硕士学位论文

面向非完全信息博弈的并行多任务强化学习方法研究

RESEARCH ON PARALLEL MULTI-TASK REINFORCEMENT LEARNING METHOD FOR INCOMPLETE INFORMATION GAME

郭振阳

哈尔滨工业大学

2020 年 12 月

国内图书分类号：TP391.4　　学校代码：10213
国际图书分类号：004.8　　密级：公开

工学硕士学位论文

面向非完全信息博弈的并行多任务强化学习方法研究

硕士研究生：郭振阳

导　　师：王轩教授

申请学位：工学硕士

学　　科：计算机科学与技术

所在单位：哈尔滨工业大学（深圳）

答辩日期：2020年12月

授予学位单位：哈尔滨工业大学

Classified Index: TP391.4
U.D.C: 004.8

A dissertation submitted in partial fulfillment of
the requirements for the academic degree of
Master of Engineering

RESEARCH ON PARALLEL MULTI-TASK REINFORCEMENT LEARNING METHOD FOR INCOMPLETE INFORMATION GAME

Candidate: Zhenyang Guo
Supervisor: Professor Xuan Wang
Academic Degree Applied for: Master of Engineering
Specialty: Computer Science and Technology
Affiliation: Harbin Institute of Technology, Shenzhen
Date of Defence: December, 2020
Degree-Conferring-Institution: Harbin Institute of Technology

摘 要

随着人工智能技术的发展，在越来越多的场景中可以看到人工智能技术的实际应用。作为和人类真实生活最贴近的人工智能研究领域，机器博弈也受到了研究者们广泛的关注。其中非完全信息的机器博弈场景因为信息感知不完全导致的复杂性以及更符合人类生活的实际规律，成为机器博弈中研究的焦点。深度强化学习作为研究机器博弈的一种有力的方法，已经在各种博弈环境中被应用，并在一些领域取得了重大的突破。

本文主要研究非完全信息条件下强化学习智能体如何在多个任务场景中取得优秀性能的问题。针对传统的单任务智能体泛化性差的问题，提出了并行学习的多任务强化学习算法，并行的在多个场景上训练智能体，使其在多个环境上均表现优异。为了加快训练的速度，获得更好的训练性能，本文结合行动者-学习者结构以及异步的行动者-评论者算法，提出了并行多任务强化学习的整体算法。通过对数据采集和知识训练的解构，使得训练性能大幅上升。同时算法针对因为解构导致的离线训练中数据延迟不匹配的问题，提出了分别针对学习者和行动者的数据校正方案，通过数据校正来修正模型学习的数据的正确性，提升智能体的学习能力。

此外本文基于多任务学习中的硬共享表示，结合辅助任务学习，提出了对多任务强化学习的优化方案。辅助任务从任务认知和辅助控制两方面出发，提升智能体的性能。一方面，通过提升智能体对整体环境的识别能力，来避免智能体在多个任务场景之间错误的使用经验进行决策的问题，使得智能体可以平衡各个环境之间学到的经验。另一方面，针对强化学习中的图像画面输入，设计辅助控制任务，强化智能体理解环境的能力。通过让智能体预测未来像素变化来强化智能体对于环境的元素的分辨能力。通过各种辅助任务和主任务结合学习，进一步的强化提出的智能体的水平。

最后使用复杂的三维非完全信息视频游戏场景作为实验验证平台，验证多任务强化学习框架中各个组成部分的有效性。通过和其他智能体对比以及将智能体应用在未训练的场景上展示本文提出的方法的优势。

关键词：非完全信息；强化学习；多任务学习；机器博弈

Abstract

With the development of artificial intelligence technology, the practical application of artificial intelligence technology can be seen in more and more scenarios. As a research field that is closest to the real life of human beings, machine gaming has also received extensive attention from researchers. Among them, the incomplete information machine game scenarios have become the focus of research in machine games because of the complexity caused by incomplete information perception and the fact that they are more in line with the actual rules of human life. Deep reinforcement learning, as a powerful method for solving machine games, has been applied in a variety of gaming environments and has led to major breakthroughs in some fields.

This dissertation mainly studies how reinforcement learning agent under incomplete information conditions can achieve excellent performance in multiple task scenarios. To address the problem of poor generalization of traditional single-task agents, a algorithm for parallel multi-task reinforcement learning is proposed, which trains agents in multiple scenarios in parallel to make them perform well in multiple environments. In order to speed up the training and obtain better training performance, this dissertation proposes an overall framework for multi-task reinforcement learning by combining an actor-learner framework and asynchronous actor-critic algorithm. The deconstruction of the data acquisition and training results in a significant increase in training performance. At the same time, the framework addresses the problem of data latency mismatch in off-policy training due to deconstruction, and proposes data correction schemes for learner and actor respectively to correct the correctness of the data learned by the model and improve the learning ability of the intelligences through data correction.

Moreover, inspired by the hard shared representation of multitask learning, we proposes an optimization of multitask reinforcement learning in conjunction with auxiliary task learning. The auxiliary task enhances the performance of the agent from two aspects: task awareness and auxiliary control. On the one hand, by improving the agnet's ability to recognize the whole environments, to avoid the problem of the agent incorrectly using experience for decision making across multiple task scenarios, so that the agent can balance the experience learned across environments. On the other hand, for various important elements in reinforcement learning, design auxiliary control tasks to strengthen

the ability of the agent to understand the environment. The ability of the intelligence to discriminate between elements of the environment is enhanced by having the intelligence predict future pixel changes. Through the combined learning of various auxiliary tasks and main tasks, the level of the proposed agent is further strengthened, and the framework of multi-task reinforcement learning is improved.

This dissertation use a complex 3D video game scenario as an experimental validation platform to verify the effectiveness of each component of the multitasking reinforcement learning framework separately. The advantages of the proposed approach are demonstrated by comparing it to other intelligences and by applying the intelligences to untrained scenarios.

Keywords: incomplete information, reinforcement learning, multitask learning, game theory

目 录

第 1 章 绪论

1.1 课题研究背景及意义

1.1.1 课题研究背景

随着计算机技术的不断发展，人工智能领域已经得到了长足的发展，现在的人工智能技术在我们生活中的诸多场景中都有了实际的应用。例如现在应用广泛的视频图像相关的领域诸如人脸识别，目标检测，以及各大应用中几乎必备的推荐系统等等。在图像，语音，以及一些游戏领域中，人工智能的单体智能表现已经接近人类，表现出了卓越的性能。

从人类自身的角度来说，人类做出每一项决策，其实都可以看作与其他人类或者环境相互博弈的过程，同时趋利避害的本能使得人类在博弈过程中倾向于做出对自己有利的决策。传统的监督或者无监督的机器学习，深度学习的原理，大多是假设存在一个统一的假设空间适用于它的任务场景，当寻找到一个在该场景上表现优秀的假设空间的时候就完成了学习的过程[1]，训练的数据来自已经收集好的样本。现实的很多情况与这种训练过程不同，它们更为复杂，是一种随着时间发展长期决策来获得最高未来收益的过程。

与这种复杂情形最为贴合的就是机器博弈这一领域，它主要着重于研究和开发人工智能系统，模拟人类的各种博弈决策智能行为，并试图达到或者超越人类的顶尖水平。因为现实情况的种种复杂性都可以归于各种博弈过程，所以机器博弈就成为了衡量人工智能能力的一个重要标准[2]。博弈这一概念涵盖了人类社会的各个方面，从宽泛的经济、军事到融入我们生活中的各种小的领域，例如棋牌游戏等。和其他领域的研究方法类似，为了研究一个宽泛的领域，往往会先对其进行抽象，使用包含其特征的小的验证环境来进行算法的探索，之后再大而化之。在研究的过程中，各种类型的游戏由于其具备博弈的特性，成为我们机器博弈的重要研究对象以及实验平台。

博弈问题按照其信息获取的程度，参与游戏的玩家对游戏环境状态信息是否全部知晓可以分为完全信息和非完全信息两种[3]，如果所有的游戏状态都是公开的，每个参与者都可以看到对方的信息，那么这种博弈称为完全信息博弈，典型的例如大多数的棋类游戏。反之，如果我们不能完全的得到所有的信息，有一些私有信息我们无从得知，这种博弈称为非完全信息博弈，

例如广泛流传的各种牌类游戏，大多都有一些私有牌的设置。这种类型的场景更加复杂，往往需要更多的考虑，例如对手建模，各种探索和利用技术。

对于博弈问题目前的解决方法大多分为两类：一类基于博弈论的方法，通过搜索技术，找到游戏的纳什均衡来进行求解；一类基于强化学习的方法，来实现一个可以完成任务的智能体。它通过对从感知到动作的端对端的学习来实现在游戏领域中具备优秀的表现这一目标。通过神经网络以及强化学习的机制来感知游戏信息，例如游戏图像，并最终输出游戏中的动作。这一特性使得人工智能具备了学习各种各样的技能的可能性。本文主要研究后一种解决方案，通过诸多研究者的努力，现在的强化学习和深度学习相结合的方法已经在各种游戏上展示出了其优秀的性能。但如果我们需要一个在多个场景上均有所表现的智能体时，当前的方法就表现出它的局限性，简单的将其推广到多任务时就会遇到问题，例如无法应对样本稀少的场景。

1.1.2 研究目的和意义

一般的深度强化学习的算法理论上都是相对普适的算法，也就是可以应用到各种适合的场景上，但是其训练过程是选择一种场景进行训练，最终获得当前场景下的智能体。对于其他场景，要么重新训练一种智能体，将两种智能体蒸馏在一起，要么在当前智能体的基础上继续训练，也就是终身学习的概念，最终我们获得可以在多个场景下适用的一个智能体。

对于训练多种智能体的方案，其实有很大的训练成本，如果有n个环境，那么我们就要训练n个智能体，也就意味着训练时所需的资源是原来的n倍，同时对于部署来说，在使用我们训练好的智能体的时候，也需要n倍的空间来容纳，针对不同的环境进行切换。对于持续训练的方案来说，其实是将n个场景的训练串行化，在一个智能体训练完成之后继续在它之上进行训练下一个场景，仅考虑训练代价来说，至少需要n倍的时间成本。因此传统的方法在直接将其迁移至多任务场景时容易产生冗余的训练成本。

为了解决这一问题，减少消耗，考虑是否可以同时在一个智能体上训练多个场景，来使得这个智能体可以在多个环境上达到不错的效果。希望在同时训练中，可以使用多个环境的共同点来加强各个环境的学习。并且因为是并行学习，所以我们就可以控制学习的进程，将其看作一个整体，不必考虑串行学习导致的提前终止时后续环境没有训练的问题。通过并行学习，可以消除冗余的训练成本以及应对样本稀少的环境，对获得一个优秀的多任务智能体来说非常具备研究意义。

1.2 国内外研究现状

强化学习是一种决策学习的过程，它的整体学习流程为：通过智能体和环境之间的互动，获得相应的反馈并形成训练数据，来完成训练。环境接受智能体的决策，并计算出下一时刻的状态以及当前动作的收益，使用图像，标量值等表示方法返回给智能体，智能体通过动作的回报来进一步优化决策策略，同时针对新时刻的状态做出决策，重复这个过程直至训练终止，其目标是学习到对当前环境的最优策略。

在Sutton提出时序差分算法[4]，Watkins基于动作状态函数提出Q学习算法[5]之后，强化学习迎来了蓬勃的发展，在这个过程中也取得了不错的成绩，目前已经在各种类型的游戏平台（如雅达利和围棋）和机器人等领域中都表现出了优异的性能。例如在游戏领域，1992年Tesauro利用在时序差分算法之上发展的的TD(λ)算法开发出的智能体在西洋跳棋的领域打败了人类顶尖大师[6]。2013年DeepMind将强化学习与深度学习相结合，实现智能体仅仅依赖纯图像输入，通过自己的学习学会玩雅达利游戏[7]，达到了人类的水平。之后2015年改进版仅仅依靠和人类玩家看到同样的画面，智能体超过了人类的游戏水平[8]。2016年谷歌DeepMind基于深度强化学习算法的围棋机器博弈智能体Alpha Go[9]在与李世石的对弈中以4:1胜出。这些进展都展现出了强化学习在解决博弈问题方面的强劲实力，人们也都期望强化学习可以完成更多更加困难的工作。

常见的强化学习与目前的多数机器学习任务一样，都是针对单个任务的学习过程。但是因为单个任务的表达能力是有限的，因此对于复杂的问题，往往需要将其分解为简单的子问题，使用相关算法解决之后，将最终的结果合并作为复杂问题的结果。这样做看似合理，其实是不正确的，因为现实世界中很多问题并不能作为独立子问题的组合来看待，即使可以将其分解为子问题，它们之间也是存在共享表示将它们联系在一起的。把现实中的复杂问题当做一个个独立的简单的单任务处理，会忽略它们之间的丰富关联信息。针对这些问题，多任务学习提出了解决方案，通过学习的共享表示联合处理多个任务[10,11]。

因为多任务是在共享表示的基础上进行训练的，所以相关算法往往需要在网络结构上进行优化，来更好的组合任务和任务之间的约束。从分享的方式来说，可以将多任务学习分为两种类型，一类是通过共享编码模块的多头网络，该类的多任务学习的整体损失函数由各个任务的损失函数通过权重加权获得，针对如何进行优化的问题，许多研究者做出了一些有效的工作，例如

针对权重需要人工调整，有的研究者通过自动学习来代替权重手动调节[12]，平衡各个不同任务之间的损失梯度，控制学习率[13]或者将多任务的问题当做多目标的优化问题来进行处理[14]等等。另一类方法中每个任务都有自己独立的网络，但是任务之间通过约束条件来表示它们之间的约束关系，例如使用交叉结构[15]、学习各层的特征之间的线性组合[16]、从瘦网络动态调整扩展来获得紧凑的多任务结构[17]等等。

在之前的发展中可以看到融合了深度学习的深度强化学习（Deep Reinforcement Learning，DRL）算法已经可以处理高维输入，在此基础上进行动作输出。但是它的一个显著的问题是在训练时需要海量的样本，对于需要训练的环境来说，收集这些训练的样本需要很大的时间开销。也因此导致样本获取困难的环境很难训练。此时如果对多个环境进行训练，按照一般的DRL的方法，需要对每个环境进行样本收集和训练，使得整体的训练时间和成本很高，在样本缺少的环境就更加困难。为了解决这些问题，研究者们开始研究多任务强化学习问题（Multi-Task Reinforcement Learning，MTRL），意图实现一个智能体完成多个任务。

和求解RL问题类似，在求解MTRL时，一个智能体实际上在求解一系列的马尔可夫决策过程(Markov Decision Processes，MDP)。对于MTRL问题，一些研究者尝试表示之前问题的分布，或者对其分布进行建模，以此作为后续的求解问题的先验条件。Aaron Wilson等人提出使用层次贝叶斯无限混合模型对MDP上的分布进行建模[18]，对于每一个新加入的MDP，使用之前学习过的分布作为一个信息先验。通过这个模型实现在不同但相关的任务中传递经验知识，加速新的问题收敛到最优策略。H. Li等人[19]研究了MTRL在部分可观测随机环境中的问题，引入了区域化策略表示来描述每个任务中智能体的行为，该表示给出了基于历史情况的当前动作的分布模型，在跨多个任务的策略表示之间使用狄利克雷过程，来作为其非参数化先验，以集群化任务，提出的算法在实验中优于单任务强化学习。

但是上面的方法，在样本稀少的情况下很难取得效果，少量的样本不足以训练出好的先验，因此有一些研究者开始考虑一些样本稀少的场景。在这些问题中，学习者被提供了一组任务，但是对于任何给定的策略，只能生成少量的样本。由于样本的数量可能不足以学习对政策的准确评估，因此研究者们提出利用任务间的相似结构进行共同学习，来加强训练的效果。A.Lazaric和M.Ghavamzadeh[20]考虑不同任务在它们的值函数中共享结构的情况，并通过假设值函数都是从一个公共先验中采样的来建模。每个任务都使用高斯过

程时间差值函数模型，采用分层贝叶斯方法对不同任务的值函数分布进行建模。Calandriello 等[21] 研究可以使用原始特征集的相同子集在近似线性空间中表示的场景，假设不同任务中值函数的参数向量是联合稀疏的，然后将 MTFS 方法与 $L_{2,1}$ 正则化以及 MTFL 方法进行扩展，共同学习多个任务中的值函数。

另外一种比较常见的思路是通过模仿来进行学习，核心是指导者或者专家，指导当前的智能体进行学习，其中指导者和专家是预先训练好的对应环境的智能体。例如 E. Parisotto 等人[22]，提出的 Actor-Mimic 方法结合了深度强化学习和模型压缩技术来训练能够学习执行多个任务的策略网络。在多位专家的指导下，训练一个单一的策略网络，学习怎么在一组不同的任务中进行行动，同时深层策略网络学习的表示可以在没有预先专家知道的情况下推广到新的环境，加快在新环境下的学习。J. Andreas 等人[23] 提出了一个基于策略草图的多任务强化学习的框架。提出了一个将每个子任务与一个模块子策略关联起来的模型，以便从策略草图中学习，策略草图用命名子任务序列来注释任务，并提供关于任务之间高级结构关系的信息。使用行动者-评论家算法进行优化训练，在整个特定于任务的策略上共同最大化奖励。

