独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得安徽大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签名：陈士洋　　签字日期：2014 年 6 月 4 日

学位论文版权使用授权书

本学位论文作者完全了解安徽大学有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权安徽大学可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后适用本授权书）

学位论文作者签名：陈士洋　　导师签名：

签字日期：2014 年 6 月 4 日　　签字日期：2014 年 6 月 4 日

摘 要

RoboCup，机器人足球世界杯，是一个国际性的综合赛事，其中的 2D 项目提出了一个复杂的实时多主体环境下的智能体决策问题。当前人工智能正处在由“单主体静态可预测环境中的问题求解”向“多主体动态不可预测环境中的问题求解”过渡的阶段，RoboCup 2D 问题中的智能决策研究代表人工智能的最新理论方向，同时 RoboCup 2D 问题的解决可以助力当前信息时代的深入发展和革新。

RoboCup 2D 问题的重点是高层决策，目前处理高层决策问题的方法有手工策略和各种人工智能的方法。传统的高层决策采用手工策略，手工策略具有很大的主观性，相关参数的选取多根据经验，不能保证很优化；同时手工策略无法考虑所有的比赛情形，对比赛情形动态变化的适应能力差，从而导致球员达成目标的效率底下。基于人工智能的方法则包括强化学习、决策树学习、神经网络学习等，它们由于具有学习能力，优于简单的手工策略。

在强化学习过程中，智能体通过不断进行动作尝试并观察动作的回报，逐渐学会在各种情形下选择对其有利的动作，以使自身在与环境交互过程中获得高的累积回报值。强化学习的环境交互特点和 RoboCup 2D 的客户-服务器交互模式一致；强化学习的连续决策特点和 RoboCup 2D 的周期性决策特点也十分一致；并且强化学习模型对动态不确定环境的适应能力，使得强化学习方法十分适于解决 RoboCup 2D 的高层决策问题，所以本文基于强化学习方法进行 RoboCup 2D 问题研究。

Keepaway，即小规模控球抢球训练问题，是 RoboCup 2D 中的典型子问题。目前有人使用强化学习的方法对 Keepaway 的高层控球策略进行研究，优化了控球球队中持球球员的高层动作决策。然而目将强化学习应用于 Keepaway 问题中抢球球员的动作决策尚无文献研究。在 Keepaway 中，抢球任务和控球任务的任务目标相反，任务特点也有所不同，因而球队策略也存在区别。控球的特点是要求无球球员进行合理的无球跑动，同时持球球员选择合理的传球路线；抢球的特点是则要求抢球球员分工对控球球员进行压迫和逼抢。控球任务对无球球员的跑动要求相对较低，研究重点是持球球员的传球决策；而对于抢球，离球最近的抢球球员的决策比较固定（他必须上前逼抢持球球员，否则球队很难抢下球），剩下的负责拦截传球路线的抢球球员的决策则具有研究价值。本文针对 Keepaway

中抢球任务的上述特点，研究将强化学习应用于抢球球员高层动作决策的问题，主要做了以下工作：

（1）针对传统手工策略效率低的问题，通过对 Keepaway 中抢球任务特点的分析，合理设计了抢球球员强化学习模型的状态空间、动作空间及回报值，并给出了抢球球员的强化学习算法，使球员的决策随着训练的进行得到优化，抢球任务完成时间缩短，抢断成功率提高。

（2）针对较大规模 Keepaway 任务进行普通强化学习耗时太长的问题，利用策略迁移技术，通过合理设计从较小规模到较大规模 Keepaway 抢球任务的迁移学习方案，以及定义两个规模的任务间状态及动作空间映射，并给出抢球球员的迁移学习算法，使抢球球员在较大规模 Keepaway 训练中重用在较小规模 Keepaway 中通过普通强化学习得到的高层策略，实现迁移学习。实验表明迁移学习在训练开始时就表现出较高的决策效率，并且比从零开始的普通强化学习更快地收敛到理想的策略水平，大大缩短了训练时间。

本文的研究成果表明强化学习方法在 Keepaway 高层抢球决策中的有效性。传统意义上，强化学习一般只应用于底层动作决策。本研究则证明了通过合理的高层回报值模型设计，强化学习也可以用来解决高层动作决策问题，体现了强化学习更广泛的应用能力。

关键词：机器人足球；Keepaway；强化学习；抢球策略；策略重用；迁移学习

Abstract

RoboCup, Robot Soccer World Cup, is an international comprehensive event, in which simulation 2D league proposes a complex decision problem in real-time and multi-agent environment. As current trend of artificial intelligence is turning from "solving single-agent problem in static predictable environment" to "solving multi-agent problem in unpredictable environment", research in agent decision problem on RoboCup 2D represents the newest theoretical direction of artificial intelligence, and solving RoboCup 2D problem contributes the deep development and revolution of current society.

The key point in RoboCup 2D problem is the high-level decision. For high-level decision, there are hand-coded strategies and a series of artificial intelligence methods. Traditional high level decision takes hand-coded strategies and suffers the issue of subjectivity: decision-related parameters are just set by experience which doesn't guarantee optimum; and hand-coded strategies naturally can't consider all possible situation in a game, thus can't adapt well to dynamic change of environment, which make players perform badly. Artificial intelligence methods include reinforcement learning, decision tree learning, neural network learning and so on. As with a learning nature, they are always better than hand-coded strategies.

In reinforcement learning process, an agent gradually learns to take the best action under each situation by keeping trying, observing reward and updating its knowledge, thus hope to make itself acquire the highest accumulated rewards. The interactive feature of reinforcement learning is in step with Client/Server interactive mode in RoboCup 2D, and sequential decision character in reinforcement learning is in accordance with periodic decision character, these facts make reinforcement learning method very suitable for solving high level decision problem in RoboCup 2D. Research in this paper is based on reinforcement learning methods.

Keepaway, in which two small teams compete for the possession of ball, is a typical subtask in RoboCup 2D. There has been researches on high-level protecting strategy based on reinforcement learning which optimizes the decision of keepers. But there hasn't been any researches on high-level stealing strategy using reinforcement learning. In Keepaway, stealing task and protecting task have opposite goal, and their task features are also different, so their corresponding strategies should be different. Protecting task needs keepers who don't ball to go to open area for keeper who has

the ball to have a route for pass the ball; while stealing task asks all takers to get close to keepers and try to touch the ball. Protecting task majorly concerns decision of the keeper who currently has the ball; while for stealing task, the taker closed to the ball has a fixed strategy(he must go to the ball, or his team will never win), and the rest taker's decision are of great research value. Focused on features of stealing task in Keepaway, this paper researches on how to apply reinforcement learning to high-level stealing strategy, and the related work are as follow:

(1) Though analysis of stealing task, we design reasonable state space, action space and reward function for the reinforcement learning model of takers, and present a reinforcement learning algorithm. Experiments show that after reinforcement learning, takers make more reasonable high-level decisions that are much better than hand-coded decisions.

(2) By rational decision of policy transferring scheme from smaller scale to bigger scale and definition of mappings between two scales, we managed to let the taker in bigger scale task do reinforcement learning in which he can reuse policy which has been learned in smaller scale task by normal reinforcement learning. Experiments prove that for the same bigger scale task, the taker using reinforcement learning with policy transferring technique performs better even at the beginning than the taker using normal reinforcement learning.

Results in the paper show the effectiveness of reinforcement learning in high-level stealing decision in Keepaway task. Traditionally, reinforcement learning is only applied for low-level decision. This paper proves that by reasonable design of high-level reward model, reinforcement learning can also be used for high-level decision, showing the wider application ability of reinforcement learning.

Keywords: RoboCup; Keepaway; Reinforcement Learning; Stealing Strategy; Policy Reuse; Transfer Learning

目 录

第一章 绪论

本章首先介绍RoboCup 2D的研究背景和选题意义，其次介绍与RoboCup 2D相关的国内外研究现状，最后介绍本论文的组织结构和内容安排。

1.1 研究背景及选题意义

1997年之前的50年中，人工智能的主要研究内容是“单主体静态可预测环境中的问题求解”，其典型问题就是国际象棋的人机对抗赛，最终以超级计算机的获胜而告终；1997年之后的50年里，人工智能研究的主要对象应该是“多主体动态不可预测环境中的问题求解”[1]。多智能体环境是为解决复杂问题而出现的若干智能体的团队，它们不仅能通过独立决策来完成自身任务，还能通过沟通合作高效地实现团队总体目标。随着信息技术的进一步发展，可以想象，未来社会一定充斥着很多智能体团队，例如为技术工程服务的智能机器人团队，网络空间中的软件智能体联盟，这些智能体同时具有自主性和社会性[2]，构成未来社会的多智能体环境，为人类生活服务。在这样的社会发展趋势及其对信息技术和人工智能科学的最新要求下，为了促进人工智能和智能机器人研究，国际上成立了RoboCup联盟。

RoboCup（Robot Soccer World Cup），机器人足球世界杯，是一个国际性的联盟[3]。其中的RoboCup仿真2D（简称RoboCup 2D）竞赛项目利用计算机模拟2D环境下的机器人，提出了让11个仿真机器人完成类似人类11人制足球比赛任务的问题[4]。RoboCup 2D问题是一个复杂的实时多自环境下的智能体决策（agent decision making）问题，极好地代表了当前人工智能发展的前沿和热点。选择RoboCup 2D问题作为研究课题具有重要意义：

首先，RoboCup 2D问题中的智能决策研究代表人工智能的最新理论方向[1]。未来的50年，人工智能研究的主要对象是“多主体动态不可预测环境中的问题求解”，RoboCup 2D问题正是上述问题的一个典型代表，它涉及到智能体设计、多主体体系结构、通讯和合作、实时推理、规划和机器学习等一系列人工智能课题。

其次，RoboCup 2D问题的解决可以助力未来社会的发展和革新。RoboCup 2D问题在提出时就紧扣当前人工智能理论研究的关节，并贴合未来应用的实际，

其研究成果易于转换成实际应用，例如公共基础设施，办公系统，教育辅助系统等，以及太空探险等新兴领域，都将发生巨大变化[1]。回顾人类历史，第一次和第二次工业革命，分别帮助人类进入蒸汽时代和电气时代，使人类从体力劳动中解放出来；第三次工业革命以来，计算机及网络技术蓬勃发展，作为人类智能的延伸，使人类进入信息时代；未来的趋势必将是由智能的机器和软件的网络构成多智能体系统，充斥着人类生活的方方面面，将人类从脑力劳动中解放出来[5]。解决了 RoboCup 2D 所代表的问题，能够实现人工智能的新突破以及人类脑力的解放。

处理 RoboCup 2D 问题，一般采取分层的方法将问题分为底层技术动作（底层技术）和高层动作决策（高层决策）[1, 6-10]。底层技术负责具体实现球员的一个基本动作，例如传球、射门、铲球、封堵等等[9]。高层决策通过合理地调用底层技术，使球员在比赛中做出利于球队赢球的决定[10]。由于高层决策负责球员的总体目标，因而是球员策略的重点部分。

目前高层决策问题的方法有手工策略和各种人工智能的方法。传统的高层决策采用手工策略，使用手工策略时，球员根据当前情形下各个球员间的位置角度关系，选择自己认为最应该拦截的路线[1, 6, 9]。然而手工策略具有很大的主观性，相关参数的选取多根据经验，不能保证很优化；同时手工策略无法考虑所有的比赛情形，对比赛情形动态变化的适应能力差，从而导致球员达成目标的效率底下。基于人工智能的方法包括决策树学习、神经网络学习、强化学习等，它们由于具有学习能力，效果优于简单的手工策略[1, 9]。

强化学习的环境交互特点[10]和 RoboCup 2D 的客户-服务器交互模式[4]一致，强化学习的连续决策特点[11]和 RoboCup 2D 的周期性决策特点[4]也十分一致，加上强化学习模型对动态不确定环境的适应能力[11]，使得强化学习方法十分适于解决 RoboCup 2D 的高层决策问题。所以本文将进行基于强化学习的 RoboCup 2D 高层抢球策略研究。

1.2 国内外研究现状

RoboCup 2D 为智能行为决策和强化学习提供了一个标准的研究平台，其核心问题是一个巨大的连续状态、动作及观察空间的多智能体学习和实时决策问题[1]。自 1997 年 RoboCup 2D 比赛首次举办，16 年来它吸引和激励了来自世界各

地的很多研究者，关于 RoboCup 2D 的文章已有数百篇[1]。在最早的比赛中，机器人的策略只是手工的分支判断与选择，没有学习和智能可言。16 年过去了，球队的智能化水平已大大提高，但是当初它提出的问题仍然代表着当今人工智能和机器人学的前沿，多智能体环境下的智能决策规划仍是研究的重点和难点。

马尔可夫决策模型（Markov Decision Process，MDP）是智能体与环境交互并进行强化学习的基本模型，经典的强化学习算法有动态规划、蒙特卡罗算法和时序差分（包括 Q 学习和 Sarsa 算法等）[11]。底层动作训练是强化学习非常适用的应用领域，Riedmiler 用强化学习的方法在 RoboCup 2D 的底层技术动作方面进行了研究[12]。他使用实时动态规划方法让球员学习的个人踢球技术，在处理连续状态空间时采用了神经网络，经过 2 小时的学习，可以很好踢球效果，踢球动作很成熟。

Stone 则尝试将强化学习的方法应用于 RoboCup 2D Keepaway 训练的高层控球决策中[13]。他先手工实现若干底层技术动作，作为高层决策的候选动作集，然后通过合理设计状态空间以及巧妙定义回报值，使 Sarsa 学习在控球球员的高层决策中发挥作用，决策效果远优于手工策略。针对连续状态空间，他使用了 tile-coding 及哈希表的技术[14]，很好地解决了状态空间离散后 Q 值表过于庞大的问题。左国玉在 Peter Stone 的基础上，考虑到控球总会失败的特点，由单智能体杆平衡系统问题的回报函数得到启发，设计了一种新的惩罚式的回报函数，进一步优化了控球球队中持球球员的高层动作决策[15]。

Taylor 和 Fernández 将对 RoboCup 2D Keepaway 训练的强化学习算法进行扩展，进行了高层控球策略的迁移学习研究。Taylor 通过定义动作值函数的迁移函数进行迁移学习，使控球球员在比传统强化学习更快地获得较好的控球效率[16]。Fernández 则使用了 PRQ-Learning 迁移学习算法，通过重用在较小规模 Keepaway 控球任务中学到高层策略，让控球球员在较大规模的 Keepaway 控球任务中进行基于策略重用的迁移学习，发现球员比只进行普通强化学习时的学习效率有所提高[17]。

1.3 本论文的主要内容

本论文首先介绍研究背景和意义，然后介绍了研究的平台 RoboCup 仿真 2D 以及相关的强化学习理论的基础，接着介绍高层抢球决策的强化学习和迁移学习

研究方案及结果分析，最后进行总结和展望。

本论文各个章节的内容规划如下：

第一章 绪论

首先介绍本文的研究背景和选题意义，其次介绍相关的国内外研究现状，最后了介绍本论文的组织结构和内容安排。

第二章 RoboCup 仿真 2D 平台

首先介绍 RoboCup 比赛的由来和概况，其次介绍 RoboCup 2D 比赛项目的概况，然后介绍 RoboCup 仿真 2D 平台的问题模型和特点，最后分析 RoboCup 2D 中的若干具有代表性的子问题。

