**广东第二师范学院计算机学院**

**课程设计报告**

**专业名称:**数据科学与大数据技术

**课程名称:**人工智能课程设计

**提交日期: 2022 年 6 月 13 日**

**任课教师: 张瑞琳**

**学生班级: 20大数据**

**学生姓名: 林怡**

**学生学号: 20551120028**

**课程成绩：**

**温馨提示：**

（1）注意课程设计报告的提交日期，若无正当理由，逾期不交者，任课教师有权不予接受或批阅其课程设计报告。若有特殊原因，不能按时提交课程设计报告，必须提交书面申请，经任课教师同意，方可缓交课程设计报告。

（2）确保留有充裕的时间完成自己所承担的工作。个人须约束自己的言行，有效地履行自己的职责。

（3）若课程设计报告中参考了他人的工作成果，必须在参考文献中予以标注。

（4）一旦发现课程设计报告中存在抄袭或剽窃行为，视情节轻重，当事人将会受到课程成绩记零分或考试作弊的处罚。

**学生声明:**

我确认课程设计报告是我自己做的。

学生签名: 林怡

附录B：课程设计小组行动计划

|  |  |  |  |
| --- | --- | --- | --- |
| 起止日期 | 工作任务 | 完成人 | 工作结果 |
| 1-1 | 收集资料 | 林怡 | 已完成 |
| 1-3 | 跑通代码 | 林怡 | 已完成 |
| 3-4 | 分析实验结果 | 林怡 | 已完成 |
| 4-6 | 撰写实验报告 | 林怡 | 已完成 |

其中起止日期采用相对日期，不能采用绝对日期；工作任务的描述采用“动词+宾语”结构；完成人须指明小组成员中的某个学生；工作结果是完成人在做完所承担的工作后，需提交的显性成果。比如：

|  |  |  |  |
| --- | --- | --- | --- |
| 起止日期 | 工作任务 | 完成人 | 工作结果 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

说明：小组行动计划必须为小组中的每个成员至少分配一项任务。

附录C: 课程设计报告格式

Ⅰ目录

目录的内容主要包括引言、正文主体、参考文献、附录等的序号、名称和页码组成。目录内容应简明扼要。

目录的编排格式如下：

①“目录”二字用小三号字、黑体、居中书写，“目”与“录”之间空两格。上空一行，下空二行

②第一级标题居中放置；第二、三、四等级标题序数顶格放置，后空一格接标题内容，末尾不加标点。标题序数采用1.、2.……1.1、1.2……1.1.1、1.1.2……1.1.1.1……的层次。