对于更加复杂的非完全信息环境，处理起来更为困难，S. Omidshafiei 等人[24] 对部分可观测环境下的多任务多智能体学习问题进行了研究，引入了一种分散的单任务学习方法，这种方法对团队成员的并发交互非常健壮，之后通过将分散的单任务策略提炼为跨多个任务的统一策略，解决了局部可观测条件下的多任务多智能体强化学习问题。A. M. Saxe 等人[25] 提出层次结构对强化学习方法的可扩展性至关重要。大多数当前的层次结构框架都是串行执行操作的。他们提出了一种基于并行执行许多操作的新方法来替代这些控制层次结构，方案利用线性可解马尔可夫决策过程 (LMDP) 框架保证并发性，这使得智能体能够同时利用多个宏操作来解决新任务。他们提出了一种多任务线性可解 MDP 来维护任务的并行分布式表示，其中每个任务都允许智能体同时绘制宏操作。

有一些研究者尝试使用辅助的额外的任务来加强智能体的性能，使得智能体可以从监督学习收益，Mirowski 等人[26] 通过构建给定 RGB 输入的深度地图，让基于 A3C 的智能体接受了额外的训练，这有助于它学习如何在 3D 环境中导航。Mirowski 等人研究中表明，预测深度比接收深度作为额外输入更有用，这进一步支持了辅助任务引起的梯度可以非常有效地提高 DRL 的观点。为了做到多任务的强化学习，可以采用共享神经网络进行多任务学习参数，通过跨相关任务的传输来提高数据效率。然而，在实际的应用中，会

因为梯度来自不同的任务会产生消极的干扰，使学习变得不稳定，导致有效数据减少。Y. W. The 等人[27]提出了一种多任务联合训练的新方法，在各个任务间分享一种“提炼”的政策，捕捉在不同的任务间共同的行为，而不是在并行的采集者间共享参数。每个并行的智能体都被训练去解决自己的任务，同时通过正则项限制来接近共享策略，此时共享策略就被蒸馏为所有策略的核心策略。

更确切的说，首先可以训练一个强的“教师”模型，然后用它来指导一个弱的“学生”模型的训练，这种想法很类似上面的指导学习的思想，和这种想法类似的技术例如监督学习中的被称为蒸馏[28]的神经网络知识转移技术已被用于将大型 DQN 学习到的转移策略转换为较小的 DQN，以及将多个单独游戏训练的 DQN 学习到的转移策略转换为单个 DQN[29]。在此基础上继续发展了各种蒸馏技术[30,31]，也被应用在各种算法中来提升整体性能[32,33]。

1.3 本文的主要研究内容

本文研究非完全信息博弈环境中的并行多任务强化学习算法，主要研究内容如下：

（1）针对单任务的强化学习算法的泛化性能不好的问题，提出基于行动者-学习者的并行多任务强化学习算法。单任务的强化学习算法在推广到多任务场景上时会出现冗余的训练成本，训练速度慢，在样本少的环境中训练效果差。本文使用行动者-学习者模型来加快数据的采集，提升训练的性能。对于因为数据延迟导致的结果不准确的问题，基于重要性采样提出对经验的校正方案，同时使用广义优势估计来平衡行动者的误差和方差。

（2）针对多任务中智能体对环境的认知能力不足的问题，提出了基于辅助任务的并行强化学习强化方案。提出任务认知辅助任务，帮助智能体识别经验来自哪个环境。提出控制辅助任务，帮助智能体更好的识别环境，预测环境的变化情况。通过硬共享的多任务学习方式，使用两种辅助任务加强主任务的学习，提升智能体的性能。

（3）在三维的非完全信息场景上进行实验设计和验证。设计实验验证本文多任务的有效性、辅助任务设计的有效性以及智能体的鲁棒性。通过消融实验验证本文方法各个部分的有效性，和当前的一些方法进行对比验证智能体整体性能。

1.4 本文的组织结构

本文主要分为5章，每一章的具体内容如下:

第1章绪论。首先介绍本课题的研究背景以及意义，并且说明本文研究内容相关的研究领域当前的国内外研究现状，并在最后对本文的研究内容进行阐述，对本文的组织结构进行总结。

第2章相关理论。介绍本课题用到的一些深度强化学习和多任务强化学习的相关知识以及相关算法，对研究的内容进行形式化的定义和描述。同时有一些现有的方法进行介绍，为本文的工作奠定基础。

第3章提出结合行动者-学习者的多任务强化学习算法。针对提出的问题给出对应的解决方案，分为行动者-学习者框架和对应的校正方案两部分来进行描述，分别解决速度慢以及离线训练出现偏差的问题。最终提出了整体的解决方案。

第4章提出对上文算法的优化方案。在第3章的基础上继续进行方案的优化，结合辅助任务学习的想法，设计任务认知和辅助控制两项辅助任务，帮助智能体提升环境的认识能力和对任务的认识能力。最终给出整体的网络结构以及优化问题。

第5章实验验证与分析。使用三维视频游戏作为验证场景，设计实验验证本文提出的方法的有效性，证明本文提出的方法和框架可以有效的提升智能体在多任务场景中的性能表现。

第 2 章 多任务强化学习相关理论

2.1 深度强化学习及其相关算法

2.1.1 基础定义

典型的强化学习场景如图 2-1所示，可以分为两个组件：环境中的学习者或者决策者被称为智能体（Agent）。智能体交互的对象，被称为环境（Environment）。智能体和环境的交互在整个强化学习过程中一直不间断，智能体接受环境的反馈进行决策，同时环境接受动作向智能体呈现新的场景。在交互过程中环境会产生一个奖励数值，智能体通过最大化奖励的数值来不断的提升自己的策略。

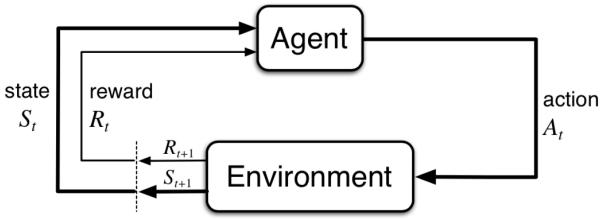


图 2-1 强化学习过程[34]

更具体的来说，一个马尔科夫决策过程定义为一个 5 元组 $(\mathcal{S},\mathcal{A},\mathcal{T},\mathcal{R},\gamma)$，其中:

• $\mathcal{S}$ 代表环境中一系列的状态，任意的 $s \in \mathcal{S}$ 表示其中一个合法的状态，使用增加下标的 S_t 表示时刻 t 的状态。

• $\mathcal{A}$ 代表当前环境下一系列的合法动作，同样的，任意 $a \in \mathcal{A}$ 表示其中一个动作。使用 A_t 表示在时刻 t 做出的动作。

• $\mathcal{T}$ 代表环境的转换函数，它表示了环境的状态转换规律，它是环境内部信息，在强化学习过程中并不暴露给智能体。其中 $T(s'|s,a)$ 表示从状态 s 选择动作 a 到达下一个状态 s' 的概率。

• $\mathcal{R}$ 是一个奖励函数，将 S 中的状态映射到 R 中的奖励。

• γ 是一个折扣因子，用来控制视野的远近，表征了对于较远时间步的权重。γ 为 1 表示所有的状态都具备相同的重要性，为 0 的时候表示不考虑未来的结果，仅关注当前的状态。

在五元组的描述下，可以定义在给定了前一个状态 s 和做出的特定的动

作值 a 的情况下，任意的 $s' \in \mathcal{S}$，$r \in \mathcal{R}$ 在时间 t 发生的概率为：

$$p(s', r|s, a) \doteq Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\} \tag{2-1}$$

式 2-1 也可以被称为整个环境的模型（Model），对整个环境下的状态转移以及奖励生成进行建模。在和环境交互中，智能体决策时的动作分布将其称为一个策略：π，它是从状态到动作的一个映射。一般的有两种策略：

- 确定性策略：$a = \pi(s)$，给定一个状态每次的决策都是相同的
- 随机性策略：$\pi(a|s) = \mathbb{P}(A_t = a|S_t = s)$, 决策是一个概率分布，根据此分布抽样出动作。

之后开始定义强化学习的优化目标，根据奖励假设：所有的目标都可以通过期望累计奖励的最大化来描述。也就是说可以通过最大化累计奖励的期望来实现我们想要的目标，一般的，累计的奖励称之为回报函数，可以定义为所有奖励的累计，也即定义回报 G_t 为：

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \cdots = \sum_{k=0}^{\infty} R_{t+k+1} \tag{2-2}$$

但是将未来的奖励看作和当前的一样的重要程度是不恰当的，对于无穷长的无限马尔可夫过程来讲，该式无法计算，因此可以对未来的回报进行一定的加权，可以定义时间 t 时候的带折扣的回报函数：

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2-3}$$

这里的 γ 也就是上文中提到的用来控制视野的折扣因子，我们希望回报不仅仅代表本次的收益，更应该考虑将来的收益，但是对于不确定的将来，不能占据过大的权重。因此强化学习的目标，也就是值函数可以描述为：找到一个最佳的策略 π，它可以最大化对于未来奖励的期望。给定一个状态 s，对应的值函数定义为：

$$V_\pi(s) \doteq \mathbb{E}_\pi [G_t|S_t = s] \tag{2-4a}$$

$$= \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s \right], for s \in \mathcal{S} \tag{2-4b}$$

特别的，如果给定了一个状态和动作对 (s, a)，对应的值函数的定义为：

$$Q_\pi(s, a) \doteq \mathbb{E}_\pi [G_t|S_t = s, A_t = a] \tag{2-5a}$$

$$= \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a \right] \tag{2-5b}$$

在此基础上可以得到常用的优势函数如下:

$$A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s) \tag{2-6}$$

根据强化学习中用到的技术，对强化学习可以进行多种类型的分类。从对 $T(s'|s,a)$ 的了解程度来说，可以分为基于模型（model-based）的和无模型（model-free）的两类方法。对于 model-based 的方法，智能体和环境交互执行动作 a 从一个状态 s 转移到另一个状态 s'，收到了奖励 r。那么智能体会从中学习到 $T(s'|s,a)$ 以及 $R(s,a)$，也就是式（2-1）所表示的概率。在学习了环境的模型之后，在进行决策的时候，使用一些规划算法就可以得到最终所需的最优动作。

此外还可以不通过对环境建模来进行学习，直接对未来的一系列回报进行预估，也就是本文要进行探讨的 model-free 的一系列方法。根据其优化的目标，可以将其分为基于值函数的方法（value-based）以及基于策略的方法（policy-based），value-based 的方法是对式（2-4a）或者（2-5a）进行学习，而 policy-based 的方法是对 π 进行学习。在深度强化学习中，如果网络的参数为 θ，那么拟合估计的对象就是 v_θ，q_θ 或者 π_θ。接下来对这两种主要方法以及其延伸算法进行介绍。

2.1.2 基于策略梯度的强化学习

基于策略梯度的方法直接对策略进行优化，来获得最优策略，代表性算法是策略梯度（Policy Gradient，PG）算法。根据前文所述，强化学习的目标可以认为是累计的奖励之和的期望，在进行决策的时候，给定一个策略之后，通过策略的不断决策，可以获得一个决策序列

$$\tau = \{s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_T, a_T, r_T\} \tag{2-7}$$

这里使用参数 θ 拟合的策略 π_θ，那么根据式（2-4a），强化学习的目标函数可以表示为

$$J(\theta) = V_{\pi_\theta}(s_0) = \mathbb{E}_{\tau\sim\pi_\theta}[G(t)] = \int \pi_\theta(\tau)G(\tau)\mathrm{d}\tau \tag{2-8}$$

在这个目标函数下，通过深度学习方法，我们需要的是求得一个最优的参数使得回报的期望最大:

$$\theta^\star = argmax_\theta \mathbb{E}_{\tau\sim\pi_\theta(\tau)}[G(t)] \tag{2-9}$$

对式（2-8）进行求导可以得到

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\nabla_\theta log\pi_\theta(\tau)G(\tau)\right] \tag{2-10}$$

之后使用式（2-10）进行后续的更新即可迭代的得到式（2-9）中表示的最优参数。各种基于梯度的强化学习方法基本上是在上面的框架下进行改进的，主要的区别在于式中 $G(\tau)$ 的形式不同。REINFORCE 算法[35] 中中 G 的取值为式（2-3），之后的优化有增加 baseline，使用优势函数等方法。TRPO[36] 中使用 $G = \pi_\theta(a|s)A/\pi_{\theta_{old}}(a|s)$ 来弥补 PG 算法中步长很难控制的问题，随后的 PPO[37] 算法在此基础上增加了剪切，优化了计算过程。

2.1.3 基于值函数的强化学习

对于上面提到的两种值函数 Q 和 V，他们有如下的关系：

$$V(s) = \sum_a \pi(a|s)Q(s,a) \tag{2-11}$$

$$Q(s,a) = \sum_{s'} p(s'|s,a)(R(s,a) + V(s')) \tag{2-12}$$

式（2-11）表示对于一个状态 s，可以得到在这个状态上采用动作 a 的值 $Q(s,a)$，对所有当前状态下的合法动作进行加权，就可以得到当前状态的值 $V(s)$。式（2-11）表示在一个状态 s 上采用动作 a 的值相当于后续所有可能的下一个状态 s' 的值的加权和。为了得到对应的值 $V(s)$，常用的方法有动态规划（Dynamic Programming，DP），蒙特卡洛抽样（Monte-Carlo Sampling，MC）以及时序差分（Temporal-Difference，TD）。根据上面的关系，我们可以得到：

$$V_{k+1}(s) = \sum_a \pi(a|s)Q_k(s,a) = \sum_a \pi(a|s)\sum_{s'} p(s'|s,a)(R(s,a) + V_k(s')) \tag{2-13}$$

其中也可以加上折扣因子表示未来的加权，该式也被称为贝尔曼方程，这样就可以使用后续的状态来更新当前值，获得当前的 V。但是大多数情况都无法获得 $p(s'|s,a)$，也就是 model-free 的情况下，我们想要得到对于一个状态 s_t 对应的 $V(s_t)$ 的值，可以使用另外两种方式进行更新。

MC 方法使用模拟的方式，从当前的状态出发，采样获得一个执行的历史记录，直到最终状态，获得对应的收益 G_t，之后使用

$$V(s_t) \leftarrow V(s_t) + \alpha(G_t - V(s_t)) \tag{2-14}$$

来更新当前的状态值，目标是让 $V(s_t)$ 尽可能的和 G_t 接近。但是这种方式估计的方差较大，需要将轨迹模拟到终止状态。另一种方法 TD 在此基础上，仅仅向下更新一步，使用：

$$V(s_t) \leftarrow V(s_t) + \alpha((r_{t+1} + \gamma V(s_{t+1}) - V(s_t)) \tag{2-15}$$

来更新，目标是使得当前的$V(s_t)$和$r_{t+1} + \gamma V(s_{t+1})$尽可能接近，后者也被称为$TD-target$。这些更新方法将状态值函数替换为动作状态值函数，即可得到对应的更新规则。

为了减少在遇到很大的状态和动作空间时，值函数存储的空间，后续引入了深度网络，使用值函数近似来使得：

$$Q_\theta(s,a) \approx Q_\pi(s,a) \tag{2-16}$$

对于 model-free 的情况，我们通过上面的 MC 或者 TD 方法来对$Q_\pi(s,a)$进行计算，之后让网络进行拟合。也即设置目标函数为：

$$J(\theta) = \mathbb{E}\left[(Q_\pi(s,a) - Q_\theta(s,a))^2\right] \tag{2-17}$$

之后的改进主要集中在对网络的改变，或者训练方式的改进。融入其他的网络形式，例如 LSTM[38]，递归网络[39]，分布式的改进等。训练方式方面增加经验回放[40,41]，增加注意力机制，使用双 DQN[42]，对称 DQN 等方面的变化，但是基本的流程基本没有发生大的改变。

2.1.4 行动者-评论者结构

行动者-评论者结合了前面提到的两种方法，使用行动者来优化策略，使用评论者来优化值函数。也就是行动者对策略进行更新，使用参数θ来进行拟合，使用上面的梯度进行更新，同时评论者使用参数w来拟合 Q 函数，使用 TD 更新w。因为其吸收了两种方式的优点，因此现在很多广泛使用的算法都使用了 AC 的结构。

比较典型的行动者-评论者的算法是异步优势行动评论算法（Asynchronous Advantage Actor Critic，A3C[43]）。如算法 2-1 所示，它使用多个行动者线程来采集数据，每个行动者中更新自己本地的梯度，随后将各个线程中累计的梯度更新量，更新到公共的网络中。

2.2 多任务场景下的强化学习

多任务下的强化学习场景和单任务的有所不同，主要表现在一个单独的智能体需要同时学习应对N个环境。首先我们形式化的描述任务场景，我们定义$D^T = \{D_i = (\mathcal{S}_i, \mathcal{A}_i, \mathcal{T}_i, \mathcal{R}_i, \gamma_i)\}_{i=1}^N$，其中的元素定义和单任务的定义相同，下标来表征不同环境索引。在多任务场景下，每个环境都有他们自己独特的

算法 2-1 A3C 算法

Input: 公共的网络参数 θ，w。本线程的参数 θ'，w'，抽样长度 T_{local}，当前线程执行的时间步 t，全局时间步 T，迭代最大时间步 T_{max}