第三章 强化学习

首先介绍强化学习的概念，其次分析强化学习问题并介绍马尔可夫模型，然后介绍用马尔可夫模型如何解决强化学习问题，最后介绍三大类强化学习算法。

第四章 高层抢球策略的强化学习

首先介绍在 Keepaway 任务中传统手工策略存在的问题，其次介绍 Keepaway 任务中抢球和控球球员的总体策略框架和高层动作，然后介绍如何将强化学习应用于抢球球员的高层动作决策中，接着介绍实验分析，最后进行总结。

第五章 高层抢球策略的任务间迁移学习

首先介绍在 RoboCup 2D Keepaway 任务中普通强化学习方法存在的问题，其次介绍迁移学习和策略重用相关概念，然后介绍如何利用策略重用技术实现抢球球员高层动作决策的迁移学习，接着介绍实验分析，最后进行总结。

第六章 总结和展望

首先对本文的工作进行总结，包括贡献和不足，然后对未来可以进行的改进和深入研究的地方进行展望。

第二章 RoboCup 2D 平台

2.1 RoboCup 比赛

RoboCup 的长期目标是：到 2050 年，一支完全自治的人形机器人足球队能够在遵循国际足联正式比赛规则的比赛中，战胜当时的人类世界杯足球冠军队伍[18]。

1992 年加拿大的 Alan Mackworth 教授最先提出了机器人足球比赛的想法[19]，1993 年，国际著名科学家北野宏明和浅田埝等研究者创办了 RoboCup J 联赛[20]，之后其他国家的学者要求扩展该项目，RoboCup 联盟由此产生。1997 年，RoboCup 机器人足球世界杯赛在日本名古屋与国际权威的人工智能系列学术大会——第 15 届国际人工智能联合会议（IJCAI-97）联合举行，机器人足球比赛被正式列为人工智能的一项挑战[21]。机器人足球由此被越来越多的研究者认识，相关的研究和会议活动逐渐广泛开展起来。一些国际有名的学术刊物如 Artificial Intelligence Journal、Applied Artificial Intelligence 等都出版了机器人足球专辑，一些有影响的国际学术会议如 IJCAI 等也都安排了这方面的专题讨论[1]。

RoboCup 在我国起步稍晚，1999 年开始举办首届中国公开赛，即 RoboCup China Open，当时只有清华大学和中国科学技术大学参赛，如今已经有 500 余支队伍，其中有中科院自动化所，清华大学，北京大学，浙江大学，上海交通大学，国防科技大学，南开大学，西安交通大学，安徽大学，厦门大学等。RoboCup 中国公开赛是世界 RoboCup 活动的一个重要组成部分，中国的 RoboCup 2D 队伍也实力不凡，中国科学技术大学近年来多次获得世界冠军，处于世界一流水平。

2.2 RoboCup 2D 平台架构

RoboCup 2D 比赛是在标准的计算机环境内进行的，因而称作仿真 2D 比赛[22]。RoboCup 2D 比赛规则基本上与国际足球联合会的比赛规则一致[23]。RoboCup 2D 平台总体系统框架如图 2-1 所示，采用 Client/Server 结构，服务器 Soccerserver 由 RoboCup 组织开发和维护，它提供了一个虚拟场地并且模拟包括球和球员在内的所有物体的移动[24]；客户端也就是仿真 2D 球队由参赛队伍开发和维护，每个 Soccerclient 相当于一个球员的大脑，控制场上该球员的移动。服务器端和客

户端之间都是通过 UDP/IP 协议进行信息交互的，当一场比赛开始时，双方的球员程序连接到比赛平台上开始比赛，每个队的目标就是按照人类足球的比赛规则比赛[4]。服务器的接口要求就是 RoboCup 2D 的问题的表示，参赛球队的策略则是 RoboCup 2D 问题的解决方案。

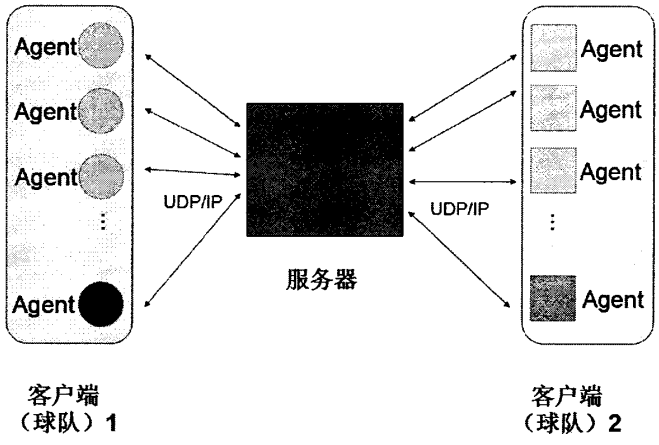


图 2-1 RoboCup 2D 平台总体系统框架

Figure 2-1 Overall Architecture of RoboCup 2D Platform

RoboCup 2D 平台服务器端系统框架如图 2-2 所示，主要包括两大部分：Soccerserver 和 Soccermonitor。Soccerserver 作为一个服务器程序提供了一个虚拟的足球场地，模拟所有球员和球的移动、和球员通讯以及根据比赛规则控制比赛进程。Soccermonitor 是一个程序将从 Soccerserver 那里获得场上信息显示到一个虚拟场地上。

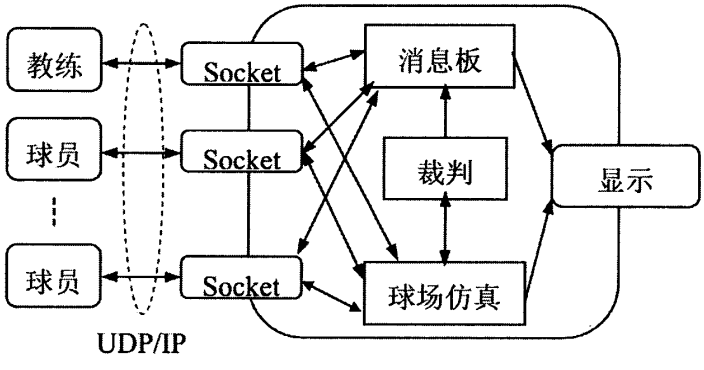


图 2-2 RoboCup 2D 平台服务器端系统框架

Figure 2-2 Architecture of RoboCup 2D Server

Soccerserver 主要由球场仿真模块、裁判模块和消息板模块三个部分组成[1]。球场仿真模块计算球场上球员和球的运动，检测他们之间的碰撞。球场仿真模块采用离散模拟的方式，在每个周期末依据是动力学定律计算更新一次球员和球的位置和速度等信息。裁判模块根据类似人类国际足球比赛的规则来控制比赛的进程，消息板模块负责与客户端之间的通讯。Soccermonitor 通过特定端口和 Soccerserver 连接，展现球场上所有球员和球的运动状况等信息，使得用户可以想观看人类足球比赛一样看到比赛的整个过程。另外为了方便的重现比赛实况，平台还提供了比赛录像播放器（Logplayer），它可以用来重放比赛。

一个球队最多可以连接 12 名队员包括 11 名球员和一个场上教练。这些球员程序向比赛平台发送请求执行相应行为（如踢球、转身、跑步等）；另一方面，它们从服务器端接收自身可以感知到的信息，如球员可以看到的视觉信息、球员自身的状态信息等[1]。RoboCup 2D 球员基本系统框架如图 2-3，图中粗箭头代表控制流程和数据流向，细箭头代表数据流向。基本球员框架包含三个模块：感知模块、规划模块和执行模块。感知模块负责接收观察信息，包括球员感知信息、局部视觉信息和听觉信息。规划模块负责实时决策，通过分析感知模块获得信息，更新世界模型，决定本周期应该采取的行为，以高级技术动作的形式表示。动作执行模块，将规划模块的高级技术分解为动作命令发送给服务器。动作命令包括 dash、turn、kick、tackle、move 和 catch 六个原子命令，和其他一些辅助命令，如 hear、attention to、turn neck 等等[4]。

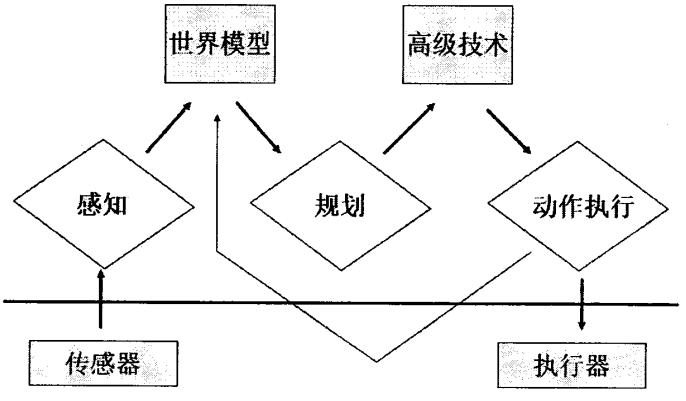


图 2-3 RoboCup 2D 球员基本系统框架

Figure 2-3 Architecture of RoboCup 2D Player Client

2.3 RoboCup 2D 问题模型

本节介绍 RoboCup 2D 的问题模型，包括球员的感知模型、运动模型、动作做模型和体力模型、球员异构模型以及裁判模型。这些模型都有服务器 SoccerServer 实现，所以 SoccerServer 承担模拟球场上的客观规律的作用，包括球和球员运动的物理规律，球员感知和体能的生理规律。

2.3.1 感知模型

RoboCup 2D 球员的感知的包括三类：听觉感知接收裁判、教练及球员们的说话信息；视觉感知接收球场上的信息，包括球员视野中的物体间的距离和方向信息；躯体感知接收球员自身的当前“物理”状态，例如球员的体力值、速度和脖子方向角[4]。三类感知在一起，为仿真球员提供球场上的各类信息，就像人类足球的球员感知一样。

球员、教练可以说话，裁判的广播信息也以说话的形式发出，这些说话信息从收听者的角度，就是听觉信息。听觉模型有一系列限制：仿真平台模仿的是一个拥挤的低带宽环境，所有球员公用一个不可靠的频道；对于收听到的听觉信息，只能得出说话球员的相对角度，不能得出其球衣号[4]。视觉信息给出了球员可以看到的在其视力范围内的球场上物体信息，是球员最重要的感知信息。视觉信息只要有四个特点：第一，视觉信息中的数据是相对于发生视觉的主体球员的，不是球场上的绝对信息；第二，视觉信息具有局部性，它不是场上所有物体的信息，而只包括球员视野中的物体；第三，视觉信息具有不完整性，视觉信息中物体距离球员越远，则信息量越少；第四，视觉信息具有不精确性，视觉信息包含噪声。躯体感知信息包含了球员的视觉模式，体力值，速度，头与身体夹角等信息，是球员自身的体力和运动情况信息。

2.3.2 运动模型

球场上球和球员的直线运动是由一个简单的线性运动规律仿真的。在每个仿真周期中，首先计算物体在该周期获得的加速的，以此修正物体的速度，同时模拟真实环境添加运动噪声和风力干扰，得到更新后的速度值；其次以新的速度值更新物体的位置，同时模拟真实环境添加运动噪声；然后对速度之进行衰减；最后将加速度值重置为零[4]。噪声的特点是，当物体的速度和加速度越大，运动中

可能产生的噪声误差也越大。

如果一个周期结束时两个对象位置发生重合，那么会把对象按照原来的运动方向后移使其不再重叠，然后两者的速度都乘以－0.1，这就是碰撞处理的模型。注意，由于仿真平台使用离散的仿真周期来模拟实时环境，根据运动模型，足球有可能穿过球员，只要在周期的末尾，足球和球员没有冲撞就行了。

2.3.3 动作模型和体力模型

球员有六个基本动作，它们是 dash（加速）、turn（转身）、kick（踢球）、tackle（铲球）、catch（扑球）和 move（瞬移）命令。这些命令互不相容，在一个仿真周期内球员这可以发送其中的一个命令给服务器，称作原子命令。这是出于对人类球员的身体限制的模拟。

在使用 dash 命令时，球员会发生体力的损耗，RoboCup 2D 仿真平台将球员的体力模型模拟为电池体力模型：体力有一个最大值，还有一个能量有限的体力池，体力在运动中发生消耗，同时会从体力池中逐渐进行补充，但在剧烈跑动中体力补充的速度跟不上消耗的速度[4]。

2.3.4 球员异构模型

球员异构模型是对人类运动员的各自特性的模拟，我们知道不同的人类运动员有不同的身体素质和技术特点：个子高的运动员一般跑得快但转身慢，个子矮的一般跑得慢但转身灵活。RoboCup 2D 仿真平台中的球员异构模型为踢球力量较大的球员设定较大的惯性值，因而其转身困难一点[4]。这样构造出不同能力特点的球员供球队选择使用。

2.3.5 裁判模型

服务器平台中的自动裁判模型可以实现比赛的中场开球、进球、球出界、越位、回传球违例、发球违例、扑球违例等情形的判定，还可以实现罚定位球时球员清理、比赛模式控制以及半场和终场的控制[4]。这些控制基本上与国际足联的规则一致。由于 RoboCup 2D 中没有高度的概念，球员只有脚没有手，所以在罚边线球时仍采用踢球的方式发球。

2.4 RoboCup 2D 问题特点

通过上一节的介绍，可见 RoboCup 2D 平台有以下特点：

（1）问题规模巨大

战胜人类国际象棋冠军的“深蓝”所处的国际象棋问题的规模约为 10^{20}；围棋问题的规模约为 10^{200}；而 RoboCup 2D 具有连续的状态及行动空间，按粗略离散，其决策问题的规模约为 10^{400} 以上。

（2）实时系统

在服务器端，以 100 毫秒为周期离散模拟球场中个物体的运动和相互作用，所有球员都必须按照这个周期决策，如果考虑网络的延迟，实际上球员每次的决策时间只有几十毫秒。

（3）多自主智能体

所有客户端程序分别控制场上的一名球员或教练，自主决策，分布运行，队友之间需要密切合作，对手之间存在激烈对抗。

（4）资源有限

由于对人类足球比赛进行了逼真的模拟，对于每个球员来说，球员的视觉范围、听觉范围都是局部的，使球员获得的比赛信息是局部的；同时场上球员能力也和真实人类球员一样，有体力和速度等方面的限制。这就给球员对场上真实形势的认知以及决策后的执行带来困难。

（5）大量不确定因素

环境是部分可观察的且存在噪音，且多个自主智能体共存于环境中，加上观察噪声和动作噪声性，使得认知具有不准确性，行动结果具有不确定性。并且由于比赛时间以周期为单位离散，感知和行为无法同步，增加了问题中的不确定因素，因此增加了决策的难度。