③目录中第一级标题用小四号字、黑体，其它标题用小四号宋体，顶格。第二级标题空1格。第三级标题空2格。各级标题中的阿拉伯数字用Times New Roman字体。

Ⅱ正文

正文是课程设计的核心部分，主要陈述个人所负责任务的实施过程及结果，占主要篇幅，一般由标题、文字段落、图、表和公式五个部分构成。

在正文的第一部分（引言）中，应说明课程设计内容。

正文的最后一部分（总结）中，应总结自己在课程设计工作中的经验与教训。

正文的编排格式如下：

①标题和文字段落

“引言”二字用4号字、黑体、居中书写，“引”与“言”之间空两格。引言正文采用5号宋体，行间距为固定值18磅。

正文中各级打印如下：

1（空一格）×××（居中、4号、黑体、前、段后间距各0.5行）

1.1×××（4号、黑体、顶格；段前、段后间距各0.5行）

1.1.1×××（4四号、宋体、加粗、顶格；段前、段后间距各0.5行）

各级标题中的阿拉伯数字用Times New Roman字体。

正文（5号、宋体）

②图

每个图必须有图名和图号。图号按章顺序编号，例如，图3.2：算法流程图，意为第三章第二图。应将分图号和图名居中标注在图的正下方。

绘图必须工整、清晰、规范。示意图应能清楚反映图示内容；对于照片，如有必要，应在右下角给出放大标尺；试验结果曲线图应制成方框图。

应将图尽可能地置于某页的开始或结尾，并且在图之前的文字段落中有“如图x.x”的字样。

③表

表的编排，一般是内容和测试项目由左至右横读，数据按内容或项目依序对齐竖排。表中只用长度相同、粗细适中的横线（居中），禁止使用竖线。

表按章顺序编号，如表5.4为第五章第四表。编号的后面空一格，居中打印表的简短确切的标题，且置于表的上方。在中文表标题和表之间，居中打印英文表标题，如“Table 5.4 xxxxxxxx”。标题分别采用五号楷体、Times New Roman。

表内的缩写词和符号等，必须与文字段落中的一致。各符号的单位应尽可能标出。表内不宜用“同上”、“同左”等类似词，一律填入数字或文字。表内空白处代表无此项内容。

应将表尽可能置于某页的开始或结尾，并且在表之前的文字段落中有“如表x.x”的字样。

④数学公式

数学在正文中另起一行打印。公式的序号按章顺序编排，如(4.2)表示第四章的第二式。

较长的公式，另行居中横排。如果式必须转行，只能在＋、－、×、÷、<、>处转行，上下式尽可能在等号“=”处对齐。

Ⅲ参考文献

“参考文献”四字采用四号黑体字，居中。其编排格式如下：

参考文献中序号用中扩号，与文字之间空一格。中文的用五号宋体，外文的用五号Times New Roman字体，行间距为18磅。参考文献的序号左顶格，用数字加方括号表示，如［1］，［2］，…，应与正文中的指示序号格式一致。每一参考文献条目的最后均以“．”结束。作者姓名写到第三位，余者写“等”，引用文献的作者姓名为汉语拼音的书写方式时，作者姓名应按照先姓后名的顺序排列。

①专著、论文集、学位论文、报告

[序号]（空一格）主要责任者.文献题名[文献类型标识].出版地：出版者，出版年.起止页码.比如：

[1]刘国钩，陈绍业，王凤翥等.图书馆目录[M].北京：高等教育出版社，1957.15-18.

②期刊文章

[序号]主要责任者.文献题名[J].刊名，年，卷(期)：起止页码.比如：

[2]王清毅，蔡智，邹翔，蔡庆生. 部分数据缺失环境下的知识发现方法[J]. 软件学报，2001,12(10)：1516-1524

[3]张颖，刘艳秋.软计算方法[M]. 北京：科学出版社，2002，5

③报纸文章

[序号]主要责任者.文献题名[N].报纸名，出版日期(版次).比如：

[4]谢希德.创造学习的新思路[N].人民日报，1998-12-25(10).

④国际、国家标准

[序号]标准编号，标准名称[S].比如：

[5]GB/T 16159-1996，汉语拼音正词法基本规则[S].

⑤专利

[序号]专利申请者.专利题名[P].专利国别：专利号，出版日期.比如：

[6]姜锅洲.一种温热外敷药制备方案[P].中国专利：881056073，1989-07-26.

⑥电子文献

[序号]主要责任者.电子文献题名[电子文献及载体类型标识].电子文献的出处或可获得地址，发表或更新日期/引用日期(任选).比如：

[7]王明亮.关于中国学术期刊标准化数据库系统工程的进展[EB/OL].http：//www.Cajcd.edu.cn/pub/wml.tex/980810-2.html,1998-08-16/1998-10-04.

[8]万锦节.中国大学学报论文文摘(1983-1993).英文版[DB/CD].北京：中国大百科全书出版社，1996.

⑦各种未定义类型的文献

[序号]主要责任者.文献题名[Z].出版地：出版者，出版年.比如：

[9]张永诛.唐代长安词典[Z].西安：陕西人民出版社，1980.