Output: 公共的网络参数 θ，w

初始化 $t \leftarrow 1$
repeat
  $t_{start} = t$
  重置 Actor 和 Critic 的梯度更新量：$d\theta \leftarrow 0$，$dw \leftarrow 0$
  从公共的网络同步参数：$\theta' = \theta$，$w' = w$
  repeat
    抽样并执行：$a_t \sim \pi_{\theta'}(a_t|s_t)$ 并获得 r_t
    $t \leftarrow t + 1$
    $T \leftarrow T + 1$
  until $t == T_{local}$ *or* s_t *is terminal*;
  计算 $R = \begin{cases} 0, & terminal \\ V_{w'}(s_t), & nonterminal \end{cases}$
  foreach $i \in \{t-1, \ldots, t_{start}\}$ **do**
    $R \leftarrow r_i + \gamma R$
    $d\theta \leftarrow d\theta + \nabla_{\theta'} log\pi_{\theta'}(a_i|s_i)(R - V_{w'}(s_i))$
    $dw \leftarrow dw + \partial(R - V_{w'}(s_i))/\partial w'$
  end
  执行异步更新，使用 $d\theta$ 和 dw 更新公共的网络参数 θ 和 w
until $T > T_{max}$;

变化规律，回报函数这些属性。即使不同的环境之间遵循相同的物理规律，可能存在动作空间部分重合，但是他们的回报或者状态转移是不同的。这就给智能体同时学习不同的环境带来了困难。智能体可能将一个环境中的经验错误的用在其他环境中，造成其他环境中的收益的下降。

为了便于描述，可以将这种场景归为一种更大的 MDP，它的状态空间为 $S = \{\{s_j, i\}_{s_j \in S_i}\}_{i=1}^{T}$，其中 i 为环境的索引，它可能是隐式的或者显式的暴露给智能体的策略，但往往只用于训练，在测试中智能体是不知道此时处在什么环境中的。对于每一个任务 i 我们定义它的转移函数为 $p_i(s'|s,a)$，回报函数为 $R_i(a,s)$。后续将在此基础探讨多任务学习的相关方法。

对于多任务场景下的强化学习方法，可以分为三类：(1) 基于策略蒸馏

的方法，主要思想是将多个任务的智能体提取为一个中心智能体，将多个智能体的经验蒸馏到一个智能体上。(2) 基于附加任务的方法，主要利用了深度学习中监督学习等便于实现的特性，对获取的经验进行了额外的学习，来加强中心智能体的性能。(3) 基于并行学习的方法，利用了最直观的想法，直接的对各个任务进行并行训练。多个环境同时学习，来达到在多个环境上都有表现的目的。

基于策略蒸馏的方法如图 2-2 所示，一般会训练一个核心的策略，之后使用这个核心策略来帮助训练其他环境下的策略，同时别的环境上的策略也在往核心智能体上进行经验的转移。首先定义 π_0 为我们需求的从多个任务中提取出来的包含了任务间公共信息的策略，之后为了约束其他任务的策略，我们使用 γ-折扣的 KL 散度对其他的任意策略 π_i 进行正则化:

$$\mathbb{E}_{\pi_i}\left[\sum_{t\geq 0}\gamma^t log\frac{\pi_i(a_t s_t)}{\pi_0(a_t s_t)}\right] \tag{2-18}$$

为了使得智能体可以更好的进行探索，也会使用 γ-折扣的熵正则来对探索进行鼓励。在使用这些技术的情况下，我们可以定义如下的目标函数:

$$J\left(\pi_0,\{\pi_i\}_{i=1}^n\right)=\sum_i\left\{\mathbb{E}_{\pi_i}\left[\sum_{t\geq 0}\gamma^t R_i(a_t,s_t)+\frac{\gamma^t\alpha}{\beta}log\pi_0(a_t|s_t)-\frac{\gamma^t}{\beta}log\pi_i(a_t|s_t)\right]\right\} \tag{2-19}$$

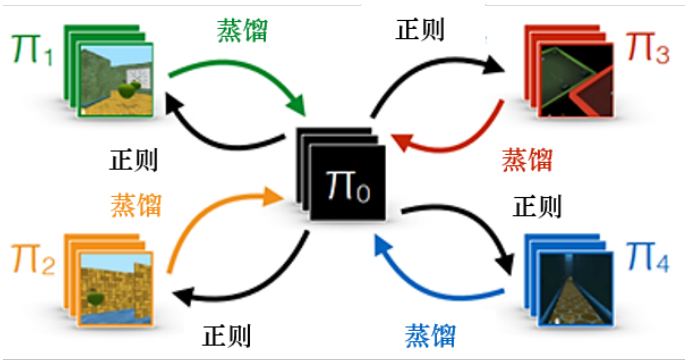


图 2-2 策略蒸馏示意图

之后就 π_0 和每个任务的 π_i 进行训练。当 π_0 固定的时候，共享策略指导其他任务的策略训练。目标函数变为对每一个单独的任务的极大化问题，此时每个任务的期望回报被正则化为 $R_i'(a,s)=R_i(a,s)+\frac{\alpha}{\beta}log\pi_0(a|s)$。该问题可以使用单任务的强化学习中的一些技巧例如 Soft Q-learning 进行优化或者求解，求解出最优策略。

反之当我们进行策略提取时，从诸多任务的策略 π_i 中学习出一个核心策略来对未来进行指导，此时考虑式 (2-19) 中仅包含有 π_0 的项:

$$\frac{\alpha}{\beta}\sum_{i}\mathbb{E}_{\pi_i}\left[\sum_{t\geq 0}\gamma^t log\pi_0(a_t|s_t)\right] \tag{2-20}$$

可以发现此时就是一个对数似然，将 π_0 在每个任务的策略下，拟合到一个混合的状态动作值概率分布。

基于蒸馏的多任务强化学习方法仍然没有克服需要多个智能体来进行训练的问题，整个算法的核心在于训练教师-学生网络，以一个核心的教师网络来表达多个环境之中的共同点，之后来指导或者正则化约束学生网络的学习。所以在获得真正的智能体之前还需要额外的训练一系列的教师网络，相当于是完成了每个任务的单任务训练之后，再进行多任务的训练。

这类算法一般有多到一和一到多两种形式，多到一也就是多个环境的网络训练完毕之后使用蒸馏技术将其汇总到一个网络上来，这就是最终的智能体，常见于各种需要压缩的场景上。这种方法其实还是要训练好多个环境下的多个智能体，和传统的强化学习的算法差距不大。一到多就是上文描述的场景，一个核心的网络来正则化的指导多个场景下的多个子网络，这样获得在各个环境下的优秀的智能体，这样没有起到我们想要的一个智能体多个环境的效果。因此本课题主要使用后面两种技术，使用并行训练来获得智能体，使用辅助任务来强化智能体。

2.3 本章小结

本章介绍了强化学习的一些基础定义以及相关的算法，从强化学习问题的一般形式出发，介绍了在深度强化学习中用到的各种定义，以及值函数之间的相互关系。在这些定义的基础上介绍了两大类常用的强化学习算法的相关知识，总结他们的基本形式以及在其之上的一些改进。随后介绍了结合了两类算法的行动者-评论者算法，也是本课题要采用的算法。定义了本课题要研究的多任务强化学习问题的形式化描述。在此基础上介绍了其上的各种算法，引出本课题要使用的技术，为后面的具体工作打下基础。

第 3 章 基于行动者-学习者的多任务强化学习算法

3.1 引言

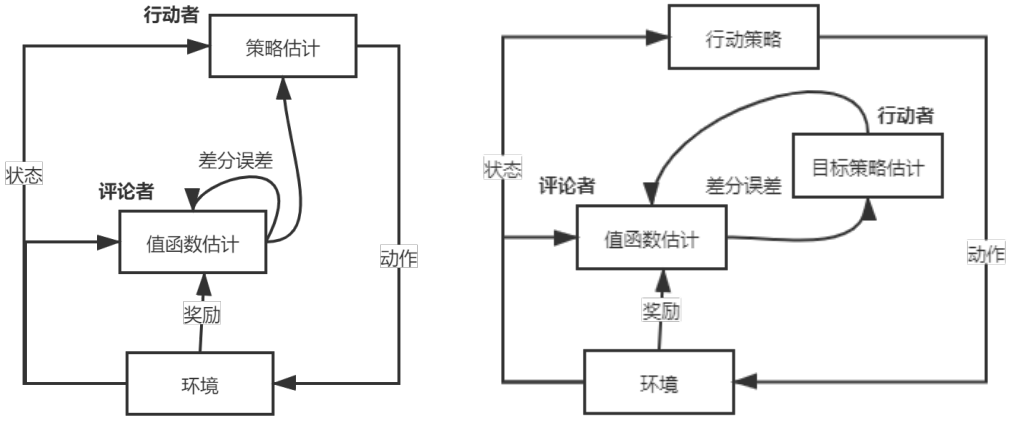
在强化学习中，限制智能体的性能的一个很重要的因素就是智能体学习经验的多寡，因此可以经常看到许多训练结果很优秀的算法需要几千万甚至几十亿的经验来进行训练。从这个角度来讲，采集数据的速度十分重要，它决定我们的智能体在同样的时间内可以训练的有多快。针对这个问题，本文使用了行动者-学习者（Actor-Learner）结构来提升数据的采集效率。同时为了应对框架中数据不及时，延时方差较大的问题，对框架中的各个部分使用了额外的校正技术来获得更好的结果。在此基础上提出了在本课题中使用的多任务场景下的强化学习算法。

3.2 行动者-学习者结构

和传统的单任务的训练一样，多任务的训练也需要庞大的数据量，直观上来看，想要在多个环境中都产生较好的效果，那么所需的数据量要比单任务要大，所以这是首先要解决的问题。一般的来说，强化学习中训练的方式可以分为两种：在线策略训练（on-policy）和离线策略训练（off-policy），两者的区别在于采集经验时使用的策略 μ 和学习时使用的策略 π 是不是相同。如果二者相同就是在线策略的训练方式，也就是采集数据的策略和学习的策略相同，用采集到的数据进行实时的训练，因此采集数据和智能体训练是一个序列式的过程。

以行动者-评论者（Actor-Critic）的结构来说明，如图 3-1 所示，对于在线策略的训练方式，每次更新都是将差分误差值更新到行动者和评论者上，同时整个训练循环的推进是由行动者来进行的，所以是采集-训练-采集这样的一个循环过程。但是对于离线策略来说，采集的循环和训练的循环相对独立，使用采集策略来和环境进行交互，之后完成数据的采集。评论者则只是用于训练循环。训练和采集的循环会在进行定期的更新，也就是完成采集策略和目标策略二者的同步过程。

为了可以更快的采集数据，加速训练，很多新的算法被开发出来，广泛的使用离线策略的方式进行训练，例如 A2C，A3C，但是这些方法需要在一个批次的完整的经验采集完毕的时候更新网络，或者各个线程之间进行梯度



a) 在线策略　　b) 离线策略

图 3-1 行动者-评论者模型

的同步累计，也就是和图 3-1 b)中的模式一样。这种模式整个的更新其实还是存在一定的瓶颈，在多任务场景中，由于各个环境之间的不同，获得一条完整的对战历史记录所需时间也不同，同时由于多线程或者分布式的启动数据的采集，所以每个环境的起始时间也不确定，这样就很容易造成虽然是做了加速处理，但是各个环境还是按照线性的顺序启动，导致数据没有采集到位，没有取得加速的效果。因此不同的环境很难同时获得相同多的数据量，那么就会有大量的时间在等待耗时最长的智能体完成数据的采集工作。所以现在的结构中数据采集和智能体训练的耦合程度还是太高，因此为了改善这种情况，需要对数据采集结构进行设计。

本文后续的工作基于行动者-学习者（Actor-Learner）的基本结构，这一结构本质上是对行动者-评论者的拓展，目的是将数据采集过程和学习优化的过程解耦，数据的采集由行动者来完成，可以根据资源情况来动态的调整部署的数量。对于行动者-评论者的优化在于将数据采集和将原本的传递梯度改为了单纯的传递经验值，因此免去了梯度更新的时效性这一限制因素。也就是意味着现在的训练过程已经没有了额外的限制。图 3-2 展示了基本的结构。每一个行动者内部维护了一个环境实例以及与学习者同步的神经网络，来负责采集数据，行动者内部其实就是一个迭代循环，不停的进行 MDP 的采集过程。采集到的数据被发送到一个经验池中，这里图中使用一个简单的队列来表示。学习者负责取出经验，然后进行迭代学习，这样的设计中，从学习者的角度来说，就是一个正常的深度学习的迭代的过程，因此速度是很快的。从行动者的角度来看，每个行动者都是独立的，一直在运行直到训练

结束，每个行动者不需要等待其他的行动者执行完毕，同样的如果发生了错误，整体的结构还是可以继续进行的。最后在学习的算法上，采用了A3C算法(算法2-1)来训练智能体，结合基于值的方法和基于梯度的方法的优势。

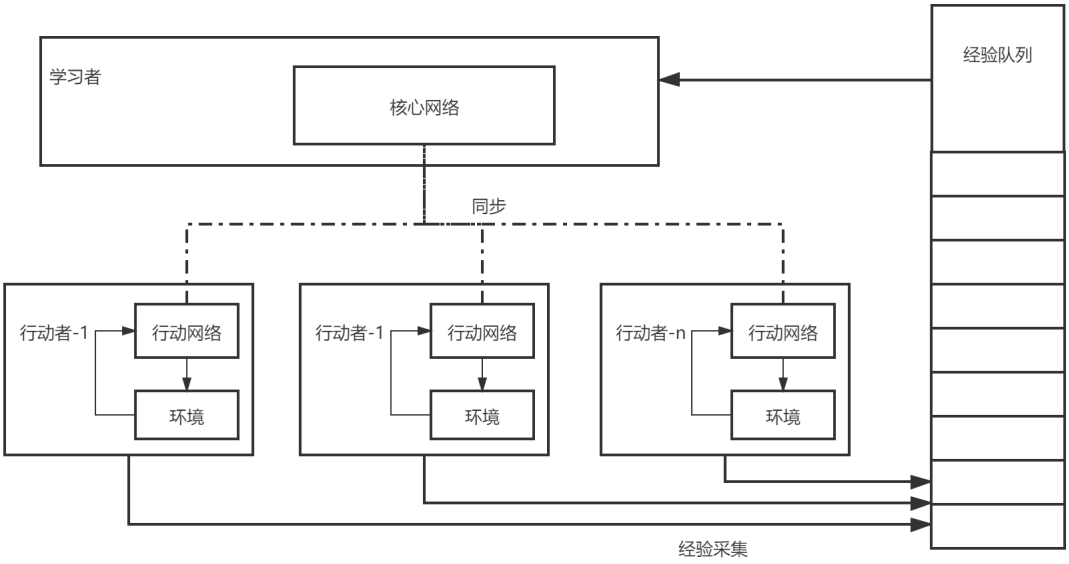


图 3-2 行动者-学习者结构

下面分别介绍行动者和学习者中使用的方法，以及校正方法，最后将其合并给出最终的更新策略。

3.3 基于值函数的学习者

行动者-学习者模型中的学习者和AC算法中的评论者一样，使用基于值的方法来进行学习。本文使用网络来拟合值函数，而不是动作价值函数，首先将上面的定义推广到值函数上。学习的基本流程是首先根据采集的经验计算出当前的策略 π 对应的 V_π，使用均方误差来使得网络进一步的拟合当前的值函数。也即对于参数为 w 的值函数网络，目标函数为:

$$J(w) = (V_\pi - V_w)^2 \tag{3-1}$$

首先计算出 V_π，之后为了减少误差，再对其进行修正。

3.3.1 目标值函数的计算

在上面的公式中，可以发现，剩余主要的工作是给出 V_π 的计算方法，之后就可以使用深度学习的方法进行优化，最小化损失函数来获得结果。在计算的时候一般使用TD方法来计算 V_π，和使用式（2-15）使用了单步的TD不

同，这里考虑更好的方式。单步的TD更新使用了下一步的一个准确值，之后的部分直接使用估计值来进行表示，这种方式存在着极大的偏差，因为每次的估计值并不都是准确的，在仅使用一个状态的准确值之后使用后续状态的估计值会产生很大的不准确性。而使用式（2-14）的更新方式的话，估计值是不断接近G_t的，相当于每次都是拿着真实的收益来进行更新的，因此这种方法不存在偏差，但是因为每次都是对一条经验进行的更新，而从一个状态往后的路径序列是很多的，所以这种方式很可能每次都是不同的路径，因此所表示的最终值方差很大。

因此一般都会将视野放的更加的远，向后看n步，使用后面n步的回报来进行加权，n步之后的部分使用值函数进行估计，这样同时考虑到当前的局面以及未来的收益情况，对于一个序列$(s_t, a_t, r_{t+1}, s_{t+1}, \dots)$，在状态$s_t$处的价值函数的带折扣因子$\gamma$的$n$步的TD更新方式：

$$G_t^{(n)} = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{n-2} r_{t+n-1} + \gamma^{n-1} V(s_{t+n}) \tag{3-2}$$

$$V^{(n)}(s_t) \leftarrow V(s_t) + \alpha(G_t^{(n)} - V(s_t)) \tag{3-3}$$

计算出n步的返回值，之后使用其进行TD的更新，这其实是使用单步的TD和MC的结合，来降低方差和偏差。综上所述，本文使用式（3-3）的方式来进行值函数的计算。

3.3.2 重要性采样

在行动者-学习者框架中，我们所获得的数据采集的经验经过汇总之后用于学习的时候，采集经验所用的策略和实际学习时的策略其实已经不一致了，对于行动策略μ以及学习策略π，在实际采集经验τ时有$\tau \sim \mu$，但是在学习时在计算从$\mathbb{E}[r(\tau)]$时却假设的是$\tau \sim \pi$，这样就产生了实际计算的不一致，因此需要纠正这种误差。