因此可以看出，RoboCup 2D 平台在设计原则上有两大特点：一是“真”，该平台真实地模拟了机器人在控制、通讯、传感和机能等方面的实际限制，使仿真球队程序易于转化为硬件球队的控制软件；二是“仿”，仿真避免了现实物理环境和当前机器人制造技术的限制，将问题侧重于球队的高级功能上，研究的核心是智能体决策理论。

2.5 RoboCup 2D 子问题

通过上面的介绍可以看到，整个 RoboCup 2D 问题由于十分逼真地定义了近似人类足球比赛的问题，问题规模十分巨大，问题非常复杂，这使得一次性整体

解决 RoboCup 2D 问题十分困难。然而我们可以分层次地分析 RoboCup 2D 问题，正对问题的子模块提取一些子问题；也可以对问题进行简化，简化出一些子问题。这些层次单一或者规模受到限制的子问题的复杂性得到控制，可便于进行研究。子问题的研究成果经过汇总和扩展，有助于整个问题的求解。下面介绍 RoboCup 2D 的几个有代表性的子问题。

2.5.1. 无球跑位

无球跑位任务要求球员把球从一个初始的位置、速度跑到一个特定的区域内，球员可以使用的原子动作有 dash 和 turn 两个。无球跑位属于底层技术动作，实现它可以通过手工策略，然而如前文所述，手工策略的效果有限。目前用强化学习的方法做无球跑位效果较好[8]。

2.5.2. 定点带球

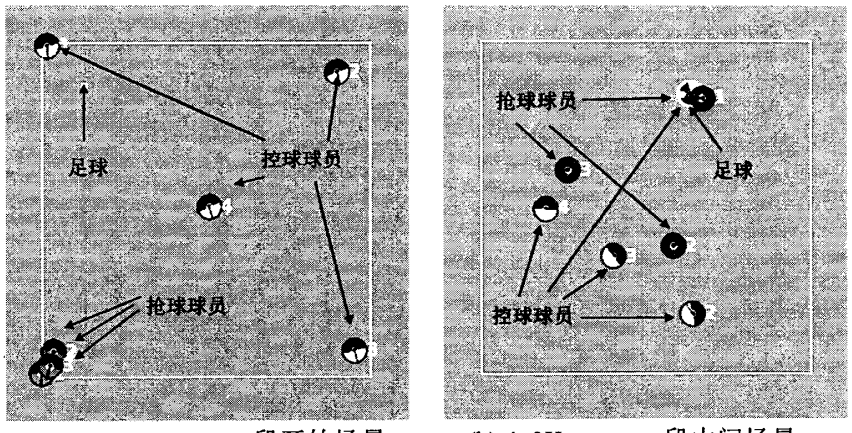
定点带球任务要求球员把球从一个初始状态转移到一个结束状态，状态变量包括球的位置、速度（包括大小和方向）、球员的位置、速度（包括大小和方向）。球员可以使用的原子动作有 dash、turn、kick 和 tackle 四个。定点带球同样属于底层技术动作，目前也已经有很多强化学习的方法做定点带球，效果比单纯的手工策略好[9]。

像无球跑位和定点带球的问题，是强化学习的典型应用领域。强化学习类似于一种客观的策略搜索过程，由于这类问题的状态空间和动作空间规模可控，强化学习应用起来的效果也十分好。

2.5.3. Keepaway

RoboCup 2D 中有一个 Keepaway 子问题，Keepaway 任务中两支球队在一定大小的场地内进行控球-抢球对抗的训练[25]。Keepaway 在场地大小、控球和抢球球员人数等方面有不同规模。由于控球相对较难，一般控球球队比抢球球队多 1 名球员；球员总数一般介于 5 到 9，常见的为 3v2（3 名控球球员、2 名抢球球员之意，后面类推）、4v3 和 5v4 规模；训练区域一般比整个比赛的场地小得多，例如 20m×20m、30m×30m 的正方形区域。图 2-4 展示了 30m×30m 场地下 4v3 规模 Keepaway 任务的训练段开始和中间场景，其中有 4 名控球球员和 3 名抢球球员。

训练段，指从训练开始状态到抢球球员抢下球或球离开训练区域为止的整个过程。在一个训练段开始时，球在一名控球球员身边，其他所有球员距离该球员一定距离；训练段开始后，抢球球队要去争夺足球，控球球队需要维持控球权。当抢球球员踢到球或者球在抢球球队的逼抢下离开训练区域，则抢球球队完成任务，当前训练段结束，开始下一个训练段，Keepaway 训练就这样不断进行下去。



(a) 4v3Keepaway 段开始场景　(b) 4v3Keepaway 段中间场景

(a)Start Scene of 4v3 Scale Keepaway　(b) Middle Scene of 4v3 Scale Keepaway

图 2-4：4v3 Keepaway 场景

Figure 2-4 Game Scene of 4v3 Scale Keepaway

Keepaway 问题从整个 RoboCup 2D 问题中提炼出来，具有独立性和代表性：一方面，Keepaway 问题相对简单，控球方只考虑维持控球权，抢球方只考虑争夺控球权，问题相对简化；另一方面，Keepaway 问题涉及到了足球比赛中核心的问题，即自主智能体的队内合作和队间对抗，对其的研究有助于整个 RoboCup 2D 问题的解决。

2.6 本章小结

本章对 RoboCup 仿真 2D 平台进行介绍。首先介绍了 RoboCup 比赛的由来和概况，其次介绍了 RoboCup 2D 的平台架构，然后介绍了 RoboCup 2D 问题模型和特点，最后分析了 RoboCup 2D 中若干具有代表性的子问题。

第三章 强化学习

3.1 强化学习概述

根据反馈信号形式的不同，机器学习可以分为监督学习、非监督学习和强化学习（reinforcement learning）三大类，其中强化学习是一种以环境反馈为输入的、特殊的、适应性的学习方法[26]。对于监督学习，例如神经网络和决策时学习，每个用例的输入都有对应期望输出值，学习完成的是和环境没有交互的记忆和知识重组的功能[27]；面对交互式并且动作具有长期后续影响的复杂环境，例如机器人行走、直升机飞行问题，监督学习就无能为力了，而强化学习正是针对这类问题提出的。

强化学习是指从环境状态到动作映射的学习，以使智能体从环境中获得的累积回报值最大[28]。强化学习与生物世界的学习有类似之处，当一个婴儿挥动手臂或学习走路时，它与外界环境相联系。通过不断地练习，它获得很多关于原因和结果的信息，它逐渐能够归纳出行为的后果，并知道怎么做能达到想要的目标。不同于监督学习的是，强化学习中没有正例、反例告诉智能体选择哪个动作，而是在采取动作执行后智能体会收到环境的反馈信号，通过反馈信号的好坏来总结应该在什么环境状态下采取什么动作。可以看到，在强化学习中，智能体必须能够感知环境的状态，并能够执行一些动作，这些动作可以改变环境状态；智能体还必须有一个目标，这个目标与环境状态相关联。

强化学习面临一项独特的挑战：利用知识（exploitation）与探索知识（exploration）的权衡。为了获得好的回报，根据已经学到的知识，智能体应该依据学到的策略采取动作，这叫利用知识；然而已经学到的知识可能不是最好的策略，智能体还必须尝试新的动作，这叫探索知识。到底是利用还是探索，智能体在每个决策周期只能做出一个动作，这种矛盾客观存在，这一挑战在其他机器学习中是不存在的。单纯地只利用已有知识或只进行探索都是不行的，并且如果环境中动作的效果存在随机性，每个动作还应该多次探索以得到可信的知识。

强化学习的研究联系了人工智能和其他工程学科如控制理论、统计理论。之前，人们认为控制理论和统计学与人工智能完全不相关，人工智能就是逻辑与符号，而与数字无关。人工智能之前是大型 LISP 程序，而不是线性代数、差分方

程，或是统计理论。从 20 世纪 80 年代末开始，随着对强化学习的数学基础研究取得突破性进展后，人们对强化学习的研究日益开展起来，强化学习成为目前机器学习领域的研究热点之一。而之前的那些关于人工智能的观点得到了改变，现代人工智能研究者同样关注统计理论和控制理论。之前被忽略的人工智能和传统工程学之间的地带如今成为了最活跃的领域，包括人工神经网络，智能控制以及强化学习。

在强化学习中，我们总是提到一个“智能体”的概念，这是有原因的。对于一般的算法而言，我们可能很清楚算法要做什么，不需要用到“智能体”这一概念。而在强化学习中，算法设计者也不是很清楚具体在某个状态该采取什么动作，这时抽象出一个“智能体”十分必要。有了这个概念，我们可以这样理解，算法设计者只告诉“智能体”总体的策略（即通过反馈调整对环境的认知），就像所谓的“授人以鱼不如授人以渔”，具体遇到了什么状态，做出哪个动作选择，就是智能体自己的工作了。这样抽象出来的这层算法就像人和其他生物一样，在试错中学习并且进步，用“智能体”这个词来形容十分形象。

3.2 强化学习问题

3.2.1 智能体与环境的交互

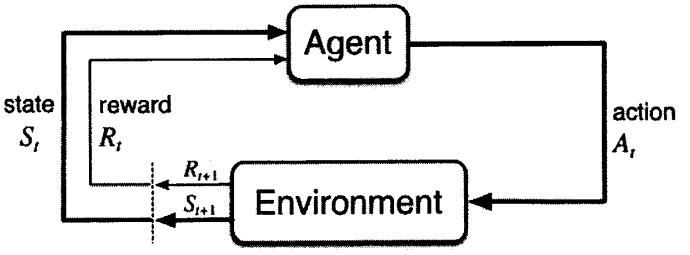


图 3-1 强化学习中智能体与环境的交互模型

Figure 3-1 Interaction Model of Agent and Environment in Reinforcement Learning

强化学习问题直接反映了智能体与环境之间的交互，智能体通过与环境交互并学习，以达到特定目标。智能体（agent）是学习者，同时也是决策执行者。智能体交互的对象，是环境（environment），包括智能体以外的所有部分。图 3-1 表示了这一不断交互的过程。智能体根据当前的环境状态 S_t 做出决策，选择一个动作 A_t；环境在 A_t 的作用下会发生变换，环境状态变为 S_{t+1}，同时智能体在

环境状态转移过程中获得回报值 R_{t+1}；如此循环[29]。

在每一个时间步，智能体需要根据当前环境状态 S_t 选择动作决策 A_t，这个从状态到动作的映射即智能体的策略 π_t。强化学习过程就是智能体根据经验的积累不断调整策略的过程。智能体的目标是最大化累积回报[30]。这一智能体与环境交互的框架适用于很多问题。例如，时间步不一定必须是严格的时间间隔，可以是任何连续的动作步。动作可以是低层控制操作，如机器人电机上的电压，也可以是高层决策，例如是否传球。同样，状态也有多种多样的形式，状态可以是低层确定性的感知信号如读数，也可以是高层的抽象信息。

强化学习中智能体和环境的界限与现实中两者的界限有一些区别。在强化学习中，智能体不能随意改变的均列为环境的一部分，例如机器人的传感设备。以人做比喻，一个人的肌肉、骨骼和感知组织都划归为环境。回报值同样属于环境的一部分。可以看到，实际上在强化学习中，智能体和环境的界限代表了智能体的绝对控制的界限，凡是受客观规律约束的均属于环境。

3.2.2 回报值和视界

智能体的目标是获得最大的期望累积回报，累积回报值包括立即回报和延迟回报两部分。累积回报是指在从当前状态开始，智能体在与环境交互过程中的获得的回报值之和，一般用 R 表示。立即回报是指在动作使环境状态发生改变后的获得的回报，一般用 r 表示。延迟回报是指从当前状态开始到智能体结束强化学习过程为止所获得后续累积回报。

视界是指强化学习的智能体完成任务所需要的执行的动作步数，一般用 T 表示。有限视界指智能体完成任务所需执行的动作步数是有限的，一般用 T（或 H）表示最大步数；无限视界指智能体完成任务所需执行的动作步数是无限的。

对于有限视界问题，累积回报值为：

$$R_t = r_{t+1} + r_{t+2} + \ldots + r_T \tag{3-1}$$

段任务（Episodic Task），也叫有终任务，即有限视界问题，是指任务有一个终结状态，当达到终结状态后，当前任务段就结束了。从初始状态到终结状态一次过程为一个任务段（Episode）。

对于无限视界问题，为了便于研究，引入折扣因子的概念。折扣因子指延迟回报值相当于即时回报的比例，一般用 γ 表示。引入折扣因子后，无限视界问题，

累积回报值为：

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots \text{其中，} 0 \le \gamma < 1\text{，为折扣因子} \quad (3\text{-}2)$$

对于无限视界问题，折扣因子的物理意义很重要，两个相隔很多步的立即回报值其对应的权重必然不一样，就像经济学中通货膨胀的道理一样。而从数学上，不引入折扣因子，累积回报值就是无穷大，无法进行数学分析。

有限和无限视界只是理论上的划分。实际上，可以把有限视界的强化学习的折扣因子看作 1，把无限视界的视界长度看作正无穷大，这样，强化学习的累积回报可以统一表示为：

$$R_t = \sum_{k=0}^{T} \gamma^k r_{t+k+1} \quad (3\text{-}3)$$

3.3 MDP 模型求解强化学习问题

本节介绍用 MDP 模型解决强化学习问题的推导过程。利用马尔可夫模型在推导强化学习解法时，除了涉及到前面提及的策略概念外，还涉及到值函数、Q 值函数、最优值函数、最优策略等一些基本的概念，下面会一一介绍。

3.3.1 马尔可夫模型

马尔科夫性又称无后效性，即在 t+1 时刻的状态仅取决于 t 时刻的状态和智能体在 t 时刻选择的动作，t 时刻之前（t-1,t-2，...,1）的状态和动作信息对环境没有额外影响，对 t+1 时刻的决策（动作选择）没有额外帮助。一个符合马尔科夫性的强化学习问题称作马尔科夫决策过程（Markov decision process，简称 MDP），而分析马尔科夫决策过程有马尔可夫模型。由于绝大多数的强化学习问题都具有马尔科夫性，所以可以说马尔可夫模型是强化学习的理论模型。强化学习中智能所处的环境往往是动态的、不确定的，经典的人工智能算法不能并应对这种复杂的环境，马尔可夫模型则十分适于动态不确定的复杂环境。

基本地，马尔可夫模型是一个四元组：<S，A，T，R>。其中，S 是进行强化学习智能体所处环境的所有状态的集合，称为状态空间，在某一时刻 t，系统只能处于某一个确定的状态 $s_t \in S$。A 是智能体可知选择的动作的集合，称为动作空间，在时刻 t，智能体选择一个动作 $a_t \in A$ 并执行这个动作。更一般地，定义特定状态的动作空间：为从状态空间到动作空间的幂集的映射，用 A(s)表示。

