓之际，谨在此由衷的向我的导师、同学、家人表示感激！谢谢你们的指导、帮助与支持！

首先，我要感谢我的导师孙晓明老师，在这三年间，孙老师给予我精心的指导和无私的帮助。孙老师严谨认真的治学态度、精益求精的工作方式和实事求是的工作作风令我受益良多，为我树立了学习的榜样。在这三年的学术熏陶中，孙老师培养了我独立思考和解决问题的能力，带我见识了理论计算机科学这一领域的深刻魅力。我由衷地感谢何昆老师在学习和科研上的帮助和指导，三年来细致认真地和我讨论每一次组会和周报的汇报，并且指导我对最终的毕业论文进行修改。还要感谢组里的张家琳、田国敬、郭城等老师的指导和帮助，从我大四来到组里后就一直关心和帮助我在北京的学习和生活。感谢自动化所的兴军亮老师和赵恩民博士在机器博弈研究领域给我的指导和建议。

感谢夏之雨、孙元、杨帅、何啸宇、寿立夫等学长在我三年硕士生活中的帮助和鼓励，与他们的讨论和交流使我能够不断地进步。感谢和我同级的陈祖志、朱钦霖、张硕、聂均鸿等同学在日常的工作和生活中对我的包容和帮助。感谢万宗琪、陶天阳等同学和我在算法博弈论上的交流和讨论，这促使我想要完成斗地主算法的研究。

感谢我的父母，焉得谖草，言树之背，养育之恩，无以回报。感谢他们长久以来对我的包容和支持，二十年多来一直照顾我的人生，哀哀父母，生我劬劳。

非常感谢百忙之中评审此论文的各位专家！

最后在临别之际，行诗一首，留为纪念：

春秋十载度关丘，长安载道自悠悠。
杉木漠漠缠根错，同路瞅瞅云畏楼。
吹灭读书半盏灯，一身不过残雪月。
桃李春风一杯酒，江湖夜雨十年灯。

作者简历及攻读学位期间发表的学术论文与研究成果

基本情况

王成龙，安徽省合肥人，中国科学院计算技术研究所硕士研究生。

研究方向

机器博弈，深度强化学习

教育经历

2019 年 9 月-至今	中科院计算所量子计算与算法理论实验室	硕士研究生
2015 年 9 月-2019 年 6 月	西安交通大学数理试验班	本科生

获奖情况

中国机器博弈竞赛（斗地主）一等奖

中国机器博弈竞赛（麻将）冠军

参与的实验室研究项目

中科院先导项目，基于组合优化和数据学习的智能博弈算法研究与验证