首先我们考虑一个场景，现在需要计算一个函数$f(x)$的期望$E = \mathbb{E}[f(x)]$，其中$x \sim p(x)$，是从其他分布中获得的。有如下的对它的估计

$$\mathbb{E}_{x\sim p(x)}[f(x)] = \int f(x)p(x)dx \tag{3-4}$$

$$= \int \frac{q(x)}{q(x)} f(x)p(x)dx = \int q(x)\frac{p(x)}{q(x)} f(x)dx$$

$$= \mathbb{E}_{x\sim q(x)}\left[\frac{p(x)}{q(x)} f(x)\right] \tag{3-5}$$

可以发现，通过引入一个新的变量，使得对$x \sim q(x)$的期望可以表征$x \sim p(x)$

的期望，该方法被称为重要性抽样，其中分布的比值 $p(x)/q(x)$ 称为重要性权重，通过重要性权重对采集到的数据 $f(x)$ 进行校正。

将此方法应用到当前场景，解决上述问题。重要性抽样的方法允许在采集策略和学习策略二者相差不大的情况下使用采集策略采集的数据来计算学习策略的期望值，从而完成学习。根据式（3-5），定义当前场景下的重要性权重为：

$$IS = \frac{\pi(\tau)}{\mu(\tau)} = \frac{p(s_1)\prod_{t=1}^{T}(\pi(a_t|s_t)p(s_{t+1}|s_t,a_t))}{p(s_1)\prod_{t=1}^{T}(\mu(a_t|s_t)p(s_{t+1}|s_t,a_t))} \tag{3-6}$$

$$= \prod_{t=1}^{T}\frac{\pi(a_t|s_t)}{\mu(a_t|s_t)} \tag{3-7}$$

3.3.3 目标值函数的修正

从上节的描述可以知道因为框架整体是一个离线策略的学习过程，因此直接使用 3.3.1一节中的计算方法来计算值函数是有偏差的，下面使用 3.3.2 的方法对其进行修正。同样的问题也存在在基于动作价值函数的方法中，首先介绍基于动作价值函数的方法中的解决方案 Retrace(λ)[44]，之后将其推广至本文使用的价值函数中。

为了得到无偏的动作价值函数的估计，可以想到可以使用上面的重要性抽样来进行处理，对于基于返回值的 off-policy 策略的算法，定义 t 时刻的差分误差为：

$$\delta_t = r_t + \gamma\mathbb{E}_\pi Q(s_{t+1},a) - Q(s_t,a_t) \tag{3-8}$$

在此基础上我们可以定义通用的基于返回值的 off-policy 的动作状态函数算子为：

$$\mathcal{R}Q(s_t,a_t) = Q(s_t,a_t) + \mathbb{E}_\mu\left[\sum_{t\geq 0}\gamma^t\left(\prod_{s=1}^{t}c_s\right)\delta_t\right] \tag{3-9}$$

其中 $\mathbb{E}_\pi Q(s_t,\cdot) = \sum_a \pi(a|s)Q(s,a)$，同时对于 $t=0$ 时，定义 $\prod_{s=1}^{t}c_s = 1$。在这个式子中 c_s 是一个权重，可以说是对于资格迹[34]（eligibility trace）的扩展。

资格迹是引入的一个和每个状态相关的向量，每个状态都有一个初始的资格迹，随后下一个状态的资格迹其实是上一个状态的资格迹乘以一个衰减因子，如果被访问了就会额外增加，也就是实现了一种被访问时增加，没有被访问时衰减的效果。在实际的更新中乘入更新表达式中，表示权重，或者说当前计算的值对于实际的贡献。这里的变量 c_s 也是同样的效果。

我们的目的是为了纠正计算中出现的偏差，很显然这里可以选择让c_s的取值为上面的重要性抽样的权重，这样可以得到无偏的对于更新值的估计如下：

$$\Delta Q(s_t, a_t) = \gamma^t \left(\prod_{1 \le s \le t} \left(\frac{\pi(a_t|s_t)}{\mu(a_t|s_t)} \right) \right) \delta_t \tag{3-10}$$

但是和之前的MC类似，这样会有很大的方差，因为它的计算方式和MC的方法类似，相当于逐节点的将重要性抽样的权重乘上去来获得最终的无偏的估计。根据方差的计算公式$Var(X) = \mathbb{E}(X^2) - \mathbb{E}(X)$，此时的$X = p(x)f(x)/q(x)$。对于采样比$p(x)/q(x)$没有限制，因此可能很大。

此外，一些方法提出使用$c_s = \lambda$或者$c_s = \lambda\pi(a_t|s_t)$，当定义$c_s$为一个常数的时候，避免了上面使用重要性的时候，无法控制其大小导致高方差的问题，取值可控。但是它要求采集策略和更新策略间足够接近（$\|\pi - \mu\|_1 \le (1-\gamma)/\lambda\gamma$）。在实际使用中并无法实时保证这一点。之后TB（$\lambda$）在其之上乘以每个时刻的学习策略，解决了不安全的问题，但是在on-policy的时候会有不必要的损失。在这些算法的基础上，Retrace(λ)算法采用了

$$c_s \overset{\text{def}}{=} \lambda \min\left(1, \frac{\pi(a_t \mid x_t)}{\mu(a_t \mid x_t)}\right) \tag{3-11}$$

和直接使用重要性抽样相比，该方法将采样率剪切到1，避免了方差过大的问题。同时又结合了后两种的优点，更加的高效和安全。

结合上面的结论，本文将其推广到使用价值函数V的情况，类似的定义：

$$\delta_t^V = r_t + \gamma V(s_{t+1}) - V(s_t) \tag{3-12}$$

之后仿照式（3-9），可以写出对应的转换：

$$\mathcal{T}V(s_t) = V(s_t) + \mathbb{E}_\mu \left[\sum_{t \ge 0} \gamma^t \left(\prod_{s=0}^{t-1} c_s \right) \delta_t^V \right] \tag{3-13}$$

$\mathcal{T}$是我们定义的基于价值函数的off-policy的算法算子。

为了便于后续的说明，对公式（3-13）进行展开转换重写：

$$\begin{aligned}
&\mathcal{T}V(s_t) \\
&= V(s_t) + \mathbb{E}_\mu \left[\left(\gamma^0 \cdot 1\right)\left(r_0 + \gamma V(s_1) - V(s_0)\right) + \left(\gamma^1 \prod_{s=0}^{0} c_s \right)\left(r_1 + \gamma V(s_2) - V(s_1)\right) + \dots \right] \\
&= V(s_t) + \mathbb{E}_\mu \left[\left(r_0 + \gamma V(s_1) - V(s_0)\right) + \left(\gamma^1 c_0\right)\left(r_1 + \gamma V(s_2) - V(s_1)\right) + \dots \right] \\
&= V(s_t) + \mathbb{E}_\mu \left[-V(s_0) + \sum_{t \ge 0} \gamma^t \left(\prod_{s=0}^{t-1} c_s \right)\left(r_t + \gamma(1 - c_t)V(s_{t+1})\right) \right]
\end{aligned}$$

$$= (1 - \mathbb{E}_\mu)V(s) + \mathbb{E}_\mu \left[\sum_{t \geq 0} \gamma^t \left(\prod_{s=0}^{t-1} c_s \right) (r_t + \gamma(1 - c_t)V(s_{t+1})) \right] \tag{3-14}$$

因为 $\mathbb{E}_\mu V(s) = \sum_a \mu(a|s)V(s)$，并且我们有 $\sum_a \mu(a|s) = 1$，所以 $\mathbb{E}_\mu V(s) = V(s)$，上式变为：

$$\mathcal{T}V(s_t) = \mathbb{E}_\mu \left[\sum_{t \geq 0} \gamma^t \left(\prod_{s=0}^{t-1} c_s \right) (r_t + \gamma(1 - c_t)V(s_{t+1})) \right] \tag{3-15}$$

接下来为了说明我们提出来的方法是有效的，我们需要证明上面定义的算子是可以收敛的，此时，存在唯一的不动点。

引理 3.1 在前面的符号定义下，给定如下关系：

$$A = \sum_{t \geq 0} \gamma^{t+1} \prod_{s=0}^{t-1} c_s \tag{3-16}$$

$$B = \sum_{t \geq 0} \gamma^{t+1} \prod_{s=0}^{t} c_s \tag{3-17}$$

我们可以得到 $A - B < 1$。

证明 因为 $\gamma \leq 1$，显然 $B \geq 0$，所以 $\gamma B \leq B$，因此我们有

$$\begin{aligned} A - B &\leq A - \gamma B \\ &= \gamma + \gamma^2 c_0 + \gamma^3 c_0 c_1 + \cdots - \gamma(\gamma c_0 + \gamma^2 c_0 c_1 + \gamma^3 c_0 c_1 c_2 + \dots) \\ &= \gamma + \gamma^2 c_0 - \gamma^2 c_0 + \gamma^3 c_0 c_1 - \gamma^3 c_0 c_1 c_2 + \cdots - \gamma^{t+2} \prod_{s=0}^{t} c_s \\ &= \gamma - \gamma^{t+2} \prod_{s=0}^{t} c_s \\ &< \gamma < 1 \end{aligned} \tag{3-18}$$

定理 3.1 给定算子 $\mathcal{T}$ 如式（3-13）所示，我们有

$$\|\mathcal{T}V_1(s) - \mathcal{T}V_2(s)\| \leq \eta \|V_1(s) - V_2(s)\|_\infty$$

也即 $\mathcal{T}$ 是一个收缩的映射。

证明 在引理 3.1 下，我们完成对上式的证明。

$$\begin{aligned} &\mathcal{T}V_1(s) - \mathcal{T}V_2(s) \\ &= \mathbb{E}_\mu \left[\sum_{t \geq 0} \gamma^{t+1} \left(\prod_{s=0}^{t-1} c_s \right) (1 - c_t)(V_1(s_{t+1}) - V_2(s_{t+1})) \right] \end{aligned}$$

其中当 $t = 0$ 时 $\prod_{s=0}^{t-1} c_s = 1$。又因为式中权重部分基本都处在 $[0,1]$ 的取值范围内，同时又有

$$\mathbb{E}_{\mu}\left[1-c_t\right]=\mathbb{E}_{\mu}\left[1-\lambda min\left(1,\frac{\pi\left(a_t\mid x_t\right)}{\mu\left(a_t\mid x_t\right)}\right)\right]\geq\mathbb{E}_{\mu}\left[1-\lambda\right]\geq 0$$

因此所有的权重累乘是非负的。在此基础上对于累计的参数和，我们有：

$$\mathbb{E}_{\mu}\left[\sum_{t\geq 0}\gamma^{t+1}\left(\prod_{s=0}^{t-1}c_s\right)\left(1-c_t\right)\right] \tag{3-19}$$

$$=\mathbb{E}_{\mu}\left[\sum_{t\geq 0}\gamma^{t+1}\left(\prod_{s=0}^{t-1}c_s-\prod_{s=0}^{t}c_s\right)\right] \tag{3-20}$$

$$=\mathbb{E}_{\mu}\left[A-B\right] \tag{3-21}$$

$$<\mathbb{E}_{\mu}\left[\gamma\right]<1 \tag{3-22}$$

因此证明了$\mathcal{T}$是一个收缩的映射。

那么对于值函数来说，根据式（3-13）以及式（3-12），可以得到纠正后的实际的值函数的计算规则如下：

$$v_t=V\left(s_t\right)+\delta_t^V+\gamma c_t(v_{t+1}-V(s_{t+1})) \tag{3-23}$$

其中V为原来的值函数的值，v为纠正之后的值，根据该式可以很方便的从反向开始根据v_{t+1}来计算出v_t。

3.4 基于梯度的行动者

行动者中使用基于梯度的方法来进行学习，在行动的同时更新行动策略。策略梯度的更新方式一般形式如式（2-10）所示，一般我们取带折扣因子的优势函数：

$$G(\tau)=A^{\pi,\gamma}(s_t,a_t)=Q^{\pi,\gamma}(s_t,a_t)-V^{\pi,\gamma}(s_t) \tag{3-24}$$

来作为更新的损失函数的一部分，和传统博弈论当中的遗憾值类似，优势函数表征了在状态上的每个动作相比整个状态的回报来说，实际的优势，用它来衡量一个动作的好坏，因此使用其来进行动作策略的损失函数是合适的。为了得到更加稳定的数据，这里使用广义优势估计(Generalized Advantage Estimator, GAE)[45]的算法来进行处理，首先定义一个前k个TD误差的带折扣因子的和为：

$$\hat{A}_t^{(k)}=\sum_{l=0}^{k-1}\gamma^l\delta_{t+l}^V=-V(s_t)+r_t+\gamma r_{t+1}+\cdots+\gamma^kV(s_{t+k}) \tag{3-25}$$

之后使用λ来进行加权得到最后的估计，如下：

$$
\begin{aligned}
\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} &:= (1-\lambda)\left(\hat{A}_t^{(1)} + \lambda\hat{A}_t^{(2)} + \lambda^2\hat{A}_t^{(3)} + \ldots\right) \\
&= (1-\lambda)\left(\delta_t^V + \lambda\left(\delta_t^V + \gamma\delta_{t+1}^V\right) + \lambda^2\left(\delta_t^V + \gamma\delta_{t+1}^V + \gamma^2\delta_{t+2}^V\right) + \ldots\right) \\
&= (1-\lambda)\left(\delta_t^V\left(1 + \lambda + \lambda^2 + \ldots\right) + \gamma\delta_{t+1}^V\left(\lambda + \lambda^2 + \lambda^3 + \ldots\right)\right. \\
&\quad \left. + \gamma^2\delta_{t+2}^V\left(\lambda^2 + \lambda^3 + \lambda^4 + \ldots\right) + \ldots\right) \\
&= (1-\lambda)\left(\delta_t^V\left(\frac{1}{1-\lambda}\right) + \gamma\delta_{t+1}^V\left(\frac{\lambda}{1-\lambda}\right) + \gamma^2\delta_{t+2}^V\left(\frac{\lambda^2}{1-\lambda}\right) + \ldots\right) \\
&= \sum_{l=0}^{\infty}(\gamma\lambda)^l\delta_{t+l}^V
\end{aligned}
\tag{3-26}
$$

其中 $\lambda = 0$ 时具备小方差但是有误差，$\lambda = 1$ 时准确但是有较大方差，因此通过调整 λ 来平衡误差和方差。

3.5 整体算法结构

结合以上的说明，可以得到最终基于行动者-学习者结构的算法。算法分为相对独立的两部分，其中行动者的算法如算法 3-1 所示。

算法 3-1 行动者算法

Input: 环境名称 $level$，采样时间步 T。

Global: 网络参数 θ、w，公共队列 Q，停止标志 $Flag = False$。

```
Function Actor(level):
    根据 level 构造环境 Env;
    repeat
        同步参数 θ, w。抽样起始状态 s。t ← 0;
        repeat
            a ~ π(a|s);
            s, r ← Env(a);
            t ← t + 1;
        until t > T;
        发送采集的经验到 Q;
    until Flag == True;
return
```

行动者根据输入的环境名称创建环境进行环境的交互，同时通过全局的变量来控制开始和关闭，在开启的过程中，始终向队列中发送采集的数据。每次采集 T 个时间步，采集结束的时候尝试同步网络参数。此时的学习者仅

仅负责采集数据，学习者则从行动者采集的数据当中抽样数据进行实际的策略网络和值函数网络的参数更新。学习者的算法如算法 3-2 所示。

算法 3-2 学习者算法

Input: 当前的训练帧数 F_{now}。批量大小 B。采样时间步 T。

Global: 网络参数 θ，w。总的训练帧数 F_{max}。公共队列 Q。停止标志 $Flag = False$。

Output: 公共的网络参数 θ，w。

Function `Learner()`**:**
- 初始化网络参数 θ，w;
- 根据需求构造学习者;
- **repeat**
  - 从队列中获得 B 条数据;
  - 使用式 3-3, 3-23, 3-26 计算 $V(s_t)$，纠正后的 v_s 以及 $\hat{A}_t^{\text{GAE}}$;
  - $d\theta \leftarrow \nabla_\theta log\pi_\theta(a_t|s_t)\hat{A}_t^{\text{GAE}}$;
  - $dw \leftarrow (v_s - V_w(s_t))\nabla_w V_w(s_t)$;
  - 更新参数 $\theta \leftarrow \theta + d\theta, w \leftarrow w + dw$;
  - $F_{now} += B \times T$;
- **until** $F_{now} \geq F_{max}$;
- $Flag = True$;

return

最终的更新策略为：对于参数化表示的值函数 V_w，以及参数化表示的策略 π_θ，对于学习者而言，这里使用 L2 Loss 来进行学习，目标是前面的修正之后的值函数 v_s。因此学习者的更新梯度为：

$$\nabla v = (v_s - V_w(s_t))\nabla_w V_w(s_t) \tag{3-27}$$

对于行动者来说，使用前面提出的 GAE 来作为整体损失函数的一部分，结合策略梯度的更新公式，得到最终的更新梯度为：

$$\nabla \pi = \nabla_\theta log\pi_\theta(a_t|s_t)\hat{A}_t^{\text{GAE}} \tag{3-28}$$

同时，像其他的 AC 算法一样，为了平衡探索和利用，防止过早的收敛，最终加上策略的熵正则项来作为惩罚：

$$-\nabla_\theta \sum_a \pi_\theta(a|s_t)log\pi_\theta(a|s_t) \tag{3-29}$$

另外，针对多个环境中出现的奖励尺度不一的问题，本文也对其进行了相应的处理，使用了奖励剪切技术，来将各个环境间不同的奖励映射到$[-1,1]$之间，来避免个别的任务因为奖励过大而学习的优先级更高，导致其他的问题没有很好的训练效果的问题。