T 描述了环境状态转移的模型，形式上，T 是这样一个函数，T：S×A→P(S;S,A)，即从状态空间和动作空间的笛卡尔积到状态的概率分布的映射。这里，智能体在状态 s 下执行动作 a，然后环境状态转移到 s'的概率记作 p(s'|s,a)。R 是环境状态转移的同时伴随的回报的模型，它是这样一个函数，S：S×A×S→**R**，即智能体在状态 s 下执行动作 a，然后环境状态转移到 s'时，智能体获得的回报值。**R** 是实数集，回报值 r(s'|s,a)是一个实数。我们常用正数表示好的汇报，负数表示不好的回报，数值的绝对值越大程度越深。

前面给出了策略的定义，策略 π_t 指出在当前环境状态 s_t 应该选择的动作决策 a_t，考虑到更复杂的情形，策略还可以具有随机性，即策略在 s_t 下以概率从若干动作中选择一个动作。这里给出一般的策略定义：策略是一个从状态空间和动作空间的叉集到实数集的映射，即 π：S×A×S→R，π(s,a)表示该策略在状态 s 下以 π(s,a)的概率选择动作 a。

在马尔可夫模型的四个基本元素中，状态空间大小代表了环境的复杂性，反映了强化学习问题的复杂性。对于问题而言，状态空间中的不同状态是差别很大的，其中的某些状态是智能体想要达到或者希望维持的，某些状态则是智能体希望避免或想要尽量少面临的。动作空间则反映了智能体自身的能力情况，代表其主观能动性。而状态转移模型和回报值模型则代表了环境和智能体相互作用的客观规律，属于客观世界，因而也称作世界模型。所以强化学习的过程，也就是具有感知、有限行动能力、会反馈的智能体通过主观能动性，逐渐掌握客观规律，以学会利用客观规律达到自身目的的过程。

3.3.2 值函数和最优值函数

值函数 V^π 是从状态空间 S 到实数集 R 的映射，对于一个给定一个策略 π，对于每个状态 s，都存在一个值函数 $V^\pi(s)$，该值给出了智能体从状态 s 出发，遵循策略 π，最终所获得的累积回报值的期望。值函数体现了一个状态对智能体的“好坏”程度，值函数最大表明智能体获得的期望累积回报越大。值函数公式为：

$$V^\pi(s) = E_\pi\{R_t \mid s_t = s\} = E_\pi\left\{\sum_{k=0}^{T} \gamma^k r_{t+k+1} \mid s_t = s\right\} \tag{3-4}$$

Q 值函数 Q^π 是给定一个策略 π，智能体在状态 s 下选择动作 a 后，遵循该策

略 π，最终所获得的累积回报值的期望。Q 值函数也叫动作值函数。Q 值函数体现了一个状态下选择不同动作对智能体的“好坏”程度，Q 值函数最大表明选择该动作后智能体获得的期望累积回报越大。Q 值函数公式为：

$$\begin{aligned} Q^{\pi}(s, a) &= E_{\pi}\{R_t \mid s_t = s, a_t = a\} \\ &= E_{\pi}\left\{\sum_{k=0}^{T} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a\right\} \end{aligned} \tag{3-5}$$

根据值函数的定义，易得：

$$V^{\pi}(s) = \sum_{a} \pi(s, a) \sum_{s'} p(s' \mid s, a)[r(s' \mid s, a) + \gamma V^{\pi}(s')] \tag{3-6}$$

这就是有名的贝尔曼等式（Bellman equation），该等式给出了相邻状态间的值函数关系。当 π(s,a)恒为 1 时，有 π(s)=a，该公式简化为：

$$V^{\pi}(s) = \sum_{s'} p(s' \mid s, \pi(s))[r(s' \mid s, \pi(s)) + \gamma V^{\pi}(s')] \tag{3-7}$$

实际上，对于任意一个策略，贝尔曼等式给出了相邻状态的值函数的约束，每个状态下的值函数有唯一解。

策略 π 不次于策略 π'，是指对任意 $s \in S$，有 $V_{\pi(s)} \geq V_{\pi'(s)}$。事实上，总是至少存在一个策略，它不次于其他所有策略，把它称作最优策略，用 π*表示。最优策略对应的值函数称为最优值函数，用 V*表示，即

$$V^{*}(s) = \max_{\pi} V^{\pi}(s) \tag{3-8}$$

最优策略也对应最优动作值函数，用 Q*表示，即

$$Q^{*}(s, a) = \max_{\pi} Q^{\pi}(s, a) \tag{3-9}$$

根据最优值函数和最优动作值函数的定义，可得两者之间有如下关系：

$$Q^{*}(s, a) = E\{r_{t+1} + \gamma V^{*}(s_{t+1}) \mid s_t = s, a_t = a\} \tag{3-10}$$

在最优策略情况下，关于最优值函数和最优动作值函数，有贝尔曼等式：

$$V^{*}(s) = \max_{a \in A(s)} \sum_{s'} p(s' \mid s, a)[r(s' \mid s, a) + \gamma V^{*}(s')] \tag{3-11}$$

和

$$Q^*(s,a)=\sum_{s'}p(s'\mid s,a)[r(s'\mid s,a)+\gamma\max_{a'}Q^*(s',a')] \tag{3-12}$$

3.4 强化学习算法

本节首先介绍强化学习问题的三类基本解法：动态规划（Dynamic Programming，DP），蒙特卡罗算法（Monte Carlo，MC）和时序差分（Temporal Difference，TD）。它们均基于马尔科夫理论，每个方法的假设和适用情况不同。动态规划基于环境状态转移模型已知的假设，相关理论发展得最完善和严密；蒙特卡罗算法不需要已知环境的状态转移模型，但仅仅适用于有终任务；时序差分同时结合了蒙特卡罗算法和动态规划两者的优点，适用范围最广。

3.4.1 动态规划

动态规划算法基于问题符合 MDP 模型且其中环境的状态转移模型已知的假设，算法主要依据贝尔曼等式原理。动态规划算法主要包含策略评估和策略优化两个步骤，这两步交替循环进行，理论上最终能得到最优策略。

策略评估（Policy Evaluation, PE）指由一个给定的策略计算对应的状态值函数的过程，策略评估步骤的算法描述为：

1．输入：待评估的策略π

2．计算过程

1）初始化：

对状态集合 S 中的每一个状态 s，初始化对应的值函数 V(s)为 0

2）循环：

Repeat

对状态集合 S 中的每一个状态 s，对值函数进行如下更新：

$$V(s)=\sum_{a}\pi(s,a)\sum_{s'}p(s'\mid s,a)[r(s'\mid s,a)+\gamma V(s')],$$

然后计算所有状态更新一次前后值函数绝对值的变化δ

Until $\delta<\theta$（θ是使迭代结束的阈值，一般是一个小的正数）

3．输出：对应策略π的值函数V_π

其中核心步骤是依照如下公式对值函数进行更新：

$$V_{k+1}(s) = E_{\pi}\{r_{t+1} + \gamma V_k(s_{t+1}) \mid s_t = s\}$$
$$= \sum_a \pi(s,a) \sum_{s'} p(s' \mid s,a)[r(s' \mid s,a) + \gamma V_k(s')] \tag{3-13}$$

值函数 V 的下标代表迭代的轮数，每一轮对所有状态的值函数进行迭代，对于 k+1 轮的状态 s，其值函数依据他可能达到的后继状态在 k 轮的值函数加权更新。

策略优化（Policy Improvement, PI）指由策略评估得到的值函数对原有策略进行更新的过程。给定一个策略 π 的值函数 V^{π}，策略优化步骤可以依据以下公式完成：

$$\pi'(s) = \arg\max_a Q^{\pi}(s,a)$$
$$= \arg\max_a \sum_{s'} p(s' \mid s,a)[r(s' \mid s,a) + \gamma V^{\pi}(s')] \tag{3-14}$$

理论证明，假设由策略 π 的值函数 V^{π} 经策略优化步骤得到新的策略 π'，π' 总是优于或等于 π，即 π' 总是不比 π 差。

策略迭代（Policy Iteration）是一种通过交替进行策略评估和策略迭代，最终收敛到最优策略的算法。策略迭代算法的过程示意图如图 3-3 所示，其中 E 代表 Evaluation，即策略评估，I 代表 Improvement，即策略优化。

$$\pi_0 \xrightarrow{E} v_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} v_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \cdots \xrightarrow{I} \pi_* \xrightarrow{E} v_*$$

图 3-2 策略迭代算法的过程示意图

Figure 3-2 Process of Policy Iteration Algorithm

由于策略评估步骤需要多次交替进行策略评估和策略优化步骤，可能十分耗时。有人提出了通过先循环进行值函数更新得到最优值函数，然后根据策略优化步骤得出最优策略的算法，即值迭代（Policy Iteration）算法，值迭代的算法描述为：

1．计算过程：

1）初始化：

对状态集合 S 中的每一个状态 s，初始化对应的值函数 V(s)为 0

2）循环进行值函数更新：

Repeat

对状态集合 S 中的每一个状态 s，对值函数进行如下更新：

$$V(s)=\max_{a}\sum_{s'}p(s'|s,a)[r(s'|s,a)+\gamma V(s')]$$

然后计算所有状态更新一次前后值函数绝对值的变化 δ

Until $\delta < \theta$ （θ 是使迭代结束的阈值，一般是一个小的正数）

3）策略优化：

由上一步得到的 V^* 和式子

$\pi(s)=\arg\max_{a}\sum_{s'}p(s'|s,a)[r(s'|s,a)+\gamma V^*(s')]$ 计算最优策略 π^*

2．输出：问题的最优策略 π^*

其中核心步骤是依照如下公式对值函数进行更新：

$$\begin{aligned}V_{k+1}(s)&=\max_{a}E_{\pi}\{r_{t+1}+\gamma V_k(s_{t+1})|s_t=s\}\\&=\max_{a}\sum_{s'}p(s'|s,a)[r(s'|s,a)+\gamma V_k(s')]\end{aligned}\tag{3-15}$$

由于这里进行了 max 最值操作，使得循环进行值函数更新能收敛到最优值函数，这是与策略迭代算法中策略评估步骤的区别。

相对与传统的策略搜索和线性规划的方法，动态规划的算法时间复杂度要小很多。但动态规划局限于状态模型已知的强化学习问题，在面对实际应用时，由于问题状态空间很大，动态规划算法的计算量仍然很大。研究动态规划的主要意义是帮助理解马尔科夫模型的特性。

3.4.2 蒙特卡罗算法

蒙特卡罗算法把交互得到的即时回报的平均值作为值函数的估计，当任务完成一次后，计算出此次任务状态序列的每个状态 s 的长期回报 R(s,a)；多次执行任务后得到平均值更新各状态的值函数。蒙特卡罗算法类似于动态规划中的策略迭代算法，同样包括同样策略迭代和策略优化两个步骤。蒙特卡罗算法的策略优化步骤与动态规划算法一样，蒙特卡罗算法中策略评估步骤的算法描述为：

1．输入：待评估的策略 π

2．计算过程

1）初始化：

对状态集合 S 中的每一个状态 s，初始化对应的值函数 V(s)为 0、和对应的访问次数 Visit(s)为 0

2）循环：

Repeat

利用策略π生成一个训练段，

对训练段中每个经历的状态 s（以第一次出现为准）：

计算回报值 $R(s) = \sum_{k=i}^{T} \gamma^k r_k$（其中 i 为 s 状态出现的时刻），然后更新对状态 s 的值函数的评估

$$V(s) = \frac{V[s] \times Visit[s] + R(s)}{Visit[s] + 1}$$

End

3．输出：对应策略π的值函数 V_π

可以说蒙特卡罗算法是第一个真正意义上的强化学习算法，因为严格地说，动态规划是模型已知的“规划”问题，而蒙特卡罗算法才是模型未知的“学习”问题，而在现实问题中，几乎不存在环境模型已知的情形。蒙特卡罗算法利用有终任务在任务段内经历状态后回报值的平均作为对状态的最优值函数估计，是一重很好的思想。然而也看到，蒙特卡罗算法的局限在于无法处理无限视界问题，而现实中却有很多无限视界问题。

3.4.3 时序差分

和蒙特卡罗算法一样，时序差分不需要已知环境模型，直接从交互中获取对环境模型的评估；和动态规划一样，时序差分不要求任务是有终的、不需要等到任务段结束，而是直接在每一步动作执行后更新值函数。动态规划和蒙特卡罗算法都是通过每种经验获得对值函数的估计值，然后进行更新。实际上其值函数更新过程可以泛化为一般的形式：

$$V(s_t) \leftarrow V(s_t) + \alpha[R_t - V(s_t)] \tag{3-16}$$

其中 S_t 为 t 时刻所处的状态之意，R_t 代表当前时刻 t 下对状态 s 的值函数的

估计。在蒙特卡罗算法中，R_t为一个任务断后获得的对 s 状态下的值函数评估；在动态规划中，则

$$R_t = r(s'|s,a) + \gamma V(s_{t+1}) \tag{3-17}$$

可以看到，动态规划对 $V(s_{t+1})$部分进行估计，因为 $V(s_{t+1})$在当前计算时是未知的；蒙特卡罗算法则直接对 R_t 进行采样估计。时序差分是这样的，它像动态规划一样对 $V(s_{t+1})$部分进行估计，同时像蒙特卡罗算法一样进行采样。

时序差分中最经典的是在策略（on-policy）算法和离策略（off-policy）算法。在策略是指强化学习算法所要求解的策略同时也是用来产生新动作的策略，即学习过程中只涉及到一个策略。在策略算法的策略一般都是软策略，即对任务 $s \in S$，有 $\pi(s,a)>0$。离策略则是指强化学习算法所要求解的策略和产生动作的策略不是同一策略。离策略算法要求产生动作的策略是软策略，而目标策略则可以是确定性策略，例如是贪心的策略。下面介绍 Sarsa 算法和 Q 学习算法，它们分别是在策略算法和离策略算法的代表。

Sarsa 算法的算法描述为：

1．计算过程：

1）初始化：

对每一个状态 s 和动作 a 的组合，初始化对应的 Q 值函数 Q(s,a)为 0

2）循环进行值函数更新：

Repeat

设立一个起始状态 s，使用 Q 值函数对应的策略选择动作 a

Repeat

执行动作 a，获得回报 r，得到新的状态 s'

使用 Q 值函数对应的策略选择动作 a'

对 Q 值函数进行如下更新：

$$Q_{t+1}(s,a) = (1-\alpha)Q_t(s,a) + \alpha[r + \gamma Q_t(s',a')]$$

更新当前状态为 s'，当前动作为 a'

Until s 为终结状态

Until 策略收敛

3）策略优化：

由上一步得到的 Q^* 和式子

$\pi(s)=\arg\max_a Q^*(s,a)$ 计算最优策略 π^*

2．输出：问题的最优策略 π^*

Q 学习算法的算法描述为：

1．计算过程：

1）初始化：

对每一个状态 s 和动作 a 的组合，初始化对应的 Q 值函数 Q(s,a)为 0

2）循环进行值函数更新：

Repeat

设立一个起始状态 s

Repeat

使用 Q 值函数对应的策略选择动作 a

执行动作 a，获得回报 r，得到新的状态 s'