3.6 本章小结

本章对传统强化学习向多任务推广中存在的一些问题，提出了对应的解决方法。为了优化数据的采集速度，使用了行动者-评论者的框架类似的行动者-学习者的结构，但是对其中的数据采集和策略学习两部分进行解构，使得两部分不再相互制约。同时针对分离形成的离线策略的性质，对根据采集的数据计算出来的值函数以及优势函数进行校正。主要将 Retrace(λ) 的方法进行推广到值函数上，并证明其具备收敛性，将广义优势估计使用在整体框架中，来进一步的校正策略梯度方面的问题。在这些方法的基础上提出了最终的多任务强化学习的整体算法，随后的实验部分验证本框架的有效性。同时更进一步的工作也是在在这个算法的基础上继续进行的。

第 4 章 基于辅助任务的多任务强化学习算法优化

4.1 引言

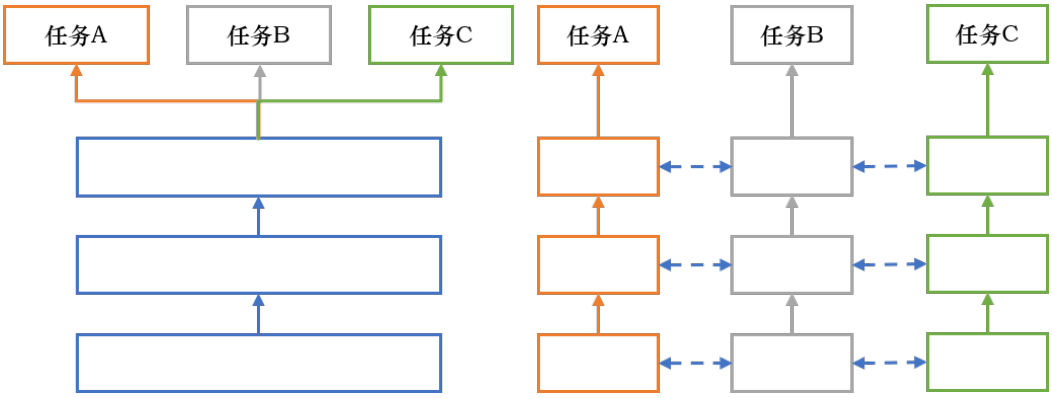
多任务学习中的任务之间往往具备主次的属性，因此使用辅助任务来帮助主任务学习是一种常用方法，目的是在关注主任务的同时，使用额外的附加任务来使得主任务获益。辅助任务一般选择和主任务相关的子任务来加强主任务的学习，例如在自动驾驶时配合物体检测，路标识别等方便主任务的判断，但是并不是所有的任务都可以找到密切相关的辅助任务，因此在难以获得相关任务的情况下也可以选择对抗任务来辅助，通过学习对抗任务来加强主任务。本章从这一想法出发，使用辅助任务来优化本文的智能体的效果，提升在环境中的表现性能。

4.2 基于辅助任务的多任务学习

多任务学习可以说是一种推导迁移的学习方法，使用相关任务的训练信息当中具备的一些领域相关的知识来辅助主任务的学习，提升主任务的泛化效果。因为存在多个任务，所以也就存在多个损失函数，因此从学习的角度来说，多个任务间的梯度是同时进行更新的。多个任务之间存在关联，这种关联使得各个任务之间的更新可以相互影响。关联的表示形式是底层的网络之间的共享表示，让它们之间相互影响共同学习。

在增加了额外的辅助任务之后，需要对本文的网络结构进行调整，根据采取的共享形式的不同，对网络的修改程度也不同。图 4-1 中展示了常见的两种共享的形式，图 4-1 a) 为硬共享，也即各个不同的任务之间共享一段神经网络，之后对于每个单独的任务来说，有自己单独的特定任务的输出层，这样在梯度更新时，在前面部分的网络就会受到所有任务的梯度的影响，达到分享经验的目的，但是后面的部分网络又保留了自己任务的特征，不会完全被其他任务所影响。图 4-1 b) 为软共享，每个特定的任务都具备自己独特的网络，各个网络之间是并行的关系。为了达到共享的目的，前面的隐藏层部分增加了额外的约束条件，通过约束条件来达到共享的目的，这种想法在许多方面都可以找到应用，例如有的方法为了学习到新的知识，就可以再新增一层网络，继续学习。

两种方式中软共享的实现相对困难低效，如果有更多的任务加入学习，



a) 硬参数共享　　b) 软参数共享

图 4-1 多任务共享经验的形式

那么对于网络的约束就变得更加的难设计，也就意味着更新的时候计算起来也会更加缓慢。同时因为网络层数的原因，也会导致网络的约束过多。同时如果现在有 n 个任务，那使用软共享就意味着需要 n 倍的参数，同时还要计算 n 组参数之间的约束关系，这种消耗是很大的。所以在本章并不使用软共享的方式来进行后续的设计，而是使用硬共享的方式来完成多任务的添加。

强化学习的常用网络特征比较简单明显，比较容易选择需要共享的区域。网络一般是特征提取的卷积神经网络部分和另外的网络的结合体，根据需要，另外一部分网络可以是长短期记忆网络（Long Short-Term Memory，LSTM）来加强对之前的记忆或者是直接全连接层输出需要的维度作为对上文提到的各种函数的估计。在决策的过程中，所有的任务都是在当前的画面基础上进行的，因此对图像的特征提取是所有的任务都需要的，所以在网络的设计上，将前一部分的视频特征提取部分使用共享的网络结构，后面部分根据每个辅助任务的不同使用不同的结构来产生不同的输出以及设计对应的损失函数来进行学习。

在本课题中主任务就是强化学习智能体的策略学习，也就是上文所述的基于行动者-学习者的多任务强化学习算法的学习过程。首先进行主任务的网络设计，主任务的神经网络如图 4-2 所示。目标是直接通过环境的输出做出决策，实际采集的数据是批量的时序信息，实验时设定一个时间步 T 以及批量大小 B，行动者采集的数据都是 T 步的一个序列，学习者从中取 B 条序列，也就是共计 $B \times T$ 个数据进行训练。

由环境输出的实际图像作为网络的输入，输入的时候通过除以 255 获得最后的浮点数输入。随后将输入传递到卷积层，卷积层由多层相似的结构组成，每层包含了一层卷积和多层残差网络，残差网络由两层卷积组成。之后

的结构因为训练时间比较长，同时处理的是具备时间特征的时序信息，因此在之后使用LSTM来进一步处理。因为使用的算法需要进行参数化的策略和参数化的值函数估计，因此LSTM输出的结果分为两个分支，通过全连接层输出，分别作为所需的值函数和策略的估计。

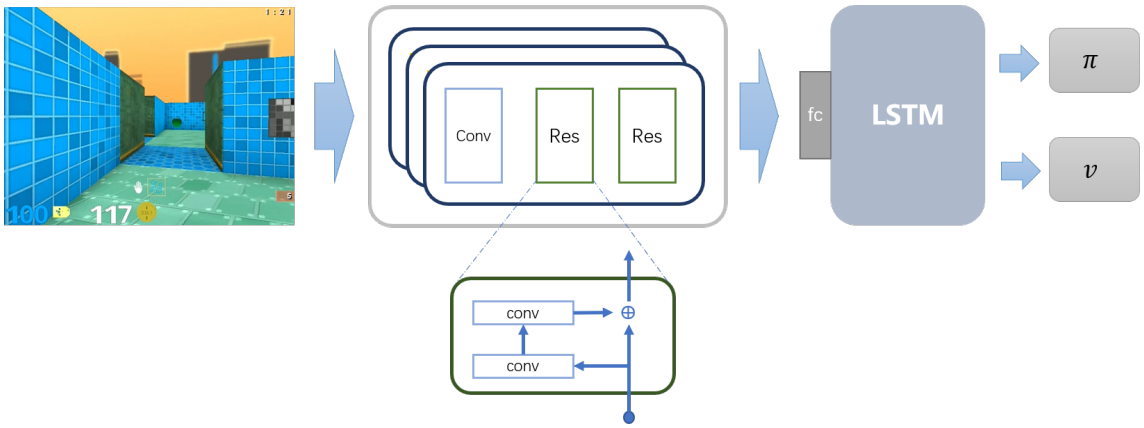


图 4-2 主任务的网络结构

4.3 任务认知辅助任务

在之前的实验场景中，学习过程是将各个环境中的数据混在一起，学习者从经验池当中抽样进行学习。在这个过程中智能体，并不知道每条经验是属于哪一个环境的，因此可能将一个环境中的输入图像错误的认为是另一个环境的输入，从而进行决策的时候执行了错误的行动。所以考虑设计辅助任务使得智能体可以更好的区分当前的画面是属于哪个任务的，借此来增强智能体在不同任务上的决策能力。

对于多任务间的区分的方案，有许多的方法进行了探索。常见的方法是将任务的标记作为训练的一部分，例如对于第 i 个环境，将其独热编码映射为向量 V_i，之后使用该向量参与训练。通过将该向量输入网络进行处理，输出和任务 id 相关的中间结果，之后用此结果参与计算。这里参考 FiLM[46] 的结构进行说明。对于上面的向量 V_i，首先计算

$$\begin{aligned} \gamma &= f(V_i) \\ \beta &= h(V_i) \end{aligned} \tag{4-1}$$

也即从该向量中学习到两个输出，作为参数，对主要的输入 I 进行后续的计算，相当于新的一层网络结构，之前计算的参数作为权重和偏差值参与计算：

$$\mathbf{F} = \gamma * I + \beta \tag{4-2}$$

这种方法在FiLM的问答系统中确实起到了很好的效果，但是在将其迁移到本文的环境的时候出现问题，训练曲线呈现平滑状态，并没有从实际的学习中学习到经验。因为对于输入而言，本身是十分稀疏的结构，一个网络将其作为输入学习到的参数本身很难说是有效的表征了这个任务序号。

实际实验中也尝试了数据挖掘中常用的特征直接进行拼接，将任务 id 拼接到特征中向后传播，依旧没有起到应有的效果。考虑到任务 id 的稀疏性，这里使用一种简单有效的方法，可以将整个任务其看作是一个多分类任务的一个需求，也就是智能体根据输入的图像判断这个图像是属于哪一个任务的，输出对应的结果。将任务id作为输出而不是输入就可以避免怎么将这一特征处理为输入的问题。具体来说，此时使用上文中拓展的状态空间 $S = \{\{s_j, i\}_{s_j \in S_i}\}_{i=1}^{T}$ 来作为本任务的动作空间。在该任务的学习过程中要求对环境进行编号，将其随着环境经验的采集一起传入进行训练。

对于 K 个环境下的多任务强化学习，如果一个状态 $S_i = \{s_i, t_i\}$，其中 t_i 表示对应的任务id，经过主网络之后的神经网络的输出为 p_i，那么当前任务的损失函数就是：

$$\mathcal{L}_{cat} = -\sum_{i}^{K} y_i log(p_i) \tag{4-3}$$

4.4 控制辅助任务

为了更进一步的增强智能体的性能，考虑新增有助于智能体分析当前环境的控制辅助任务。通过这些任务让智能体更好的理解画面，或者说更好的处理马尔科夫决策过程中的数据。对于智能体来说，对环境的理解来自于图像的输入以及奖励的反馈。本节主要从图像的方面着手，提出辅助智能体动作决策的任务，通过强化对图像的分析，来强化智能体的性能。

本课题选择的环境是一个3D的环境，主要选择的任务是探索收集类的任务。因为环境的复杂性，所以智能体想要取得比较好的成绩，首先要可以分辨环境中的各种元素，例如区分环境中的走廊和墙，找到可以走的通路，不至于卡在原地，或者一直向墙的方向走。区分苹果等收集的东西来更好的获得收益，区分门来走到新的分区等等。从人的观念出发，我们在玩游戏的时候，可以发现特殊的物体和周围的环境之间有明显的区分，这一表现在画面上就是像素的变化，从图像的像素的界限来区分出每个物体的位置。因此注意到在游戏画面中，如果突然出现了和旁边的像素区别很大的区域，那一般就代表这一个新的物体或者不同的环境元素的出现，如果智能体可以区分

当前的输入中各个部分的差异，那么对于智能体进行环境的探索或者是物品的收集是很有利的。其实这一想法的目的就是训练智能体的图像分割能力，如果智能体可以很好的分割出元素的位置，那么对于决策来说就很有利。

基于这一想法，设计任务来使得智能体可以分辨视野中的各种元素，给出正确的评分。可以发现在主任务前面的特征提取中，使用了卷积层来进行数据特征的提取，得到了提取的视频特征输出，而现在的目的是使得智能体可以分辨画面中的元素，这里使用类似于图像分割领域中的一些方法。在图像分割领域中使用使用全卷积神经网络（Fully Convolutional Networks，FCN）的想法来对其进行处理。FCN[47] 主要包括两部分：卷积和反卷积。一般情况下，卷积网络是在卷积层之后使用全连接层继续处理，得到固定长度的特征向量，随后使用这一向量来进行分类，而 FCN 则是通过接受任意大小的图像作为输入，使用反卷积层对最后得到的特征进行上采样，最终使它的尺寸变为和输入图像相同。因为需要对经过卷积变换过的图像进行再次的预测，所以使用反卷积使得图像变回原来的大小。

之后在经过反卷积之后的结果作为新的输入来继续后续的工作。为了衡量网络对像素预测的性能，需要一个标准来进行评判。整个处理过程使用新的强化学习来进行预测。输入为反卷积之后的结果，新的奖励值为像素变化的情况。像素变化的计算方法如图 4-3 所示，对行动者的环境进行额外的处理，在采集数据的时候同时保留上一帧的情况，在获得新的一帧的图像之后，开始计算两帧之间的像素变化情况。

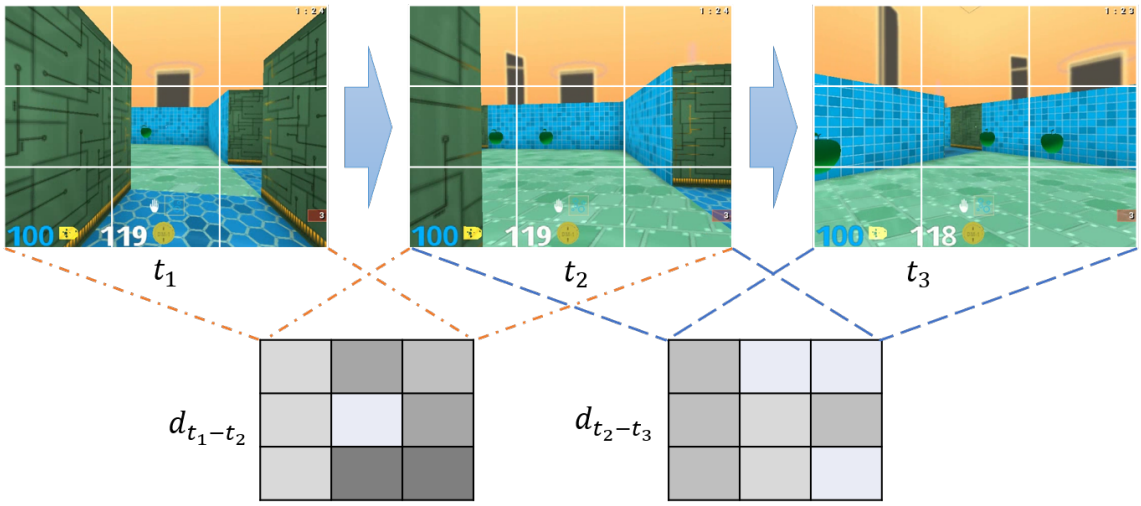


图 4-3 计算每一帧之间的像素变化

为了简化计算的过程，首先将每一帧的图像分为几个部分，例如图中将其分为 3×3 的区域，之后以分区为单位计算前后两帧之间计算差值，按照

分割的区域再求和作为每个区域的像素变化取值。对于一个 t 时刻的图像 S_t，假设将其分为了 n 个区域，也就是 $S_t = \{Z_t^1, Z_t^2, \ldots, Z_t^n\}$，并且使用 Z^i 表示任意图像中第 i 个区域（因为每个图像在计算的时候都被分为了 n 个区域，所以这种表示没有歧义）。那么两帧的某个区域 i 之间的差值为：

$$\Phi(Z_t^i, Z_{t+1}^i) = \sum_{(x,y)\in Z^i} S_t(x,y) - S_{t+1}(x,y)$$

所以两帧之间的实际的差值矩阵为

$$\hat{r}_t = \left\{\Phi(Z_t^1, Z_{t+1}^1), \Phi(Z_t^2, Z_{t+1}^2), \ldots, \Phi(Z_t^n, Z_{t+1}^n)\right\}$$

之后使用这个差值矩阵作为新的任务的奖励来进行新的更新过程，此时的奖励不在是一个标量值，而是一个差值矩阵。目的是让网络的输出尽可能的接近新的回报值。这里计算回报值也使用和式（2-3）一样的 γ-折扣的计算方式，从时间步 t 到 n 的回报为：

$$\hat{G}_t^n = \hat{r}_t + \gamma\hat{r}_{t+1} + \cdots + \gamma^{n-t}\hat{r}_n = \sum_{i=t}^{n} \gamma^{i-t}\hat{r}_i \tag{4-4}$$

在获得了奖励值之后，所需的强化学习经验就已经被定义出来，接下来定义新的强化学习过程。新的强化学习过程使用 Dueling DQN[48] 中的类似的结构来计算最后的 Q 值。和之前的结构不同的是，Dueling DQN 中使用了额外的输出，通过值函数和优势函数两个估计器来实现 Q 函数的计算，而不是直接通过全连接输出估计结果。当值函数网络的参数为 α，价值函数网络的参数为 β，共同部分的参数为 w，那么形式化的描述 Q 的计算过程为：

$$Q(s,a,w,\alpha,\beta) = V(s,w,\alpha) + \left(A(s,a,w,\beta) - \frac{1}{|\mathcal{A}|}\sum_{a'\in\mathcal{A}} A(s,a',w,\beta)\right) \tag{4-5}$$

相比较之前的结构而言，新的结构可以区分奖励到底是由状态带来的还是动作带来的，转态带来的奖励值一般不会影响到实际的动作，而优势函数的部分则说明了是由动作引起的，此时采取新的动作会带来回报的提升，和之前的相比粒度更细。

结合之前的方法，可以得到本任务的整体结构如图 4-4 所示，在之前主任务的基础上增加新的结构来完成附加任务的训练，LSTM 之前的结构是主任务的除了输出以外的全部结构，将任务直接放在 LSTM 之后。其中反卷积和前面的卷积起到了类似 FCN 的作用，对输出的结果进行上采样，得到更大的矩阵。整体的结构则是 Dueling DQN 的形式，分为两个分支，分别上采样到对应的大小输出到全连接层，作为最后的价值函数和优势函数的估计值，

最后通过式4-5计算出此时对于 Q 的估计值。

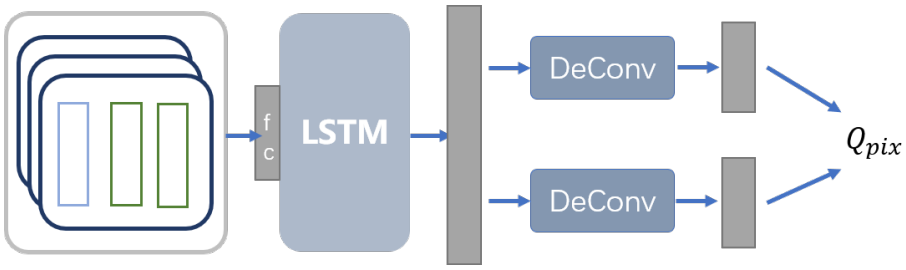


图 4-4 像素预测的整体结构

对于新的强化学习过程，将此任务的损失函数定义为 n 步的Q学习的损失函数：

$$\mathcal{L}_p = \mathbb{E}\left[\left(G_t^n + \gamma^n max_{a'} Q'(s,a') - Q(s,a)\right)^2\right] \tag{4-6}$$

和第3章提到的方法类似，可以在实际的计算中，使用反向计算的方式来方便的计算出来目标值，将 $max_{a'}Q'(s,a')$ 作为初始值，对于上一个的结果 v_t，新的结果 $v_{t-1} = G + \gamma \times v_t$，通过反向累计目标值来迭代的获得最后的结果。通过最小化L2正则来使得本文的网络更好的预测像素的变化情况。

4.5 整体结构

最终合并前面的方法，最终的整体结构图如图4-5所示。图中的淡灰色框为上文的卷积层，灰色的都代表全连接层。使用了3层相同的神经网络堆叠来做图像特征的提取层，其中每一层包含一层卷积和两层残差，每层残差网络包含两层卷积网络。输入的图像经过神经网络之后获得2个输出，其中1个用于多分类任务的训练，另外一个继续输入LSTM中进行训练，提取时序特征，输出3个结果。其中两个作为主任务的输入，训练策略网络和值网络，另一个则继续控制辅助任务的训练输入，此外还要加上实际行动策略 π 的交叉熵。通过将各个任务的损失函数叠加，即可得到最终的损失函数，进行整体的训练。

定义主任务的损失函数为 L_{main}：

$$\mathcal{L}_{main} = -\mathbb{E}_\pi\left[V(s_t)\right] + \mathbb{E}\left[(v_s - V_w(s_t))^2\right] \tag{4-7}$$

那么最终的训练结构的损失函数就是：

$$\mathcal{L} = \mathcal{L}_{main} + \lambda_1 \mathcal{L}_p + \lambda_2 \mathcal{L}_{cat} \tag{4-8}$$

式中的 λ_1, λ_2 为超参数。

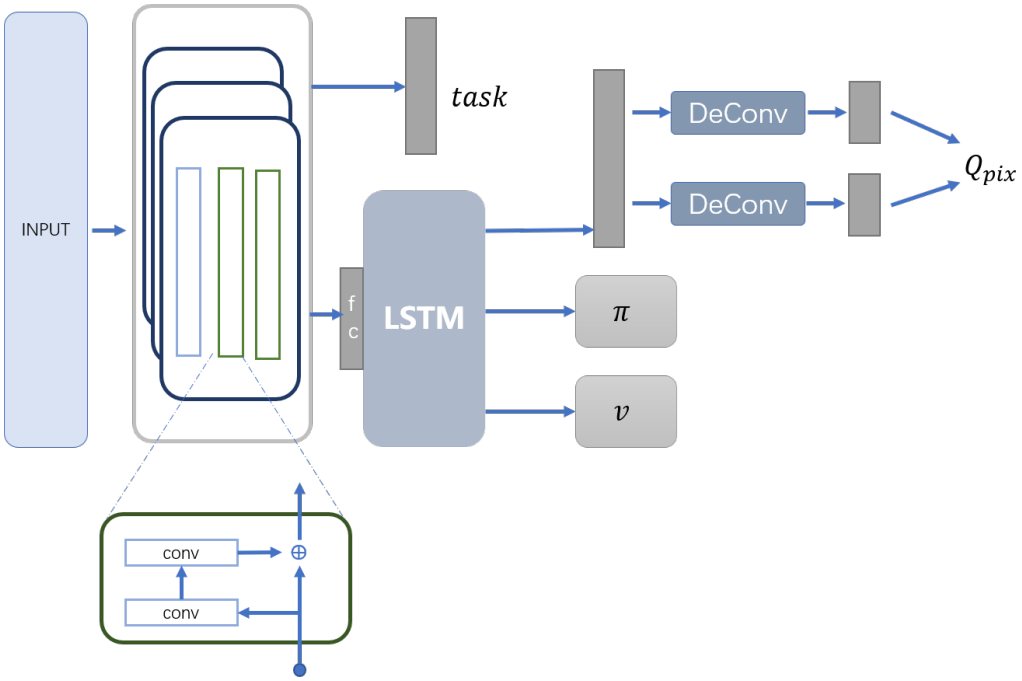


图 4-5 完整框架图

4.6 本章小结

本章提出了基于辅助任务的多任务强化学习的优化方案，分别提出任务认知和辅助控制的辅助任务，来帮助主任务学习。同时给出了每个辅助任务以及主任务的网络结构，在此基础上给出了整体的神经网络结构以及训练的损失函数，完善了上文所提出的算法，提升整体性能。

第 5 章 实验设计与分析

5.1 引言

本章是对本文中提出的多任务强化学习方法的实验验证与分析。通过对本文提出的方法进行实现，并在 DeepMind Lab 平台上进行训练和实验，与其他智能体进行对比来验证方法的有效性。本部分首先介绍实验的环境以及实验平台的特性，之后主要从多任务方法的有效性、辅助任务的有效性以及整体框架的有效性三个方面进行实验设计与验证。将结果和其他智能体进行性能比较，从实验数据上展示出了本文提出的方法的优越性。

本章所有的实验均在 Ubuntu 16.04.4 LTS 系统的 Linux 服务器上进行。服务器 CPU 为 32 个虚拟核心的 Xeon 系列 CPU，同时配备 4 块 P100 显卡。训练时一般采用 80 个行动者进行数据的采集，为了加快速度使用了更多的行动者的速度数据经过换算到和 80 个同等水平，在此基础上进行性能的比较。

5.2 DeepMind Lab 3D 环境

本文的实验环境选取 DeepMind 开发的 3D 环境 DeepMind Lab。DeepMind Lab 是一个完全 3D 的游戏类平台，专为基于智能体的 AI 研究而定制。该环境是一个部分可观察的环境，智能体仅可以看到视野内的物体，并且游戏中的场景可能还会进行随机的刷新等，因此是十分符合本文研究内容的非完全信息的场景。整个环境从第一人称视角，通过模拟物的眼睛观察。场景以丰富的视觉效果呈现。可用的操作允许智能体向四周观察，进行一些移动开火跳跃等操作。智能体自身是一个浮动的圆球，它通过激活与其所需运动方向相反的推进器来悬浮和移动。在总体场景下有很多内置的，也可以由玩家自由定制的环境。本文在此环境下选择了一些游戏场景作为后续的实验验证平台。

5.2.1 环境相关介绍

强化学习环境中最重要的部分就是观测的信息（Observation）以及智能体可以执行的动作，环境的变化由环境的内部机制自行计算，接下来介绍此环境中可以获得的观测信息，以及在后续实验中采取的动作集。

DeepMind Lab 中支持的操作如图 5-1所示。环境可以给出的信息包括奖

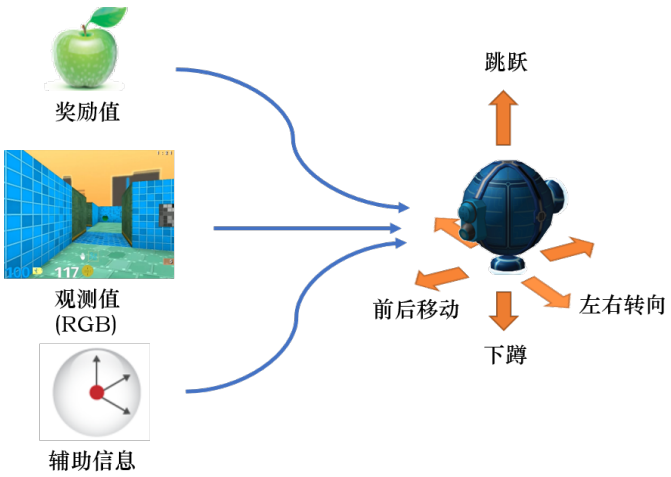


图 5-1 环境示意图

励值，图像信息以及一些辅助信息。DeepMind Lab 给出了丰富多样的观测信息，包括普通的 RGB 表示的图像，RGBD 表示的图像，速度信息等等。其中奖励值是一个标量，和具体的关卡规则相关。本文在观测信息方面选择使用 RGB 表示的图像信息。

动作方面图中展示了智能体自身可以做出的动作，其他的还可以进行移动视角，开火这些额外的动作。在 DeepMind Lab 中，动作表示为一个向量 $a=\{l_1,l_2,s,m,f,j,c\}$，分别代表视野的调整，左右转向（s），前后移动（m），开火（f），跳跃（j）蹲下（c）几种动作。为了更容易处理，采用了 DeepMind 在处理过程中常用的动作映射，将智能体的动作固定为表 5-1 所示的几种，基本囊括了常用的操作，主要是对视角进行了限制。

表 5-1 离散的动作映射表

动作名称	动作向量取值
前进	(0,0,0,1,0,0,0)
左转	(0,0,−1,0,0,0,0)
向左看	(−20,0,0,0,0,0,0)
左看直行	(−20,0,0,1,0,0,0)
开火	(0,0,0,0,1,0,0)
后退	(0,0,0,−1,0,0,0)
右转	(0,0,1,0,0,0,0)
向右看	(20,0,0,0,0,0,0)
右看直行	(20,0,0,1,0,0,0)

5.2.2 实验选取的游戏介绍

本文的实验部分选取了一组迷宫探索收集类的游戏，来进行实验验证，共 8 个游戏，可以分为 4 种规则的游戏。

• Object Locations。本任务要求智能体收集一个迷宫内所有的苹果，每个苹果给 +1 的奖励。每次的游戏有时间限制，苹果被放置在迷宫内的房间里，智能体必须在时间结束前收集尽可能多的苹果，以使自己的总分数最大化。当智能体在游戏时间使用完之前收集到所有苹果，关卡会重置，重复进行，直到所有的时间耗尽，一局游戏结束。苹果的位置、关卡迷宫的布局和主题是每次游戏随机的，但是来自于相同的概率分布。智能体的产生位置在每次重置时都是随机的。包含大地图和小地图两个游戏，大地图的时间更长。

• Goal Locations。这个任务要求智能体尽快找到目标对象。找到目标对象后，关卡重新开始。目标位置等信息是每局游戏随机的。智能体的产生位置在每次重置时随机化。分为大地图和小地图两个环境。每次收集到目标获得 +10 的奖励。

• Obstructed Goals。本任务要求智能体找到目标。相比较之前的游戏，现在增加了随机打开和关闭的门，因此增加了难度。当智能体找到目标后，关卡重新开始。目标位置、关卡布局和主题是每局游戏随机的。智能体的产生位置是随机的，每次都会重置。门的状态(打开/关闭)是每次重置时随机选择的，保证通往目标的路径始终存在。同样包含大地图和小地图两个游戏，大地图的任务与小地图的障碍物相同，游戏持续时间更长。训练和测试来自相同的分布。每次收集到目标获得 +10 的奖励。

• Object Rewards。该任务要求智能体收集放置在房间周围的人类可识别的物体。和前面的实验不同，这次有些物体有正向的奖励，有些则是负向的奖励。在收集完所有正向奖励类别的对象后，关卡重新开始。智能体产生位置、对象位置和每类对象的数量在每次重置时随机化。

5.3 多任务和单任务的性能对比实验

首先为了测试本文第三章工作的行动者-学习者多任务训练框架的有效性，验证多任务并行学习的可行性，进行单任务和多任务的性能对比实验。实验环境设置为之前选择的 8 个游戏环境，为了排除其他因素的影响，本节使用相同的智能体，也即相同的网络结构和训练参数。同时在多任务和单任务上都使用前文中的行动者-学习者框架。本节没有对比使用和不使用行动者-学习者框架的性能差异，因为这个框架本质上来说是多线程的多个环境的采集速度的叠加，此外行动者-评论者框架的性能对比已经可以说明这种结构可以得到数据采集的性能提升，所以这里没有进行额外的对比。

实际的训练时设定 8 个环境共计 10^8 帧的训练数据。保证多任务和单任务

都使用相同的帧数进行训练，不同的地方在于多任务是总的经验数总计 10^8 帧，而单任务进行了分割，平均每个任务进行 1.25×10^7 帧的训练，加起来总计 10^8 帧的数据。之后对他们训练完成的智能体进行 8 个环境的测试，测试的评分使用多次测试的平均值，同时统计 8 个单任务完成的总共的时间以及多任务完成的时间。结果对比如表 5-2 所示。

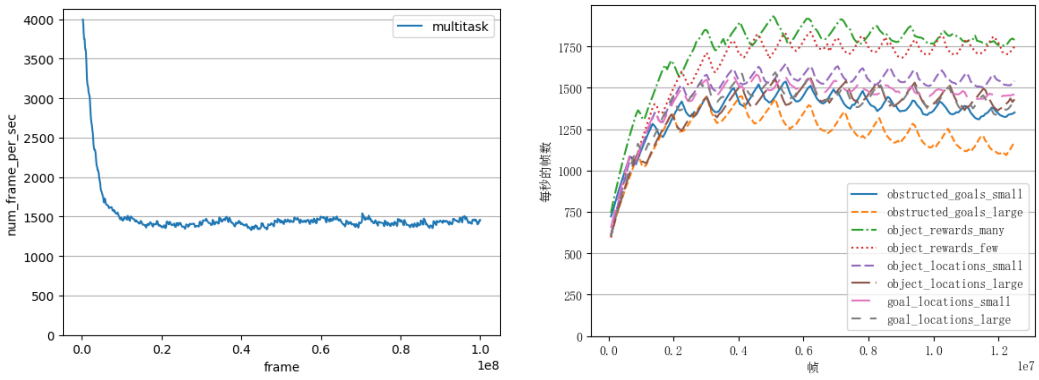
表 5-2 同等条件下单任务和多任务的性能对比

游戏名	多任务同时学习结果	单任务学习结果
object_locations_small	30.20	**45.90**
object_locations_large	**27.10**	15.70
obstructed_goals_small	**67.50**	10.20
obstructed_goals_large	**31.00**	2.50
goal_locations_small	**131.00**	40.90
goal_locations_large	**57.50**	6.70
object_rewards_few	12.30	**19.19**
object_rewards_many	15.65	**25.70**
总用时	**16.8h**	20h

可以发现在本节实验的环境中，多任务在 8 个任务中的 5 个取得了比单任务更好的成绩，单任务在小地图上的物品收集任务以及有正负奖励的物品收集任务中取得了更好的成绩。在更加困难的有阻碍的目标获取任务中多任务取得了优势，值得注意的是，多任务的优势往往是绝对优势，例如 obstructed_goals_large 任务中的 31 对比 2.5。这一结果也符合对多任务的期望，任务设计中每种类型的任务都有大小两种地图，我们期望可以从容易训练的小地图中更快的学习到经验，将其应用在更复杂的大地图中，使其受益，同时也期望可以将其他任务中学习到的经验互相共享，可以发现多任务很好的达到了预期。此外从整体来说，多任务采集到 10^8 帧的经验所用时间小于单任务的时间总和，进一步说明了这一方法的有效性。