对 Q 值函数进行如下更新：

$$Q_{t+1}(s,a)=(1-\alpha)Q_t(s,a)+\alpha[r+\gamma\max_a Q_t(s',a)]$$

更新当前状态为 s'

Until s 为终结状态

Until 策略收敛

3）策略优化：

由上一步得到的 Q^* 和式子

$\pi(s)=\arg\max_a Q^*(s,a)$ 计算最优策略 π^*

2．输出：问题的最优策略 π^*

它们的区别在于，Sarsa 算法在 Q 值函数更新时采用转移到的状态 s'下实际选择的动作 a'对应的 Q 值函数 Q(s',a')作为 R_t 的一部分，而 Q 学习算法在 Q 值函数更新时采用状态 s'下估计最大 Q 值函数 Q(s',a')作为 R_t 的一部分。

时序差分是最能代表强化学习核心思想的算法，它同时结合了蒙特卡罗算法和动态规划两者的优点。和蒙特卡罗算法一样，时序差分不需要已知环境模型，

直接从交互中获取对环境模型的评估；和动态规划一样，时序差分不要求任务是有终的、不需要等到任务段结束，直接在每一步动作执行后更新值函数，支持在线学习。很多强化学习的实际应用场景或者具有很长的任务段，或者则是无限视界问题，动态规划和蒙特卡罗算法均无法应对这些场合，而时序差分则可以处理。

3.5 本章小结

本章首先介绍了强化学习的概念、特点及其在机器学习领域的地位，其次介绍了强化学习问题特点和相关概念，然后介绍了用马尔可夫模型求解强化学习问题的分析思路，最后介绍了三大类主流的强化学习算法。

第四章 高层抢球策略的强化学习

4.1 问题描述

Keepaway 中的关键问题是高层动作决策，抢球球员需要从对方所有传球路线中选择一条封堵。传统的动作决策采用手工策略，使用手工策略时，球员根据当前情形下各个球员间的位置角度关系，选择自己认为最应该拦截的路线。然而手工策略具有很大的主观性，常根据经验选取相关参数，不能保证最优；同时手工策略无法考虑所有的比赛情形，对比赛情形动态变化的适应能力差，从而导致抢球任务完成时间较长、抢断成功率较低。抢球任务完成时间（Task Finish Time）指一个训练段持续的时间长度，一般以一个服务器模拟仿真为单位。抢断成功率（Stealing Success Rate）指对于待统计的 N 个训练段，在给定时间限制 t 下抢球成功的训练段数 n 占总的训练段数 N 的百分比。

对 Keepaway 任务，Peter Stone 将 Sarsa 强化学习算法应用于控球球员的高层动作决策[13]，并利用线性映射技术巧妙地存储了 Q 值[14]，使控球球队中持球球员的高层动作决策得到优化，球员的传球路线更加科学，延长了控球队伍的控球时间。左国玉在 Peter Stone 的基础上，考虑到控球总会失败的特点，由单智能体杆平衡系统问题的回报函数得到启发，设计了一种新的惩罚式的回报函数，进一步优化了控球球队中持球球员的高层动作决策[15]。

然而目前尚无将强化学习应用于 Keepaway 任务中抢球球员动作决策的文献研究。Keepaway 中抢球和控球的任务目标相反，任务特点也有所不同，因而球队策略也存在区别。控球的特点是要求无球球员进行合理的无球跑动，同时持球球员选择合理的传球路线，抢球的特点则是要求抢球球员分工对控球球员进行压迫和逼抢[31]。控球任务对无球球员的跑动要求相对较低，研究重点是持球球员的传球决策；而对于抢球，离球最近的抢球球员的决策比较固定（例如，必须上前逼抢持球球员，否则球队很难抢下球），对于其他负责拦截传球路线的抢球球员的决策则具有研究价值。本文针对 Keepaway 中抢球任务的上述特点，研究将强化学习应用于抢球球员高层动作决策的问题，以缩短抢球任务完成时间和提高抢断成功率。

4.2 Keepaway 的高层动作和总体策略

4.2.1. 抢球球员的高层动作和总体策略

Peter Stone 根据人类足球的知识，通过封装球员的底层原子动作，为 Keepaway 任务定义了一系列的高层动作。由于本文针对高层策略进行研究，文中涉及的手工策略和强化学习策略均基于这些现有的高层动作。抢球球员使用的高层动作包括[25]：

（1）“踢球”：试图踢到球；

（2）“跑向球”：直接跑向球，以达到截球或控球的目的；

（3）“拦截路线(i)”：移动到对方当前控球球员 K_1 和对方其他球员 K_i 连线上的一个位置，以期截获对方传球，$2 \leqslant i \leqslant m$，其中 m 为控球球员数，下同。

在 Keepaway 中，如引言中所述，考虑到抢球任务的特点，本文的总体策略是：当一名抢球球员离球很近可以获得控球权时，使用“踢球”动作，这样就可以完成抢球任务；当控球球员在控球时，抢球球员中距离控球球员最近的那名抢球球员应该上前逼抢，即使用“跑向球”动作。对于以上这两种情况，总是采用这样的固定策略；而在其他情况下，动作决策分为传统手工策略和强化学习策略。抢球球员的总体策略用伪代码表示为：

Step 1:如果球在我的控制范围内，返回“踢球”动作返回；

Step 2:如果我是离球最近的那名抢球队员，返回“跑向球”动作；

Step 3:利用手工策略或进行强化学习，在{拦截路线(2)，拦截路线(3)，拦截路线(4)}中选择动作。

Step 3 中，传统手工抢球策略一般思路是：预测抢球球员最有可能的传球路线进行拦截，一般会选取具有最大安全角度的路线。

4.2.2. 控球球员的高层动作和总体策略

控球球员是 Keepaway 中与抢球球员对抗的另一方，控球球员的高层动作包括：

（1）“控球”：保持对球的控制并不让对方靠近球；

（2）“传球(i)”：将球传给第 i 名队友，$2 \leqslant i \leqslant m$；

（3）“跑位”：跑到一个不受对方紧逼的安全区域，给队友提供传球路线；

（4）“跑向球”：跑向球，以达到截球或控球的目的。

控球球员的总体策略用伪代码表示为：

Step 1:如果存在一个正在持球或可以更快踢到球的队友，返回“跑位”动作；

Step 2:如果球不在自身控制范围，返回“跑向球” 动作；

Step 3:如果 4m 内没有抢球球员，返回“控球”动作；

Step 4:选择具有最大传球角度的传球路线 i，返回“传球(i)” 动作；

4.3 Keepaway 中高层抢球策略的强化学习

4.3.1. 状态空间与动作空间

球场上的关键元素是球的位置、各个球员的位置和球场中心。由于 Keepaway 中的核心是控球权的争夺，所以球和球员间的绝对位置不是那么重要，关键的是球、球员及场地中心的一系列相对位置和角度关系[32]。据此，设 m 和 n 分别为控球和抢球球员人数，本文为状态空间定义以下五类分量：A 类：所有球员到球场中心的距离；B 类：其他球员到持球球员的距离；C 类：正在进行强化学习的球员到对方 m-1 条传球路线中点的距离；D 类：对于对方 m-1 条传球路线的中点，另外 n-1 抢球球员到该点的最近的距离；E 类：对于对方 m-1 条传球路线，以为持球球员顶点，另外 n-1 抢球球员的最小夹角。

设 K_1 为离球最近的控球球员；T_1 为当前进行强化学习的抢球球员，$T_2,T_3,..,T_n$ 依据各自距 K_1 的距离由远到近排序。定义 $MidK_i$ 是连结 K_1 和 K_i 的线段的中点，其中 $2\leqslant i\leqslant m$；K_i 依据 $MidK_i$ 距 T_1 的距离由远及近排序。设 dist(A,B) 为 A 和 B 的距离，ang(A,B,C)为∠BAC。这样，状态空间表示为以下分量：

A 类：$dist(K_1,C)$，$dist(K_2,C)$，…，$dist(K_m,C)$，$dist(T_1,C)$，$dist(T_2,C)$，…，$dist(T_n,C)$；

B 类：$dist(K_1, K_2)$，$dist(K_1, K_3)$，… ，$dist(K_1, K_m)$，$dist(K_1, T_1)$，$dist(K_1, T_2)$，… ，$dist(K_1, T_n)$；

C 类：$dist(T_1,MidK_2)$，$dist(T_1,MidK_3)$，… ，$dist(T_1,MidK_m)$；

D 类：$MIN\{dist(MidK_2,T_2)$，… ，$dist(MidK_2,T_n)\}$，

$MIN\{dist(MidK_3,T_2)$，…，$dist(MidK_3,T_n)\}$，

…，

$MIN\{dist(MidK_m,T_2)$，…，$dist(MidK_m,T_n)\}$；

E 类：$MIN\{ ang(K_2,K_1,T_2)$，…，$ang(K_2,K_1,T_n)\}$，

$MIN\{ang(K_3,K_1,T_2),\ldots,ang(K_3,K_1,T_n)\}$，

…，

MIN{ang(K_m,K_1,T_2), ang(K_m,K_1,T_n) }。

根据 4.2.1 节中的分析，在 Keepaway 过程中，对于一名抢球球员，如果他不是离球最近的抢球球员，则进行强化学习，在{拦截路线(2)，拦截路线(3)，…拦截路线(m)}中选择一个高层动作。{拦截路线(2)，拦截路线(3)，…拦截路线(m)}就是抢球球员进行强化学习的动作空间。

4.3.2. 回报值设计

抢球球员通过强化学习选择动作往往应该持续超过一个周期，因为如果每个周期都重新选择，抢球球员可能疲于在若干个拦截线路的拦截点之间往返，而未能实际对方传球造成威胁。因而本文让强化学习选择的高层动作持续执行多个周期，直到随着形势的变化该抢球球员变为离球最近的抢球球员，这样他采取固定的策略，上一次强化学习的动作才算执行完成。鉴于这样的设计，回报值应该在一个强化学习选择的动作结束之后再给出。Peter Stone 在控球球员高层动作决策的强化学习时，定义控球球员的回报值为动作持续的周期数[13]，这一指标正好与强化学习和手工策略转换的周期一致，在此启发下，定义抢球球员的回报值如公式(4-1)所示。

$$reward=\begin{cases}lastActionTime-currentTime+300, 任务完成\\ lastActionTime-currentTime, 任务继续\end{cases} \tag{4-1}$$

公式(4-1)中，currentTime 代表当前周期编号，lastActionTime 代表上次执行强化学习动作的周期编号。lastActionTime—currentTime 表示高层抢球动作所花周期的相反数，如果动作使抢球任务完成，给予额外的回报值 300。300 是一个很高的正回报，通过大量实验表明，一般一次抢球任务所需周期数总小于 300。如果一个动作没有抢下球，则动作越费时，给以越多的惩罚，任务完成则有高的正回报以鼓励使得任务完成的对应动作，这样抢球球员就会逐渐学会采取那些能尽快结束任务的动作。

4.3.3. 高层抢球策略的强化学习算法

Sarsa[33, 34]算法是由 Rummery 和 Niranjan 提出一种基于模型的强化学习算法，Peter Stone[13]在将强化学习应用于 Keepaway 中控球球员的决策时，使用 Sarsa 算法得到了很好的效果，本文将 Sarsa 算法应用于抢球球员的高层动作决策的学习。

在 RoboCup 2D 中，服务器和球员都是按照每 100ms 的周期离散处理的。球员在一个周期选择一个高层动作并最终执行后，并不能立即得到下一个周期的状态，而是要等到下一个周期开始。同样，对于抢球球员对应动作的回报值也是要在等待若干个周期后，直到从手工策略转到执行强化学习策略时，才能得到对应上次强化学习选择的动作的回报值。所以本文在应用 Sarsa 算法时，将强化学习过程分为“训练段开始”、“训练段中”和“训练段结束”3 个阶段，如图 4-1 所示。“训练段开始”只进行动作的选择，“训练段中”进行动作选择、并更新上一个动作的 Q 值，“训练段结束”只进行上一个动作的 Q 值更新。

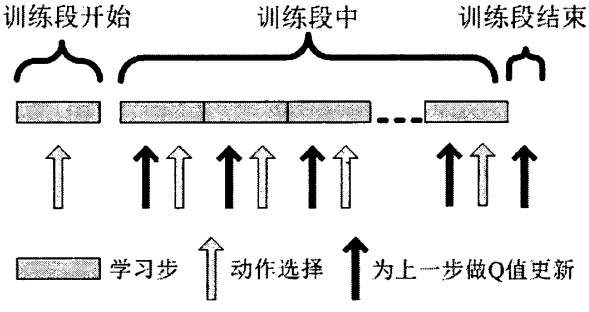


图 4-1 抢球球员的强化学习过程

Figure 4-1 Takers' Reinforcement Learning Process

抢球球员的强化学习 Sarsa 算法的伪代码如下：

Step 1：初始化状态 s 为当前球场状态，使用 ε-greedy 方法选择出动作 a，并记录动作选择的时间为当前周期；

Step 2：设置状态 s 下使用动作 a 的标记，执行动作 a；

Step 3：获得新的球场状态 s'，如果 s'不是任务结束状态，跳转至 Step 4；否则跳转至 Step 8；

Step 4：使用 ε-greedy 方法为 s'选择出动作 a'，并记录动作选择的时间为当前周期，同时根据公式(4-1)计算出状态 s'下的回报值 reward；

Step 5：为上一步的动作选择更新 Q 值：$Q(s,a) := (1-\alpha)\, Q(s,a) + \alpha\, [reward + \gamma\, Q(s',a')]$；

Step 6：设置状态 s'下使用动作 a'的标记，执行动作 a'；

Step 7：将 s 更新为 s'，a 更新为 a'，跳转至 Step 3；

Step 8：为上一步的动作选择更新 Q 值：$Q(s,a) := (1-\alpha)\, Q(s,a) + \alpha\, reward$

其中 Step 1- Step 2 为“训练段开始”阶段，Step 3- Step 7 为“训练段中”阶段，Step 8 为“训练段结束”阶段。Q 为 Q 值函数，α 是强化学习的学习率，γ 是折扣因子，reward 是立即回报值。

4.4 实验分析

4.4.1. 实验设置

为了分析强化学习在不同规模中的训练效果，实验对象采用最典型的 30m ×30m 场地下 4v3 和 5v4 规模的 Keepaway 任务。本文让抢球球员在不同学习率下进行强化学习训练，并统计任务完成时间和给定时间内的抢断成功率这两个指标随着训练过程的变化情况，与传统手工抢球策略下的训练过程进行比较。实验环境为 Ubuntu Linux3.5.0-17-generic，Intel x86_32 3.20GHz，2.00GB RAM。服务器设置开启 360 度球员视角和无噪声视觉模式。

Peter Stone 在控球球员的学习中使用了 0.125 的学习率[13]，为了进一步验证不同学习率下的强化学习的训练效果和收敛性，分别选取 0.125、0250、0.375 的学习率进行试验。在抢球球员的强化学习过程中，根据实验优化，设定折扣因子 γ 取 1，ε-greedy 动作选择策略的参数 ε 取 0.01。在统计抢断成功率时，任务段的总数 N 取 2000；根据实验结果统计，4v3 规模的时间限制取 75 个周期，5v4 规模时间限制取 65 个周期。场地大小设定为 30m×30m。

4.4.2. 实验结果和分析

图 4-2 和图 4-3 分别展示了 4v3 规模的 Keepaway 中任务完成时间和给定周期内的抢球成功率随着训练时间变化的情况，横坐标训练时间以小时（h）为单位。图 4-2 中，手工策略任务完成时间基本维持在 87 周期。0.125 学习率下，任务完成时间从 83 周期经学习最终稳定到 73 周期；0.250 学习率下，任务完成时间从 85 周期经学习最终稳定到 76 周期；0.375 学习率下学习曲线不够典型，基本维持在 78 周期。以手工策略为基准，对于任务完成时间， 0.125 学习率下降 16.1%，0.250 学习率下降 12.6%，0.375 学习率下降 10.3%，可见 0.125 的学习率最好。从图 4-3 看到，在 75 周期的时间限制下，手工策略的抢断成功率基本维持在 55.0%；三种学习率的强化学习抢断成功率最后都提升到 70.0%左右，提升了 15.0%。

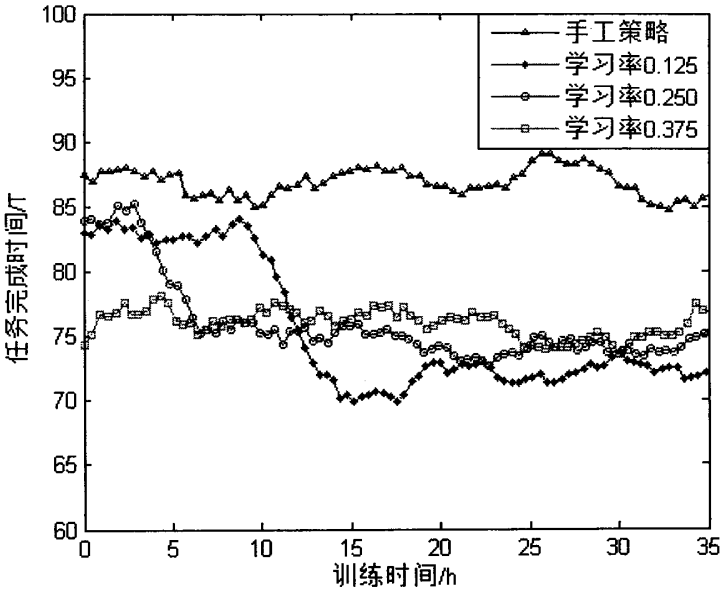


图 4-2 4v3 Keepaway 任务完成时间

Figure 4-2 Task Finish Time of 4v3 Scale Keepaway

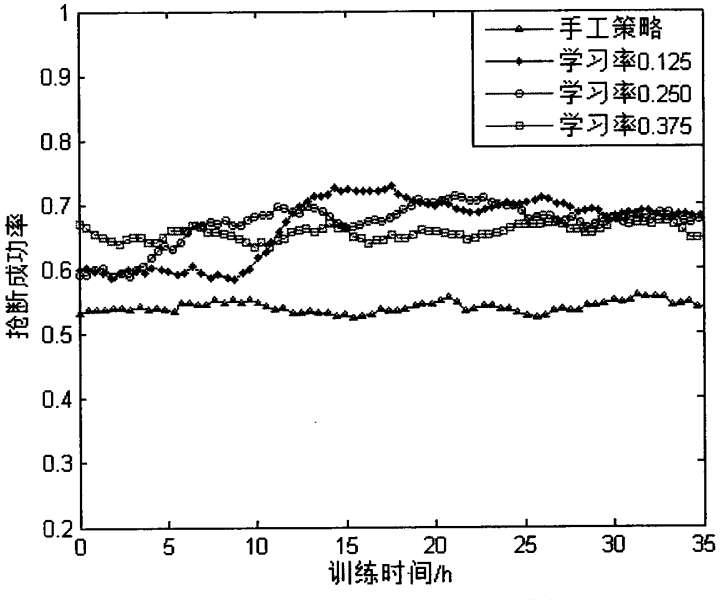


图 4-3 4v3 Keepaway 抢断成功率

Figure 4-3 Stealing Success Rate of 4v3 Scale Keepaway

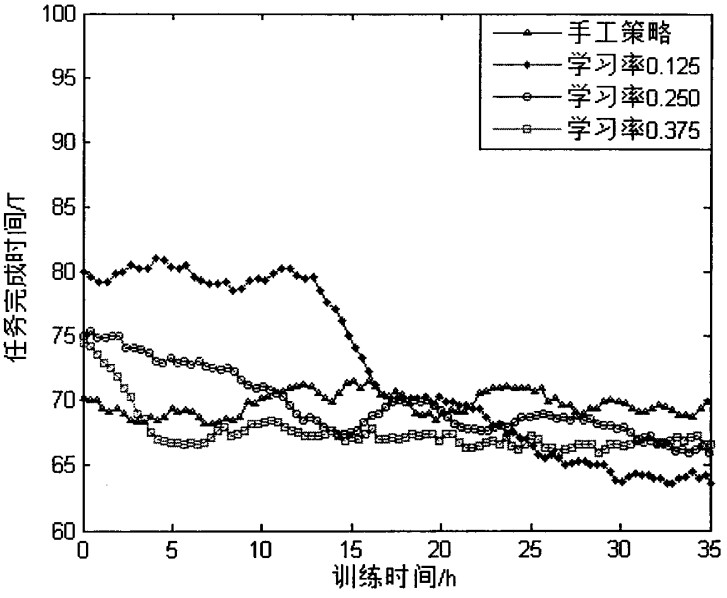


图 4-4 5v4 Keepaway 任务完成时间

Figure 4-4 Task Finish Time of 5v4 Scale Keepaway

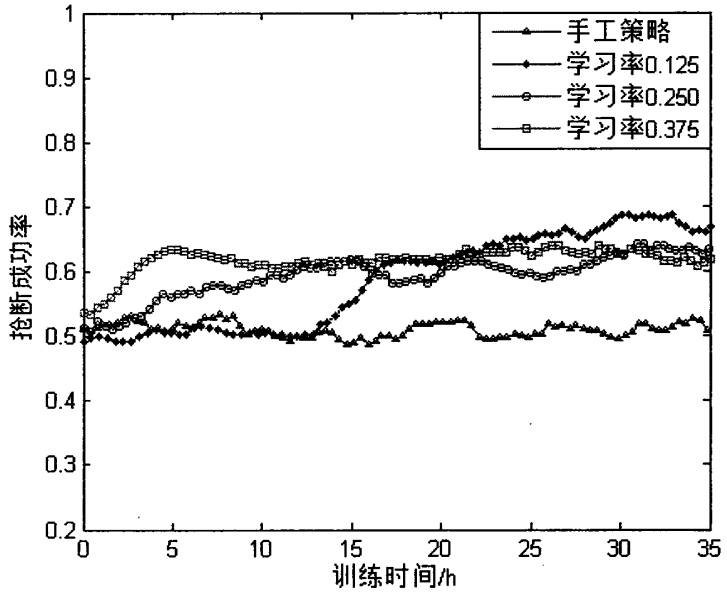


图 4-5 5v4 Keepaway 抢断成功率

Figure 4-5 Stealing Success Rate of 5v4 Scale Keepaway

图 4-4 和图 4-5 分别展示了 5v4 规模的 Keepaway 中任务完成时间和给定周期内的抢球成功率随着训练时间变化的情况。图 4-4 中，手工策略任务完成时间基本维持在 70 周期。0.125 学习率下，任务完成时间从 80 周期经学习最终稳定

到 65 周期；0.250 学习率下，任务完成时间从 75 周期经学习最终稳定到 68 周期；0.375 学习率下，任务完成时间从 74 周期经学习最终稳定到 68 周期。以手工策略为基准，对于任务完成时间，0.125 学习率下降 7.1%，0.250 学习率下降 2.9%，0.375 学习率下降 2.9%，可见 0.125 的学习率最好。从图 4-5 看到，在 65 周期的时间限制下，手工策略的抢断成功率为 52.0%；0.250 和 0.375 学习率的强化学习抢断成功率最后都提升到 63.0%左右，提升了 11.0%；0.125 学习率的强化学习抢断成功率最后都提升到 67.0%左右，提升了 15.0%，可见 0.125 的学习率最好。

从四幅图中看到，手工策略下任务完成时间和抢断成功率随着训练进行的变化不大。这是因为手工策略不具有学习和记忆能力，不能随着训练进行获取经验并提高决策。而进行强化学习时，不管使用何种学习率，学习收敛后抢球球员的任务完成时间缩短，抢断成功率提高，强化学习后的策略总优于手工策略。学习率越小，学习收敛后对应的抢球效果越好，在本实验中，0.125 的学习率下强化学习最终能获得最好的抢球策略。

4.5 本章小结

在 RoboCup Keepaway 任务训练中，传统手工抢球策略的主观性强，对训练情形变化的适应性差，导致抢球球员任务完成时间长、抢断成功率低。针对这一问题，将强化学习应用于 Keepaway 中抢球球员的高层动作决策。通过对抢球任务特点的分析，合理设计了抢球球员强化学习模型的状态空间、动作空间及回报值，并给出了抢球球员的强化学习算法。实验结果表明经强化学习后，抢球球员能够根据比赛情形做出更客观的决策，决策效果显著优于手工策略。对于 4v3 和 5v4 规模的典型 Keepaway 任务，抢球球员采用学习后的策略决策时，抢球任务完成时间至少缩短了 7.1%，抢断成功率至少提升了 15.0%。不同学习率下的强化学习对比实验，表明 Keepaway 中学习率是影响学习效果的关键因素，如何选取更优的学习率需要进一步研究。同时看到强化学习的达到收敛需要很长时间，训练很耗时，如何缩短训练时间有待研究。

第五章 高层抢球策略的任务间迁移学习

5.1 问题描述

第四章将强化学习的方法应用于 Keepaway 中抢球球员的动作决策，通过合理的状态空间、动作空间及回报值设计，使球员的决策随着训练的进行得到优化，抢球任务完成时间缩短，抢断成功率提高。然而由于 Keepaway 问题规模很大，强化学习需要很多步才能收敛，学习十分耗时，一般需要 10 个小时以上才能学到较好的策略，在这段时间内，抢球球员完成任务时间长，抢球效率很低。针对这一问题，Taylor[16]和 Fernández[17]对 Keepaway 中高层持球决策的普通强化学习进行延伸，通过使用策略重用技术，优化了高层持球策略的学习效率。策略重用是进行强化学习的智能体通过利用过去在类似任务中已经学得的策略来加快当前任务的学习进程以达到迁移学习的一项技术。策略重用时，由于具有先前类似任务的已有经验，智能体能快速地学得较好的策略。

然而目前尚无将策略重用应用于 Keepaway 问题中抢球动作决策迁移学习的文献研究。第四章述及，在 Keepaway 中，抢球和控球的任务目标相反，任务特点也有所不同，因而球队策略也存在区别。本文针对 Keepaway 中抢球任务的上述特点，以高层抢球决策的强化学习为基础，研究基于策略重用的抢球决策的迁移学习，以提高学习效率。

5.2 迁移学习和策略重用

5.2.1. 相关概念

迁移学习[35]通过找到相同领域内不同规模问题间的相似之处，利用已解决的较小规模问题策略来帮助解决较大规模问题的学习。迁移学习中问题虽然规模不同，但是往往有很多相似之处，例如对于 4v3 和 5v4 规模的 Keepaway 任务，在为这两个规模的抢球策略设计强化学习模型时，可以看到它们的状态空间、动作空间存在自然延伸的关系。

策略重用[35]是一项实现迁移学习的技术，指进行强化学习的智能体利用过去在类似任务中学到的策略，来帮助加速当前任务学习进程的技术。策略重用适用于分段任务的迁移学习。策略重用前需要智能体事先通过学习获得类似任务的策

略，作为待重用的旧策略。策略重用时智能体既有对新任务的强化学习，又有对在旧任务中学到的策略的重用，要求智能体具有在学习当前问题和利用过去策略之间平衡的机制。可以通过在学习过程中维护并更新代表每个策略重要性的权值、依权值作为依据概率性地选取每个策略的方法实现这个平衡机制。在具体利用旧策略时，由于新旧策略的问题空间规模不同，包括状态空间和动作空间规模上的差别，需要分别对新旧问题的状态空间和动作空间进行映射。可以通过分析新旧问题的状态空间和动作空间的特点，找到合理的映射方案。

5.2.2. 策略重用算法

我们知道，学习的过程包括很多个任务段，假设一个任务段的最大长度是 H，则学习的目标是最大化所有任务段平均回报值 W：

$$W = \frac{1}{K}\sum_{k=0}^{K}\sum_{h=0}^{H}\gamma^{h} r_{k,h} \tag{5-1}$$

其中 K 是执行的任务段的总数，$r_{k,h}$ 是任务段 k 中的第 h 步获得的立即汇报值 γ 是折扣因子。使任务 Ω 的回报值 W 最大化的策略为最优策略，对应的回报值 W 为 W_{Ω}^{*}。

策略重用技术旨在通过利用之前在类似任务中学到的策略加速当前学习的进程。形式化地，就是已知之前智能体通过对任务集$\{\Omega_1, \ldots, \Omega_n\}$的学习获得了对应的策略集$\{\Pi_1^*,..., \Pi_n^*\}$，现在面对任务 Ω，需要求解Π_{Ω}^{*}。前文述及，对此需要解决两个问题：一是如何从策略集$\{\Pi_1^*,..., \Pi_n^*\}$中选择一个策略Π_i^*；二是在选定Π_i^*后，怎么将该策略集成在当前的强化学习过程中。PRLearning（Policy Reuse Learning）算法解决了这两个问题[35]；PRLearning 算法中解决第二个问题的子过程，为 π-reuse 算法。

π-reuse 的算法流程如下：

1. 输入：旧策略Π_{past}，执行次数 K，任务视界 H，初始重用概率 ψ 和概率衰减指数 υ
2. 计算过程：
   1）初始化：

对每一个状态 s 和动作 a 的组合，初始化对应新任务的

Q 值函数 $Q_{new}(s, a)$ 为 0

2）变量 k 从 1 递增至 K：

设立一个起始状态 s，设定 $\psi_1 = \psi$

变量 h 从 1 递增至 H：

以 ψ_h 的概率使用旧策略选择动作，$a = \Pi_{past}(s)$

以 $1 - \psi_h$ 的概率使用当前策略选择动作，$a = \Pi_{new}(s)$

执行动作，获得回报 $r_{k,h}$，进入新的状态 s'，更新 Q 值

$$Q^{\Pi_{new}}(s,a) = (1-\alpha)Q^{\Pi_{new}}(s,a) + \alpha[r + \gamma \max_{a} Q^{\Pi_{new}}(s',a)]$$