为了探索多任务比单任务速度快的原因，在训练过程中记录下数据采集的速度，得到图 5-2所示的结果。图像的横轴为帧数，这里可以理解为训练中的全局时间步，纵轴表示每秒处理的帧数。图像展示了在对应 x 时间时的不同方法的处理速度。

为了可以更清楚的对比，图中增加了 y 轴对应的网格线。从图 5-2 a)可以发现，多任务混合训练的时候，采集速度稳定在了 1500 帧每秒的范围内。并且大部分时间内速率十分的稳定，一直持续到训练完成。图 5-2 b) 画出了每个任务单独训练的速度曲线，图中仅有正负奖励类的游戏整体采集速度大于 1500，处在 1750 帧每秒左右，其他有 4 个任务速率低于 1500，两个任务在



a) 多任务混合训练　　b) 单任务训练

图 5-2 单任务和多任务的速度性能对比

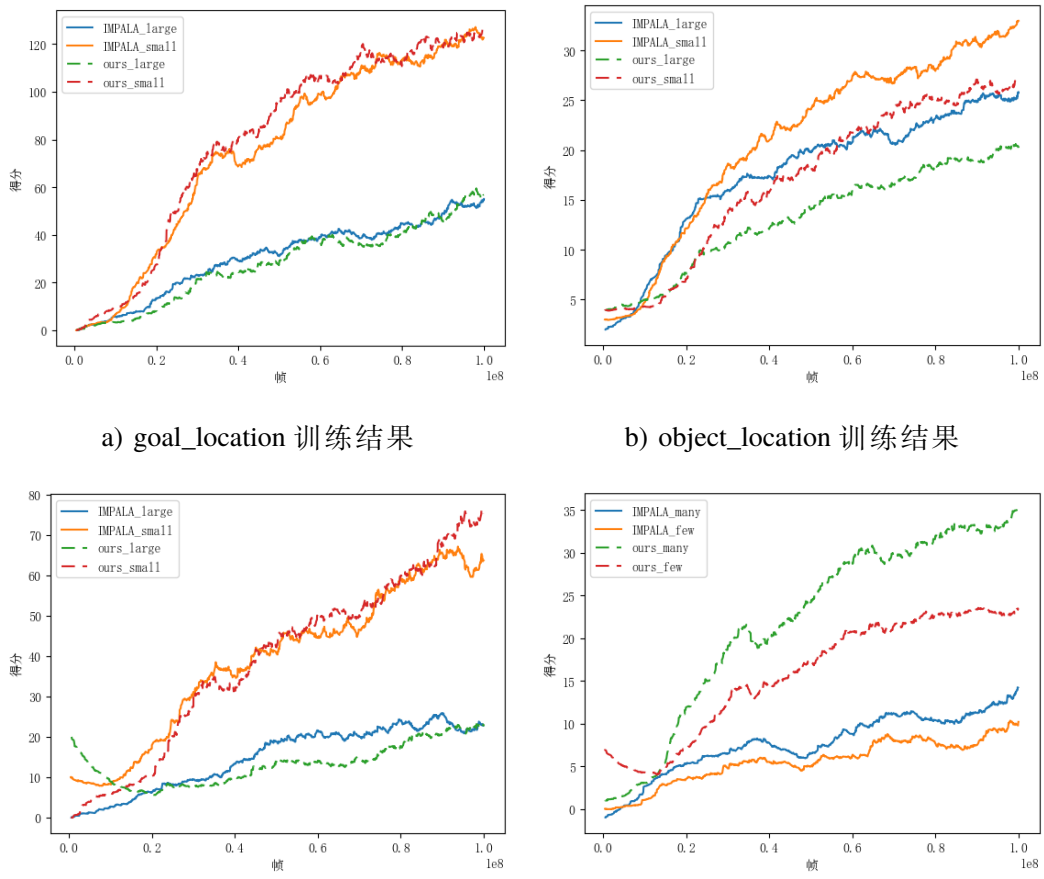
1500 附近。同时每个任务的收集速率十分不稳定，呈现规律性的上下起伏。

这一现象的原因在于开始进行训练时，行动者几乎是同时启动的，并且本文选择的实验环境都是限时取得更好的收益的任务，这就导致了相同环境的行动者的采集周期是完全重叠的。假设一个任务是限时 90 秒，那么同时启动的采集进程在 90 秒之后同时完成采集，将数据输入到队列中，此时学习者开始进行学习直至队列数据被耗尽。如果在数据耗尽前行动者不能完成数据的采集，就会出现学习者学习-停止-学习这样的循环过程，导致整体训练曲线的不稳定，图 5-2 b) 中曲线的规律起伏也说明了这一点。而多任务此时也体现了它的优势，因为任务的不同，不同行动者之间的采集周期相互岔开，使得整个采集训练过程更加的稳定。

5.4 任务认知辅助任务的有效性实验

在第 4 章中本文提出使用任务认知辅助任务来加强对环境的分辨能力，本节通过实验来验证其有效性。对比对象为当前同等设置下最好的智能体：DeepMind 提出的 IMPALA[49] 智能体（以下简称 IMPALA），本文的智能体则使用了任务认知的辅助任务来进行加强。首先在 8 个环境上使用相同的条件分别训练，得到图 5-3 所示的训练曲线。所有环境上的整体性能的训练曲线在下节中展示，本节主要关注智能体在具体的每个任务上的表现。

图中每个子图表示一类游戏的训练结果，对于一类游戏的两种不同尺寸的地图的子游戏，使用不同的颜色进行区分。可以发现在 goal_location 的任务中，对于小地图而言，大部分时间本文的智能体具备优势，而对于大地图则基本没有差距，总体上两种方法的差别不大。同样的在 obstructed_goal 上本文的方法和 IMPALA 性能相近。有较大差距的是 reward 场景和 object_location 场



a) goal_location 训练结果　　b) object_location 训练结果

c) obstructed_goal 训练结果　　d) reward 训练结果

图 5-3 不同环境下的训练结果对比

景，在 reward 场景中可以发现本文的智能体有了相当大的优势，而在之前对比的多任务智能体在此任务场景上表现不佳，从表 5-2 上可以看出，在 reward 场景上多任务的智能体均未胜过单任务训练的智能体。

这种情况也说明了多任务同时进行学习的一种弊端，多任务之间相互混杂很容易将其他环境学来的经验错误的应用到了另一个环境，使得这种环境的表现不佳。reward 场景和其他场景不同，它收集的物品并不都是正向的奖励，而其他的 3 类游戏只要收集到了物品或者是到达了目的地，都会获得一个正向的奖励。因此智能体为了获得更好的收益，会学习到收集物品这一技能，当此技能附加到了 reward 场景下就会出现问题，因为收集到的物品会减少奖励。同时收集物品得到正向奖励这一场景占绝大多数，所以训练就偏向去收集物品，也就导致 reward 的场景收益并不高。可以发现通过任务认知的辅助任务，能很明显的获得更优的结果，说明该辅助任务可以有效的让智能体分辨出具体的任务，从而减少将经验错误的应用的概率。

对于 object_location 场景，可以发现本文的方法比 IMPALA 要弱，出于优

化智能体性能的目的，进行了另外的一组实验来继续进行探索改进。目前本节的实验环境由4组不同的游戏构成，现在从中移除一组，将剩下的3组作为新的实验环境，这样组合之后就有了新的4组环境。随后在新的4组环境之上进行新的验证实验，来探究每组任务在4组任务组合中的角色，同时验证两种方法在面对不同的游戏组合下的鲁棒性。

因为实验环境包含多个具体的游戏场景，所以为了衡量一个智能体在整个实验中的性能表现，使用随机智能体的分数和人类水平的分数对其进行加权，其中随机分数和人类分数来自DeepMind的数据。对于K种游戏中的第i种，假设被测试智能体的分数为s_i，随机智能体的分数为r_i，人类水平的分数为h_i，那么定义被测试智能体在当前的环境下的得分为：

$$S_i = \frac{s_i - r_i}{h_i - r_i} \times 100 \tag{5-1}$$

在此基础上，对于K个环境的整体分数定义为：

$$S = \frac{1}{K}\sum_{i}^{K} S_i \tag{5-2}$$

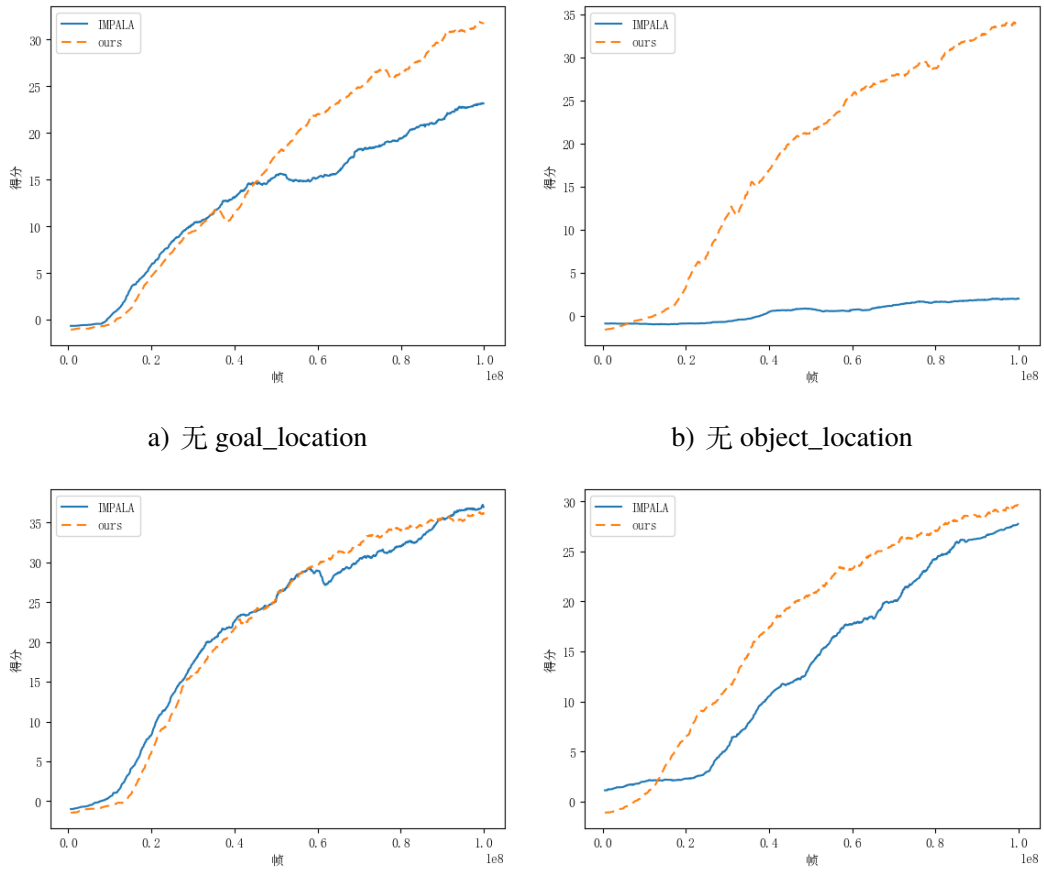
在之后的实验中对于整个环境的智能体评分均使用此定义。

在新的4组环境上的实验结果如图5-4所示。图5-4 c)的结果也验证了前文的结论，在去除了reward场景之后，两种方法几乎没有差距，说明reward场景的特殊性是智能体间的水平差距的重要原因。图5-4 a)和图5-4 d)的结果中可以观察到本文的方法仍旧对IMPALA有优势。以上的结果说明了第4章使用的任务认知方法可以有效的让智能体平衡收集奖励和获得分数，平衡reward中学到的经验和其他的普通收集任务中学习到的经验，这也是第4章的方法对比IMPALA效果好的原因所在。

需要额外注意的是，在图5-4 b)中，对比的IMPALA几乎完全失去了学习能力。object_location场景在整个游戏中占据很重要的地位，整体的实验环境的任务选择大概可以分为目标收集、在目标收集的基础上增加了障碍、仅仅收集一个目标（更加的稀疏，难以学习）、在收集的基础上增加了正负奖励这几个阶段。可以发现整体的设计上是一个递进的关系。而object_location中存在大量的目标可以让智能体有机会学习到收集目标这个技能。去除这个环境之后，就需要从其他的更困难的环境上进行学习，增加了学习的难度。

IMPALA的表现说明它在面对各种组合的时候并不具备很强的鲁棒性，会在失去简单任务的时候学习不到知识，也会在存在冲突的时候效果不佳。同时上面的实验也证明了使用辅助任务可以切实的增加面对不同环境的鲁

棒性以及各个环境间的知识应用能力。



a) 无 goal_location　　b) 无 object_location

c) 无 reward　　d) 无 obstructed_goal

图 5-4 不同环境组合下方法的鲁棒性

5.5 整体框架的性能对比实验

本节继续使用 8 个游戏作为实验环境，进行整体框架的性能比较。通过在智能体上逐步的使用本文提出的方法，同时进行智能体间的性能对比，来说明本文提出的各种方法的有效性。为了更好的说明框架中各个部分的作用，本节提出的三个智能体进行了消融实验，同时和同样设置下的最好的智能体 IMPALA 进行对比：

- ours_v1：在行动者-学习者框架的基础上使用了第 4 章中提出的任务认知辅助任务的智能体。
- ours_v2：使用了第 3 章中提出的校正技术的智能体。
- ours_final：使用了第 3 章和第 4 章中所有技术的智能体。

最终的实验图如图 5-5 所示，同时也在 8 个环境上进行了智能体性能的测试，各个环境的测试结果以及最终的测试结果详见表 5-3 以及表 5-4。表 5-

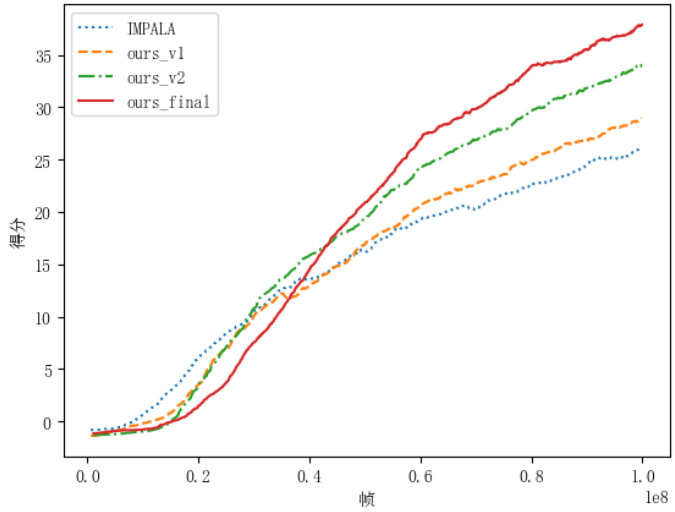


图 5-5 训练曲线对比

3 中展示了 IMPALA 和本文的方法在各个任务上的得分，表 5-4 中对比了本文的智能体和 IMPALA，A3C，以及 IMPALA 的一个非限制动作的扩展方法的整体性能得分。从整体性能上来说，可以发现各个版本的智能体性能都在 IMPALA 之上。增加了任务认知辅助任务的智能体提升了 reward 环境的奖励，同时对于 obstructed_goals 环境也有提升，因此使得整体的性能上升。从曲线上看，增加了辅助任务的智能体的前期增长趋势变得迟缓，应当是因为增加了额外的认知任务，使得任务间经验的分享受到了限制，但是因此后续的增长变得更加强劲，最终性能变得更加的优秀，说明这种损失是有价值的。

随后的 ours_v2 智能体应用了第二章提出的校正框架，对行动者和学习者学习的经验进行了数值上的校正。从曲线上体现出来的就是增长的趋势变得更加的陡峭，因此获得了更好的结果，比起 v1 版本的智能体在 IMPALA 上提升了 9.8%，v2 智能体在 IMPALA 的基础上增长了 20%，学习的能力比较 v1 的智能体更加的优秀，从每个环境上的得分来看，智能体在保持了原本的优势的基础上继续提升了 obstructed_goals 的结果，让整体的结果得以继续提升。

表 5-3 各个环境下的测试平均分数

游戏名	IMPALA	ours_v1	ours_v2	ours_final
explore_object_locations_small	30.20	26.00	**35.25**	33.05
explore_object_locations_large	**27.10**	23.10	25.00	25.75
explore_obstructed_goals_small	67.50	69.50	75.00	**97.50**
explore_obstructed_goals_large	31.00	19.50	31.00	**51.00**
explore_goal_locations_small	131.00	133.00	132.00	**158.50**
explore_goal_locations_large	57.50	68.00	66.00	**70.00**
explore_object_rewards_few	12.30	23.75	24.90	**25.00**
explore_object_rewards_many	15.65	34.30	31.40	**36.15**

最后在v2的智能体的基础上增加了额外的辅助任务，得到了最后完整的智能体ours_final，从表中可以看出最终的智能体在几乎所有的任务和最终成绩上都达到了最优结果，整体对比IMPALA提升了38%左右。在object_location任务上没有取得最优的结果，但是差距基本很接近，没有很明显的短板。从曲线上可以看出来，随着任务的新增，智能体的训练启动速度变得更加的缓慢，前期的速度比其他的智能体都要慢，但是之后的效果也更加的优秀。

表5-4 不同方法的综合得分

方法	综合得分
IMPALA	28.89
no-limited IMPALA	34.00
A3C	15.50
ours_v1	31.14
ours_v2	34.40
ours_final	**39.80**