更新 $\psi_{h+1} = \psi_h \upsilon$，更新当前状态为 s'

$$W = \frac{1}{K}\sum_{k=0}^{K}\sum_{h=0}^{H}\gamma^h r_{k,h}$$

3．输出：W，Q_{new}

其中 ψ（$0 < \psi < 1$）是控制学习在重用旧策略和进行当前问题学习间平衡的参数。ψ 值随着时间的推移指数减小，代表随着学习的进行逐渐减少重用旧策略的程度，这也符合实际情况。同时注意，不管是重用旧策略还是进行当前问题的学习，均只更新当前问题的 Q 值表。K 个任务段结束后，返回平均任务单回报值 W，作为调用该算法的主控程序的权值。K 值可以取 1，实际上 PRLearning 算法调用 π-reuse 算法时，K 即取 1。

PRLearning 的算法流程如下：

1．输入：需要学习的新任务 Ω，旧策略库{Π_1，…，Π_n}，执行次数 K，

任务视界 H，初始重用概率 ψ 和概率衰减指数 υ

2．计算过程：

1）初始化：

对每一个状态 s 和动作 a 的组合，初始化对应新任务的

Q 值函数 $Q_\Omega(s, a)$ 为 0，W_0（W_Ω），W_1，…，W_n 均赋初值为 0

2）变量 k 从 1 递增至 K：

$$P(\Pi_j)=\frac{e^{\tau W_j}}{\sum_{p=0}^{n} e^{\tau W_p}}$$

依据公式 算得的概率权重选择一个策略 Π_k

若 Π_k 为 Π_Ω，进行新策略学习，

否则调用 π-reuse(Π_k,1,H,ψ,υ)，并根据返回的 W 更新 W_k

3）策略优化：

由上一步得到的 Q^*_Ω 和式子 $\pi(s)=\arg\max_a Q^*_\Omega(s,a)$ 计算最优策略 π^*_Ω

3．输出：π^*_Ω

假设已经学到策略库{Π_1，…，Π_n}首先依据权值表 W_0，W_1，…W_n 和 soft-max 概率选择公式选出当前任务段 k 将要使用的策略 Π_k：

$$P(\Pi_j)=\frac{e^{\tau W_j}}{\sum_{p=0}^{n} e^{\tau W_p}} \tag{5-2}$$

如果选到的策略 Π_k 为 Π_Ω，则进行当前任务的强化学习，否则 Π_k 为{Π_1，…，Π_n}中之一，进行策略重用，即执行 π-reuse 算法，返回任务段 k 的回报值 R。然后算法根据回报值 R 更新策略 Π_k 对应的权值表 W_n。如此循环 K 个任务段，K 可以是无穷大。

5.3 Keepaway 中高层抢球策略的任务间迁移学习

本节针对 5v4 规模 Keepaway 问题，利用策略重用技术实现高层抢球策略的迁移学习，包括给出迁移学习方案，4v3 和 5v4 任务间状态与动作空间的映射以及基于策略重用的迁移学习算法。

5.3.1. 迁移学习方案

为了在 5v4 抢球训练中进行迁移学习，我们需要一个 4v3 抢球策略作为旧策略，这个旧策略可以通过进行 4v3 抢球训练的强化学习得到。我们可以为 5v4 任务中的 4 名抢球球员设置如表 1 的迁移学习方案：3 名抢球球员 T1，T2 和 T3 先进行 4v3 任务的普通强化学习，学到 4v3 任务下的抢球策略，然后在 5v4 任务中基于学到的 4v3 任务下的抢球策略进行迁移学习；第 4 名抢球球员 T4 从零开

始进行 5v4 任务的强化学习。

表 5-1 4v3 规模到 5v4 规模的策略迁移方案

Table 5-1 Policy Transfer Plan from 4v3 Scale to 5v4 Scale

球员 / 任务	抢球球员 T_1	抢球球员 T_2	抢球球员 T_3	抢球球员 T_4
4v3 规模	从零开始学习策略 $\Pi_{4v3}^{T_1}$	从零开始学习策略 $\Pi_{4v3}^{T_2}$	从零开始学习策略 $\Pi_{4v3}^{T_3}$	不参与
5v4 规模	通过重用策略库 $\{\Pi_{4v3}^{T_1}\}$ 学习策略 $\Pi_{5v4}^{T_1}$	通过重用策略库 $\{\Pi_{4v3}^{T_2}\}$ 学习策略 $\Pi_{5v4}^{T_2}$	通过重用策略库 $\{\Pi_{4v3}^{T_3}\}$ 学习策略 $\Pi_{5v4}^{T_3}$	从零开始学习策略 $\Pi_{5v4}^{T_4}$

5.3.2. 任务间映射

在 5v4 任务中进行策略重用时，旧策略是 4v3 规模的，而当前问题是 5v4 规模的。当球员更新获得一个 5v4 任务空间 S_{5v4} 的环境状态 s_{5v4} 时，为了利用旧策略，需要将它映射为一个 4v3 问题空间 S_{4v3} 的世界状态 s_{4v3}；这样可以利用旧策略得到在旧问题空间下的解，假设得到对应动作决策 a_{4v3}，该解属于 4v3 问题的动作空间 A_{4v3}；还要把 a_{4v3} 映射回当前 5v4 规模问题的动作空间 A_{5v4} 下，得到 a_{5v4}，作为重用策略得到的动作决策。可以看到，在策略重用时，要进行任务间的两次映射，第一次是新状态空间到旧状态空间的映射 ρ_S：$S_{5v4} \rightarrow S_{4v3}$，第二次是旧动作空间到新动作空间的映射 ρ_A：$A_{4v3} \rightarrow A_{5v4}$。

对于 ρ_S，由于抢球训练的状态空间在定义时是选取一些关键的相对量，状态维度随着任务规模的扩大而自然延伸，所以较小规模任务的状态分量总包含在较大规模任务的状态向量之中，映射是只需进行投影即可。对于 ρ_A，这里选取简单的策略 $\rho_A(a)= a$，就是依旧策略选择的动作作为最终的决策。

5.3.3. 基于策略重用的迁移学习算法

在 4v3 规模 Keepaway 中通过强化学习得到策略 Π_{4v3} 后，抢球球员在 5v4 规模 Keepaway 中进行迁移学习。迁移学习算法在本质上属于强化学习算法，它与普通强化学习的不同在于，学习过程中动作决策不单由当前学到的策略给出，还可能由重用旧策略给出。本文在第四章普通强化学习算法 Sarsa 的基础上，给出抢球球员的迁移学习算法 PR-Sarsa 算法，算法步骤如下：

Step 1：初始化 Q 值表，使表中的所有值为接近 0 的随机值；初始化权重值

表，$W_{\Omega}=W_{\Pi 4v3}=0$，（Ω 代表当前学习的策略）；初始化训练段的编号 k 为 0；

Step 2：训练段的编号加 1，s 初始化为当前环境状态；

Step 3：依照当前权重表和 soft-max 概率选择公式(5-2)选出当前任务段 k 将要使用的策略 Π_k，如果 Π_k 是 Π_{Ω}，进行正常的强化学习，否则重用策略 Π_k，选择动作 a；

Step 4：执行动作 a，观察回报值 r 和新的环境状态 s'；

Step 5：依照当前权重表为依据随机选择一个策略 Π_k，如果 Π_k 是 Π_{Ω}，进行正常的强化学习，否则重用策略 Π_k，选择动作 a'；

Step 6：更新 Q 值：$Q_t(s, a):=(1 - \alpha)Q_{t-1}(s, a) + \alpha [r + \gamma Q_{t-1}(s', a')]$；根据 r 更新权重值表；

Step 7：将 s 更新为 s'，a 更新为 a'；

Step 8：如 s 是任务结束状态，转到 Step 2；否则转到 Step 5。

Step 1 是初始化操作。Step 3 首先依据权值表 W_0，W_1 和 soft-max 概率选择公式实现概率性地进行当前策略的学习和旧策略的重用，并做出动作决策。Step 4 执行动作，观察新的环境状态。Step 5 为新的状态选择动作，暂不执行动作。Step 6 为上一步更新 Q 值表，并且根据回报值 r 更新权值表 W_0，W_1。Step 7 进入下一周期，更新环境状态。Step 8 根据当前环境状态决定继续当前训练段还是开始新的训练段。

5.4 实验分析

5.4.1. 实验设置

为了分析基于策略重用的迁移学习的训练效果，实验对象采用最典型的 30m×30m 场地下的 5v4 规模 Keepaway 任务。本文让 5v4 任务中高层抢球策略的迁移学习与普通强化学习过程进行比较。实验分为两步：第一步，进行 4v3 任务中高层抢球策略的普通强化学习；第二步，利用第一步学到的策略 Π_{4v3}，进行 5v4 任务中基于策略重用的高层抢球策略的迁移学习。

实验环境为 Ubuntu Linux3.5.0-17-generic，Intel x86_32 3.20GHz，2.00GB RAM。设定场地大小设定为 30m×30m。服务器设置开启 360 度球员视角和无噪声视觉模式。在抢球球员的强化学习过程中，根据实验优化，设定折扣因子 γ 为 1.0，ε-greedy 动作选择策略的参数 ε 为 0.01，学习率 α 为 0.125。

5.4.2. 实验结果和分析

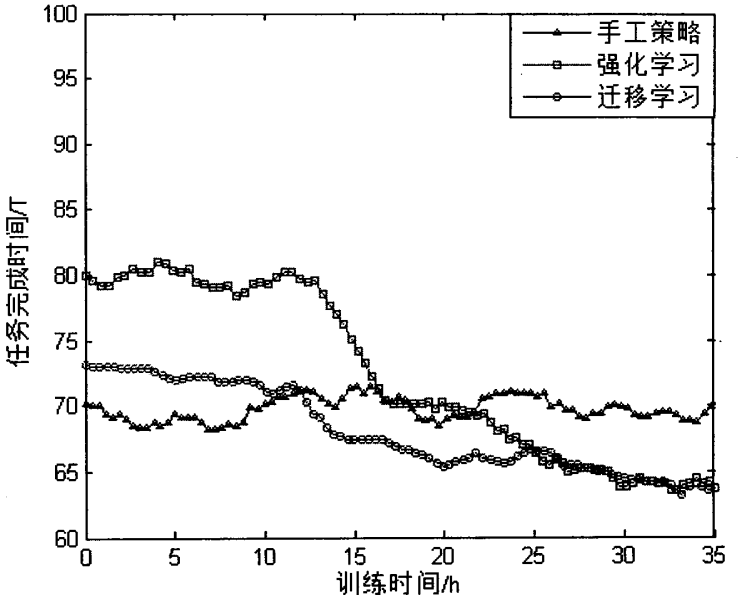


图 5-1 5v4 Keepaway 任务完成时间
Figure 5-1 Task Finish Time of 5v4 Scale Keepaway

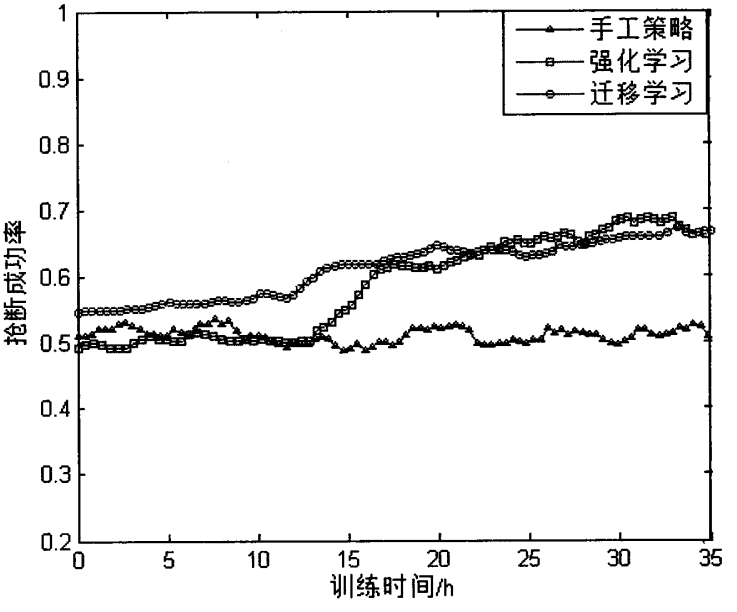


图 5-2 5v4 Keepaway 抢断成功率
Figure 5-2 Stealing Success Rate of 5v4 Scale Keepaway

图 5-1 和图 5-2 分别展示了 5v4 规模 Keepaway 手工策略、普通强化学习和迁移学习的任务完成时间和抢断成功率随着训练时间变化的情况，横坐标训练时间以小时（h）为单位。手工策略下任务完成时间基本维持在 70 周期，抢断成功率维持在 52.0%；普通强化学习下，任务完成时间从 80 周期经学习最终稳定到 65 周期，抢断成功率提升到 67.0%左右。手工策略下任务完成时间随着训练进行的变化不大，这是因为手工策略不具有学习和记忆能力，不能随着训练进行获取经验并提高决策。而强化学习时收敛后策略明显优于手工策略，任务完成时间较手工策略缩短 7.1%，抢断成功率提高 15.0%。

但是同时从图 5-1 看到普通强化学习的任务完成时间在开始的 13 个小时内维持在 80 个周期左右，到 13 小时后才出现显著下降；而迁移学习的任务完成时间在开始时就比普通强化学习少 8 个周期，相对少 8.8%；在学习 13 个小时候就已降低到 70 个周期，普通强化学习完成任务时间降低到 70 个周期则需要 22 个小时。从图 5-2 看到，在训练开始的 18 个小时内，迁移学习的抢断成功率均比普通强化学习高 5.0%。相对于普通强化学习，基于策略重用的迁移学习在训练开始时就获得较优的决策，在相同训练时间的约束下，迁移学习比强化学习获得较优的高层策略，达到同样训练效果所需的时间总是较短。这是因为迁移学习利用了有助于解决当前问题的相关策略，实现了经验的借鉴。

5.5 本章小结

在 RoboCup Keepaway 中，球员使用强化学习能获得很好的高层策略。然而由于 Keepaway 任务的状态空间巨大，强化学习需要探索很多步才能收敛，学习过程十分耗时。针对这一问题，对 5v4 规模的 Keepaway 任务，将策略重用技术应用于抢球球员高层决策的强化学习中，实现了迁移学习。实验表明，对 5v4 任务，与普通强化学习相比，进行迁移学习的抢球球员其任务完成时间在训练开始阶段就相对少 8.8%，在训练时间相同的条件下，迁移学习总能学得较优的策略。为达到比较理想的策略水平，迁移学习所需的训练时间比强化学习显著减少。而如何优化策略重用时不同规模任务间状态空间和动作空间的映射，使智能体的动作选择更加合理，可以作为下一步的研究方向。

第六章 总结和展望

6.1 全文工作总结

在当前人工智能研究方向由“单主体静态可预测环境中的问题求解”向“多主体动态不可预测环境中的问题求解”过渡的背景下，RoboCup 2D 提出了一个当前人工智能的标准问题。

对于 RoboCup 2D 中最具代表性的子问题 Keepaway，本文针对抢球任务，进行了高层策略的研究，主要工作如下：

（1）介绍了 RoboCup 2D 研究平台的问题模型，通过分析可以看到 RoboCup 2D 是当前人工智能的标准问题；

（2）介绍了强化学习的概念、由来、发展和应用情形，介绍了分析强化学习问题的理论模型马尔可夫模型，并介绍了马尔可夫模型解决强化学习问题的推导过程和主流的强化学习算法；

（3）针对传统手工策略效率低的问题，通过对 Keepaway 中抢球任务特点的分析，合理设计了抢球球员强化学习模型的状态空间、动作空间及回报值，并给出了抢球球员的强化学习算法，使球员的决策随着训练的进行得到优化，抢球任务完成时间缩短，抢断成功率提高。

（4）针对较大规模 Keepaway 任务为得到较优策略进行普通强化学习耗时太长的问题，利用策略迁移技术，通过合理设计从较小规模到较大规模 Keepaway 抢球任务的迁移学习方案，以及定义两个规模的任务间状态及动作空间映射，并给出抢球球员的迁移学习算法，使抢球球员在较大规模 Keepaway 训练中重用在较小规模 Keepaway 中通过普通强化学习得到的高层策略，实现迁移学习。实验表明迁移学习在训练开始时就表现出较高的决策效率，并且比从零开始的普通强化学习更快地收敛到理想的策略水平，大大缩短了训练时间。

本文的研究具有一定的学术价值和应用价值。在学术价值方面，首先，本文的研究成果表明强化学习可以用来解决 Keepaway 中的抢球决策问题。其次，本研究通过在高层合理定义对应高层动作的回报值模型，将强化学习应用于高层决策中，实验结果证明了其有效性。我们知道，传统上强化学习只适用于解决底层动作的优化问题，本研究则证明了强化学习亦可以用来解决高层动作决策问题，

展示了强化学习更广泛的应用能力，扩展了对强化学习解决问题范围的认知。

在应用价值方面， Keepaway 问题从整个 RoboCup 2D 问题中提炼出来，具有独立性和代表性：一方面，Keepaway 问题相对简单，控球方只考虑维持控球权，抢球方只考虑争夺控球权，问题相对简化；另一方面，Keepaway 问题涉及到了足球比赛中核心的问题，即自主智能体的队内合作和队间对抗，对其的研究有助于整个 RoboCup 2D 问题的解决。而 RoboCup 2D 问题自身涉及通往未来智能化社会的关键技术，十分切合当前现代工业的应用实际。所以本研究有助于人工智能新理论向技术和产业的转化，具有巨大的应用价值。

6.2 未来展望

本文针对 RoboCup 2D 中典型子问题 Keepaway 中的抢球任务，利用强化学习和策略重用的方法，对高层策略进行了研究。实际上 RoboCup 2D 的研究还有很多工作可以继续开展：

（1）优化策略迁移技术中任务间状态及动作空间映射的定义，使智能体进行强化学习时选择的动作得到优化。

（2）将高层决策和底层决策贯穿起来，进行分层强化学习的研究。

（3）将 Keepaway 子问题进行延伸，抽象出更加复杂的 RoboCup 2D 子任务，考察强化学习能否处理这样的问题。

（4）考虑多智能体强化学习模型。本文使用的是单智能体强化学习的模型，RoboCup 2D 实际上涉及到球员智能体间通信和合作，也可以从多智能体的角度进行研究。

参考文献

[1] 科大蓝鹰球队网站. 仿真机器人足球：设计与实现[EB/OL]. http://www.wrighteagle.org/2d. 2012-12-23.

[2] 郝晓云. 多智能主体系统的社会规范[J].重庆工学院学报(社会科学版), 2009, 23(6): 75-78.

[3] 仵博, 吴敏, 刘兴东, 聂哲. Multi-Agent层次协作模型及其在RoboCup中的应用[J]. 计算机工程与设计, 2004, 07: 1069-1071.

[4] Chen M, Klaus D, Ehsan F. User Manual: RoboCup Soccer Server Manual for Soccer Server Version 7.07 and Later. [EB/OL]. http://sourceforge.net/projects/sserver/files. 2010-12-24.

[5] 范长杰. 基于马尔可夫决策理论的规划问题的研究[D]. 中国科学技术大学, 2008.

[6] Bai A, Wu F, Chen X. Towards a Principled Solution to Simulated Robot Soccer[J]. RoboCup-2012: Robot Soccer World Cup XVI, Lecture Notes in Artificial Intelligence, 2013, 7500: 1-12.

[7] Bai A, Wu F, Chen X. Online planning for large MDPs with MAXQ decomposition[J]. AAMAS 2012 Workshop on Autonomous Robots and Multirobot Systems, 2012.

[8] Bai A, Chen X, MacAlpine P. WrightEagle and UT Austin Villa: RoboCup 2011 Simulation League Champions[J]. RoboCup-2011: Robot Soccer World Cup XV, Lecture Notes in Computer Science, 2012, 7416: 1-12.

[9] Hidehisa A. RoboCup Simulation 2D Guide Book[EB/OL]. http://sourceforge.jp/projects/rctools. 2005-08-05.

[10] Bertsekas D. Dynamic Programming and Optimal Control [M]. Nashua: Athena Scientific, 2012: 493-509.

[11] Sutton R S, Barto A G. Reinforcement Learning: an Introduction [M]. Cambridge, MA: The MIT Press, 2012.

[12] Riedmiller M, Gabel T, Hafner R. Reinforcement Learning for Robot Soccer [J]. Autonomous Robots, 2009, 27(1): 55-73.

[13] Stone P, Sutton R S, Kuhlmann G. Reinforcement Learning for RoboCup Soccer Keepaway [J]. Adaptive Behavior, 2005, 13(3): 165-188.

[14] Sherstov A A, Stone P. Function Approximation Via Tile Coding: Automating Parameter Choice in Abstraction, Reformulation and Approximation [M]. Berlin: Springer Verlag, 2005: 194-205.

[15] 左国玉, 张红卫, 韩光胜. 基于多智能体强化学习的新强化函数设计 [J].控制工程, 2009, 16(2): 239-242.

[16] Taylor M, Stone P, Liu Y. Transfer Learning via Inter-task Mappings for Temporal Difference Learning[J]. Journal of Machine Learning Research), 2007, 8 (1): 2125–2167.

[17] Fernández F, García J, Veloso M. Probabilistic Policy Reuse for Inter-task Transfer Learning[J]. Robotics and Autonomous Systems, 2010, 58(7): 866-871.

[18] 连晓峰, 张弢, 刘载文, 苏维钧. RoboCup中型组机器人足球相关技术研究[J]. 机器人技术与应用, 2009, 03: 35-40.

[19] 何泽宇, 付庄, 曹其新, 陈卫东. 具有输入饱和特性的中型足球机器人运动控制研究[J]. 计算机工程与应用, 2003, 18: 105-107.

[20] 王书理. 机器人足球系统改进与实现的研究[D]. 燕山大学, 2006.

[21] 魏涛. RoboCup仿真球队的研究与实现[D]. 南京理工大学, 2007.

[22] 李实, 徐旭明, 叶榛, 孙增圻. 国际机器人足球比赛及其相关技术[J]. 机器人, 2000, 05: 420-426.

[23] 宁春林, 田国会, 尹建芹, 路飞, 李晓磊. 机器人足球比赛及其发展[J]. 山东大学学报(工学版), 2002, 05: 480-484.

[24] 李实, 徐旭明, 叶榛, 孙增圻. 机器人足球仿真比赛的Server模型[J]. 系统仿真学报, 2000, 02: 54-57.

[25] Stone P, Kuhlmann G, Taylor M E and Liu Y X. Keepaway Soccer: from Machine Learning Testbed to Benchmark in RoboCup 2005: Robot Soccer World Cup IX[J]. Berlin: Springer Verlag, 2006: 72-85.

[26] 张汝波, 顾国昌, 刘照德, 王醒策. 强化学习理论、算法及应用[J]. 控制理论与应用, 2000, 05: 637-642.

[27] 陈学松, 杨宜民. 强化学习研究综述[J]. 计算机应用研究, 2010, 08: 2834-2838.

[28] 高阳, 陈世福, 陆鑫. 强化学习综述[J]. 自动化学报, 2004, 30(1): 86-100.

[29] 毛俊杰, 刘国栋. 基于先验知识的改进强化学习及其在MAS中应用[J]. 计算机工程与应用, 2008, 24: 156-158.

[30] 连传强, 徐昕, 吴军, 李兆斌. 面向资源分配问题的Q-CF多智能体强化学习[J]. 智能系统学报, 2011, 02: 95-100.

[31] 范建明. 机器人足球防御仿真中强化学习方法的研究[D]. 大连理工大学, 2006.

[32] 闵华清, 曾嘉安, 罗荣华, 朱金辉. 一种状态自动划分的模糊小脑模型关节控制器值函数拟合方法[J]. 控制理论与应用, 2011, 02: 256-260.

[33] Rummery G A, Niranjan M. On-Line Q-learning using Connectionist Systems[R].Cambridge, England: Cambridge University Engineering Department, 1994.

[34] 殷锋社. 基于知识的Agent强化学习算法分析与研究[J]. 电子设计工程, 2011, 11: 115-117.

[35] 覃姜维. 迁移学习方法研究及其在跨领域数据分类中的应用[D]. 华南理工大学, 2011.

[36] Fernández F, Veloso M. Probabilistic Policy Reuse in a Reinforcement Learning Agent[C]. Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems. AAMAS'06, 2006.

[37] Walsh T J, Li L, Littman M. Transferring State Abstractions between MDPs[C]. in Proceedings of the ICML'06 Workshop on Structural Knowledge Tranfer for Machine Learning, 2006.

[38] Taylor M E, Stone P. Behavior Transfer for Value-function-based Reinforcement Learning[C]. The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems, 2005.

[39] Fernández F, Veloso M. Policy Reuse for Transfer Learning Across Tasks with Different State and Action Spaces[C]. ICML'06 Workshop on Structural Knowledge Transfer for Machine Learning, 2006.

[40] 章慧龙, 李龙澍. Q学习在RoboCup前场进攻动作决策中的应用[J]. 计算机工程与应用, 2013, 49(7): 240-242.

[41] Riedmiller M, Gabel T, Hafner R. Reinforcement Learning for Robot Soccer [J]. Autonomous Robots, 2009, 27(1): 55-73.

[42] Gabel T, Riedmiller M. On Progress in RoboCup: the Simulation League Showcase in RoboCup 2010: Robot Soccer World Cup XIV [J]. Berlin: Springer Verlag, 2011: 36-47.

[43] Kalyanakrishnan S, Liu Y, Stone P: Half Field Offense in RoboCup Soccer: a Multi-agent Reinforcement Learning Case Study in RoboCup 2006: Robot Soccer World Cup X [J]. Berlin: Springer Verlag, 2007: 72-85.

[44] 程显毅, 朱倩. 一种改进的强化学习方法在RoboCup中的应用[J].广西师范大学学报(自然科学版), 2010, 18(2): 21-26.

[45] 刘春阳, 谭应清, 柳长安. 多智能体强化学习在足球机器人中的研究与应用[J].电子学报, 2010, 38(8): 1958-1962.

[46] 李瑾, 刘全, 杨旭东. 一种改进的平均奖赏强化学习方法在RoboCup训练中的应用[J].苏州大学学报, 2012, 28(2): 21-26.

[47] Stone P, Sutton R S. Scaling Reinforcement Learning toward RoboCup Soccer [C]. In the Eighteenth International Conference on Machine Learning. Massachusetts: Williamstown, 2001: 537-544.

[48] Whiteson S, Kohl N, Miikkulainen R. Evolving Soccer Keepaway Players Through Task Decomposition [J].Machine Learning, 2005, 59(1): 5-30.

[49] Kalyanakrishnan S, Stone P. Characterizing Reinforcement Learning Methods Through Parameterized Learning Problems [J]. Machine Learning, 2011, 84(1-2): 205-247.

[50] Sutton R S, DoinaPrecup, SatinderSingh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning[J]. Artificial Intelligence, 1999, 112(1-2): 181–211.

[51] Hansen E A, Zilberstein S. Lao*: A heuristic search algorithm that finds solutions with loops[J]. Articial Intelligence, 2001, 129(1-2): 35-62.

致 谢

不经意间，三年的研究生学习生涯就要结束了；蓦然回首，不得不感慨时光荏苒，如白驹过隙。回顾这期间的学习和生活经历，自己在专业理论和实践方面都有了提升，要感谢三年里结识的良师益友。

首先，本论文是在导师李学俊副教授的悉心指导下完成的。李老师严谨务实的作风和追求卓越的态度，是我学习的榜样。李老师一丝不苟地指导我的论文工作，在研究思路和研究方法方面给出了很多重要的建议，并耐心指导我的文献撰写工作，使我的宏观把握能力和书面表达能力都有了很大的提高。在生活中，李老师十分亲切，他认真负责的态度、团结合作的意识、助人为乐的精神，也一直给我很大的感染和启迪。总之，李老师给了我太多的教导、帮助和关怀，他使我进步很多，在此表示衷心的感谢和崇高的敬意。

其次，感谢李龙澍教授对我研究方向的支持，以及对我的指导和关心。感谢杨为民老师和罗罹老师在机器人足球 2D 项目方面给我的指导。感谢赵鹏、徐怡、刘慧婷、贾兆红等老师的帮助和教诲。

然后，我要感谢组里的同学，马飞、刘涛、朱伟伟、王占东、李丹、孟宪龙、许恒飞、周开申、王建、方园和方菲，以及我的宿舍好友姜锟、王浩，他们在生活和学习中都给了我很多帮助。

再次，我要感谢我的家人。父亲和母亲给了我生命并用一生向我付出，姐姐也给了我数不清的付出和教导，姐夫给我很多帮助和中肯的意见，外甥带给我很多快乐。他们为我完成学业提供了物质保障，也是精神动力。

最后，向评阅本文的所有老师、专家和学者致以最诚挚的敬意。

攻读硕士学位期间的学术论文、科研项目与相关奖项

1. 攻读硕士学位期间的学术论文

[1] 李学俊,陈士洋,张以文,李龙澍.基于强化学习的RoboCup Keepaway高层抢球策略[J].计算机应用与软件.2014.（已录用）

[2] 李学俊,陈士洋.RoboCup仿真2D实验平台[J].实验室研究与探索.2014, 33(4):58-61.

2. 攻读硕士学位期间的科研项目

[1] 陈士洋, 李学俊, 杨为民. 机器人2D足球中智能决策规划研究. 安徽大学研究生学术型创新项目.2013.

3. 攻读硕士学位期间的相关奖项

[1] 陈士洋, 胡开亮, 陶成亮. 2013中国机器人大赛暨RoboCup公开赛RoboCup仿真组（2D）项目比赛一等奖.2013.

[2] 赵发君, 陈士洋, 陈亮, 张胜龙, 陈伟健. 2012中国机器人大赛暨RoboCup公开赛RoboCup仿真组（2D）项目比赛二等奖.2012.