在逐步使用本文提出的方法的同时，智能体的性能也在逐步的提升，说明了框架各个部分都是有效的，提升学习速率或者是让智能体的某个方面变得更强。同时最终智能体在测试中的优秀表现，也充分说明了本文提出的整体框架可以有效的提升多任务下的强化学习的性能，在各个任务上达到比较不错的性能权衡。

5.6 对智能体性能迁移的探索

为了研究智能体具体学习到了什么样的经验，尝试将学习到的智能体应用到其他的没有学习过的场景中进行测试，进行对智能体的性能迁移方面的一点探索。实际选取了 3 个场景作为实验平台：collect_good_objects、exploit_deferred_effects、natlab_varying_map_regrowth。分别侧重一种想要验证的智能体的能力，实验的结果如表5-5所示，表中给出了智能体的得分，IMPALA的得分以及对应游戏的人类水平用来进行对比。

collect_good_objects环境是一个和reward环境一样拥有正负奖励的环境。在这个环境中，智能体收集到红色的物体就会获得+1的奖励，收集到蓝色

表5-5 智能体迁移结果

游戏名	ours	人类得分	IMPALA
collect_good_objects	5.90	10.00	-1.40
exploit_deferred_effects	23.35	85.65	11.85
natlab_varying_map_regrowth	6.60	24.45	5.45

的物品会获得 −1 的奖励。此环境是为了验证智能体能否区分要收集的物体，是否真的学到了 reward 环境中的经验。在这个环境中，智能体的得分很高，智能体在有意识的搜寻红色的物体，并且尽可能的避开蓝色的物体，这说明智能体会去分辨物体的类别再开展自己的收集工作。此环境的优秀表现也说明了对于和原本参与训练的环境类似的游戏环境，智能体可以直接迁移过来，并保持原有的效果。但是需要注意的是如果刚好规则相反，那么在分数上智能体就会表现的很差，因为它没有接受过这个环境的训练，并不知道这个环境要求去收集的物品的类型。

exploit_deferred_effects 环境为了验证智能体是否掌握了训练环境中迷宫的行动技巧，环境包含两个房间，通过一个比较难发现的通道连接。智能体出现的初始房间中苹果的奖励很小，另一个房间的物品奖励大。智能体应当能够通过通道到达另一个房间，来获得更好的奖励，而不是在一个房间里一直收集。从结果来看，智能体很好的掌握了这一项技能，可以通过视野中的图像分辨出通道的入口，来在其中获得更好的奖励。

最后的 natlab_varying_map_regrowth 环境是更为复杂的自然环境，游戏中的时间是随机的（白天，黎明，夜晚），同时地形也更加崎岖不平。使用此实验来验证智能体是不是依赖环境，验证它能否在不同的环境中也可以很好的完成收集任务。从表中数据来看，智能体仍旧具备在复杂环境下收集物品的能力，但是因为地形的不同，性能有所下降，因为并没有学习到在复杂环境中行动的能力，经常会被环境中的石头等低小障碍挡住。

本节的结果说明了智能体在当前环境中进行训练之后确实掌握了相应的技巧，同时具备将这些技巧应用到没有见过的环境的能力，和 IMPALA 在这些环境中的表现进行对比，发现本文提出的方法在迁移方面更有优势。

5.7 本章小结

本章是对本文提出方法的实验设计验证的部分。通过充分的实验来证明本文所提出的非完全信息下的多任务学习算法以及基于辅助任务的优化方案的有效性。首先证明多任务学习对比单任务学习的优势，从整体性能到采集速度稳定性两方面展示多任务框架的优越性。随后针对提出的改进方案以及框架中的校正方案，设计单独的实验，分别说明各自在智能体中的有效性。同时设计了额外的实验验证本文的方案在不同环境组合中的鲁棒性。随后通过逐步实现本文的方案来进行性能对比展示了本文所提出方案是有效的。最终，进行了性能迁移方面的一些探索，展示了在性能迁移方面的潜力。

结　论

本文致力于非完全信息场景下多任务强化学习智能体的策略学习问题。针对单任务的智能体泛化性能不够以及训练性能差的问题，提出了基于行动者-学习者结构的多任务强化学习算法，并对此算法使用辅助任务进行了提升，训练出具备更好性能的智能体。本文的主要贡献如下：

（1）本文提出了多任务强化学习的算法。使用行动者-学习者结构来优化数据的采集性能，让智能体可以在相同时间内学习更多的数据，提升了智能体的训练效率。针对此框架导致的数据延时问题，提出了对应的数据校正方案。本文从行动者和学习者两个角度，分别通过将 Retrace(λ) 推广到值函数以及将广义优势估计器应用于优势函数的估计进行数据的校正，在此基础上得到新的梯度计算方式。结合两种校正方式，对整体的算法进行了完善。

（2）本文提出了通过辅助任务对多任务强化学习算法进行优化的方案。提出任务认知的辅助任务，让智能体可以更好的分辨具体的每一个任务场景对应的实际输入的特征。同时使用控制辅助任务来强化智能体对于画面的元素的认知能力，此外，使用的强化学习结构也进行了改变，强化智能体的认识能力。最终结合所有的任务，提出了整体的训练框架的网络模型，以及对应的损失函数。

（3）本文工作基于三维视频游戏这一复杂博弈环境得到了验证。从有效性、训练速度性能、各部分的消融实验对比、对环境的鲁棒性等几个角度对本文的工作进行了详细的验证和分析。通过和最新的IMPALA智能体在DeepMind Lab平台上进行对比实验，展示了本文方法对于当前的复杂环境是切实有效的。此外，通过迁移性的对比实验来验证所实现的智能体是否确实学习到了对应技能的经验，展示了本文的方法在性能迁移方面的潜力。

虽然本文在此场景下取得了一些研究成果，但在研究过程中仍有许多难题值得进行进一步的探索。首先，多任务的场景中，不可避免的各个任务中可能存在一些相互冲突的部分，如何消解这些冲突是一个很有价值的挑战。其次，本文简单的介绍了在迁移方面的一些技术表现，当一个多任务智能体训练完成之后，如何可以使用这个智能体来加速新任务的训练也是值得研究的方向。最后，本文所用的核心强化学习算法并未过多的使用一些强化技巧，例如经验回放等。后续可以考虑使用这些技巧来强化智能体的性能。

参考文献

[1] Russell S J, Norvig P. Artificial Intelligence - a Modern Approach, 2nd Edition[M]. 2003.

[2] Rhalibi A E, Wong K W, Price M. Artificial Intelligence for Computer Games[J]. International Journal of Computer Games Technology, 2009 : 1-3.

[3] Billings D, Davidson A, Schaeffer J, et al. The Challenge of Poker[J]. Artificial Intelligence, 2002 : 201-240.

[4] Sutton R S. Learning to Predict by the Methods of Temporal Differences[J]. Machine Learning, 1988 : 9-44.

[5] Watkins C J, Dayan P. Q-learning[J]. Machine Learning, 1992 : 279-292.

[6] Tesauro G. TD-Gammon, a Self-Teaching Backgammon Program, Achieves Master-Level Play[J]. Neural Computation, 1994 : 215-219.

[7] Mnih V, Kavukcuoglu K, Silver D, et al. Playing Atari with Deep Reinforcement Learning[C] // Proceedings of Neural Information Processing Systems Deep Learning Workshops. 2013 : 1-9.

[8] Mnih V, Kavukcuoglu K, Silver D, et al. Human-Level Control through Deep Reinforcement Learning[J]. Nature, 2015 : 529-533.

[9] Silver D, Huang A, Maddison C J, et al. Mastering the Game of Go with Deep Neural Networks and Tree Search[J]. Nature, 2016 : 484-489.

[10] Vandenhende S, Georgoulis S, Proesmans M, et al. Revisiting Multi-Task Learning in the Deep Learning Era[J]. CoRR, 2020.

[11] 李亚. 多任务学习的研究[D] : 中国科学技术大学, 2018.

[12] Kendall A, Gal Y, Cipolla R. Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics[C] // Procedings of the International Conference on Computer Vision and Pattern Recognition. 2018 : 7482-7491.

[13] Chen Z, Badrinarayanan V, Lee C, et al. GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks[C] // Proceedings of the International Conference on Machine Learning. 2018 : 793-802.

[14] Sener O, Koltun V. Multi-Task Learning as Multi-Objective Optimization[C] // Advances in Neural Information Processing Systems. 2018 : 525-536.

[15] Misra I, Shrivastava A, Gupta A, et al. Cross-Stitch Networks for Multi-task Learning[C] // Proceedings of the International Conference on Computer Vision and Pattern Recognition. 2016 : 3994-4003.

[16] Ruder S, Bingel J, Augenstein I, et al. Latent Multi-Task Architecture Learning[C] // Proceedings of the AAAI Conference on Artificial Intelligence. 2019 : 4822-4829.

[17] Lu Y, Kumar A, Zhai S, et al. Fully-Adaptive Feature Sharing in Multi-Task Networks with Applications in Person Attribute Classification[C] // Proceedings of the International Conference on Computer Vision and Pattern Recognition. 2017 : 1131-1140.

[18] Wilson A, Fern A, Ray S, et al. Multi-Task Reinforcement Learning: a Hierarchical Bayesian Approach[C] // Proceedings of the International Conference on Machine Learning. 2007 : 1015-1022.

[19] Li H, Liao X, Carin L. Multi-task Reinforcement Learning in Partially Observable Stochastic Environments[J]. Journal of Machine Learning Research, 2009 : 1131-1186.

[20] Lazaric A, Ghavamzadeh M. Bayesian Multi-Task Reinforcement Learning[C] // Proceedings of the International Conference on Machine Learning. 2010 : 599-606.

[21] Calandriello D, Lazaric A, Restelli M. Sparse Multi-Task Reinforcement Learning[C] // Advances in Neural Information Processing Systems. 2014 : 819-827.

[22] Parisotto E, Ba L J, Salakhutdinov R. Actor-Mimic: Deep Multitask and Transfer Reinforcement Learning[C] // Proceedings of the International Conference on Learning Representations. 2016 : 1-16.

[23] Andreas J, Klein D, Levine S. Modular Multitask Reinforcement Learning with Policy Sketches[C] // Proceedings of the International Conference on Machine Learning. 2017 : 166-175.

[24] Omidshafiei S, Pazis J, Amato C, et al. Deep Decentralized Multi-task Multi-Agent Reinforcement Learning under Partial Observability[C] // Proceedings of the International Conference on Machine Learning. 2017 : 2681-2690.

[25] Saxe A M, Earle A C, Rosman B. Hierarchy Through Composition with Multitask LMDPs[C] // Proceedings of the International Conference on Machine Learning. 2017 : 3017-3026.

[26] Mirowski P, Pascanu R, Viola F, et al. Learning to Navigate in Complex Environments[C] // Proceedings of the International Conference on Learning Representations. 2017 : 1-16.

[27] Teh Y W, Bapst V, Czarnecki W M, et al. Distral: Robust Multitask Reinforcement Learning[C] // Advances in Neural Information Processing Systems. 2017 : 4496-4506.

[28] Hinton G E, Vinyals O, Dean J. Distilling the Knowledge in a Neural Network[C] // Proceedings of Neural Information Processing Systems Deep Learning Workshops. 2014 : 1-9.

[29] Rusu A A, Colmenarejo S G, Gülçehre Ç, et al. Policy Distillation[C] // Proceedings of the International Conference on Learning Representations. 2016 : 1-13.

[30] Czarnecki W M, Pascanu R, Osindero S, et al. Distilling Policy Distillation[C] // Proceedings of the International Conference on Artificial Intelligence and Statistics. 2019 : 1331-1340.

[31] Lai K, Zha D, Li Y, et al. Dual Policy Distillation[C] // Proceedings of the International Joint Conference on Artificial Intelligence. 2020 : 3146-3152.

[32] Jhunjhunwala A, Lee J, Sedwards S, et al. Improved Policy Extraction via Online Q-Value Distillation[C] // Proceedings of the International Joint Conference on Neural Networks. 2020 : 1-8.

[33] Wadhwania S, Kim D, Omidshafiei S, et al. Policy Distillation and Value Matching in Multiagent Reinforcement Learning[C] // Proceedings of the International Conference on Intelligent Robots and Systems. 2019 : 8193-8200.

[34] Sutton R S, Barto A G. Reinforcement Learning: An Introduction[M]. 2018.

[35] Williams R J. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning[J]. Machine Learning, 1992 : 229-256.

[36] Schulman J, Levine S, Abbeel P, et al. Trust Region Policy Optimization[C] // Proceedings of the International Conference on Machine Learning. 2015 : 1889-1897.

[37] Schulman J, Wolski F, Dhariwal P, et al. Proximal Policy Optimization Algorithms[J]. CoRR, 2017.

[38] Narasimhan K, Kulkarni T D, Barzilay R. Language Understanding for Text-based Games using Deep Reinforcement Learning[C] // Proceedings of the Conference on Empirical Methods in Natural Language Processing. 2015 : 1-11.

[39] Hausknecht M J, Stone P. Deep Recurrent Q-Learning for Partially Observable MDPs[C] // Proceedings of Association for the Advancement of Artificial Intelligence Fall Symposia. 2015 : 29-37.

[40] Schaul T, Quan J, Antonoglou I, et al. Prioritized Experience Replay[C] // Proceedings of the International Conference on Learning Representations. 2016 : 1-21.

[41] 王庭晗, 罗禹贡, 李克强, 等. 基于考虑状态分布的深度确定性策略梯度算法的端到端自动驾驶策略[J]. 清华大学学报 (自然科学版), 2020 : 1-8.

[42] van Hasselt H, Guez A, Silver D. Deep Reinforcement Learning with Double Q-Learning[C] // Proceedings of the AAAI Conference on Artificial Intelligence. 2016 : 2094-2100.

[43] Mnih V, Badia A P, Mirza M, et al. Asynchronous Methods for Deep Reinforcement Learning[C] // Proceedings of the International Conference on Machine Learning. 2016 : 1928-1937.

[44] Munos R, Stepleton T, Harutyunyan A, et al. Safe and Efficient Off-policy Reinforcement Learning[C] // Advances in Neural Information Processing Systems. 2016 : 1054-1062.

[45] Schulman J, Moritz P, Levine S, et al. High-Dimensional Continuous Control Using Generalized Advantage Estimation[C] // Proceedings of Conference Track in International Conference on Learning Representations. 2016 : 1-14.

[46] Perez E, Strub F, de Vries H, et al. FiLM: Visual Reasoning with a General Conditioning Layer[C] // Proceedings of the AAAI Conference on Artificial Intelligence. 2018 : 3942-3951.

[47] Shelhamer E, Long J, Darrell T. Fully Convolutional Networks for Semantic Segmentation[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017 : 640-651.

[48] Wang Z, Schaul T, Hessel M, et al. Dueling Network Architectures for Deep Reinforcement Learning[C] // Proceedings of the International Conference on Machine Learning. 2016 : 1995-2003.

[49] Espeholt L, Soyer H, Munos R, et al. IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures[C] // Proceedings of the International Conference on Machine Learning. 2018 : 1406-1415.

攻读硕士学位期间发表的论文及其他成果

(一) 发表的学术论文

[1] **Zhenyang Guo**, Xuan Wang, Shuhan Qi, Tao Qian, Jiajia Zhang, Heuristic Sensing: An Uncertainty Exploration Method in Imperfect Information Games, Complexity, vol. 2020, Article ID 8815770, 2020. https://doi.org/10.1155/2020/8815770 (JCR-2区，已发表)

(二) 参与的科研项目及获奖情况

[1] 复杂环境下的博弈决策技术研究。重点实验室基金一般项目，项目编号6142110190410。

[2] NIPS2019 侦查盲棋比赛综合成绩第 7，不确定性控制单项第 1。

[3] 2020 年中央军委科技委六人德州扑克挑战赛第一名。

致　谢

转眼间在哈工深的研究生生涯就要结束了，我也即将开启新的篇章。在哈工深的两年半的时间里，作为实验室的一份子，我也担任了一些职位，忙碌的同时也收获了很多。老师同学家人朋友的帮助与关心，让我的这段研究生经历变得充实和宝贵，在此我向他们表达我真挚的谢意。

在研究生的生涯里，首先要感谢的是实验室的老师们，作为领路人，指导带领我进入了我如今的研究领域。首先是我的导师王轩教授，感谢王老师对我的教导，在科研上王老师指导方向和学习方法，带领我们进入博弈这一领域，生活上王老师鼓励我们锻炼身体有一个好的体魄，经常和我们交流了解我们的课题，求职进展，给我们提供了很多的帮助。我还要感谢漆舒汉老师和张加佳老师，漆老师在日常生活，课题论文以及项目申请各个方面都给予了我很大的帮助，还组织我们开展每周的学习交流和定期的课题进展汇报，为我的课题开展提出了很多宝贵的意见。我的比赛的完成，论文的发表，以及参与项目的过程中都离不开张加佳老师的帮助和督促，同时他还给我的毕设论文提供了很多建议，细心修改。在两位实验室的师兄的帮助和鼓励下，我得以顺利的完成我参与的项目课题，从中收获颇多。

此外，我还要感谢胡甄博、杜明欣、吴卓、郭颐冰、范茂顺、黄泽涛等各位18级的同学们，和各位同窗一起度过了研究生生涯，大家一起共同学习，共同进步，感谢他们在日常生活和学习上的帮助。最后，感谢我的父母的无微不至的关心，疫情期间和父母一起的日子里，以及在课题遇到难题、求职不顺利的时候父母的叮嘱和开解都让我感到家的温暖，他们是我最坚强的后盾。