⑧引用参考文献类型及其标识说明

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 参考文献类型 | 专著 | 论文集 | 单篇论文 | 报纸文章 | 学位论文 | 期刊文章 | 报告 | 标准 | 专利 | 其它文献 | 数据库  （网上） | 计算机程序  （磁盘） | 光盘  图书 |
| 文献类型标识 | [M] | C | (A) | N | D | J | R | S | P | Z | DB(EB/OL) | CP(CP/DK) | M/CD |

Ⅳ附录

附录是作为课程设计主体的补充部分，主要是程序源代码，需要注意的是必须为有关的语句给予必要的注释。附录用“附录A”，“附录B”，“附录C”等编号。如果只有一个附录，也写成“附录A”。采用5号黑体、居首编排“附录A：附录名称”。如：

附录A：查询源代码

在附录题名下空一行，采用Times New Roman,10磅编排程序源代码。

目录

[一、引言 5](#_Toc29376)

[二、介绍 5](#_Toc4183)

[三、背景 6](#_Toc21130)

[四、相关工作 6](#_Toc27644)

[五、实验 7](#_Toc11449)

[5.1 DQN算法简述 7](#_Toc31672)

[5.2 预处理和模型架构 12](#_Toc29130)

[5.3 卷积神经网络 15](#_Toc25176)

[5.4 体验回放 16](#_Toc25742)

[5.5 DDQN改进 17](#_Toc15803)

[5.6 实验结果可视化 18](#_Toc16503)

[5.7 实验评估 23](#_Toc17519)

[六、结论 23](#_Toc12079)

[七、引用文献和资料 24](#_Toc31078)

# 使用DQN算法玩Atari游戏

## 一、引言

**DeepMind**提出了第一个成功使用强化学习直接从高维感官输入中学习控制策略的深度学习模型。它采用**DQN**算法进行训练，其输入是原始像素，输出是估计未来奖励的值函数。我使用了该算法及其变体来实现机器在不需要为某一游戏特意调整架构和学习算法的基础上进行多款**Atari2600**游戏的自我学习,并且将训练效果可视化,和人类玩家做比较以展现该算法的一些成就和可以改进之处。

## 二、介绍

机器学习的两个重要分支是神经学习和强化学习,这两个分支致力于解决不同类型的问题. 深度学习主要解决机器的认知问题,通过构建多层神经元提升神经网络数据拟合的潜力,之后通过样本训练不断修正神经元参数提升数据拟合准确度,即提升对事物的认知和识别能力. 当下流行的语音识别就应用到了RNN深度神经网络,图像识别应用到了CNN深度神经网络. 强化学习主要解决MDP问题,致力于在复杂场景做出动态决策的能力. 二者相互结合可以实现更多强大的功能.让机器通过不断玩Atari游戏来提升分数来接近甚至超过人类玩家得分是一件相当有挑战的事情,但是现在机器学习的发展完全可以支撑我们去做这件事情. 将强化学习的Q-Learning算法中的Q-table替换为CNN来实现以实现二者结合可以达到一个理想的效果.

学习直接从高维感觉输入(如视觉和语言)控制代理是强化学习(RL)的长期挑战之一。在这些领域上运行的大多数成功的RL应用程序都依赖于人工分析环境的一些特征要素,这类系统的性能在很大程度上依赖于特征表达的质量。

深度学习的最新进展使得从原始感觉数据中提取高级特征成为可能，导致了计算机视觉和语音识别的突破。这些方法利用一系列神经网络结构，包括卷积网络，多层感知器、受限Boltzmann机器和递归神经网络，并利用了监督和非监督学习。 人们似乎很自然地会问，类似的技术是否也会对拥有感官数据的RL有利。从深度学习的角度来看，强化学习也带来了一些挑战。很多成功的深度学习应用都需要大量的人工标记的训练数据。另一方面，RL算法必须能够从标量奖励中学习。信号经常是稀疏的、嘈杂的和延迟的。行动和结果奖励之间的延迟，它可能有数千个时间步长，与直接的时间步长相比，似乎特别令人望而生畏。在监督学习中发现的输入和目标之间的关联。 另一个问题是最深层次的。学习算法假设数据样本是独立的，而强化学习算法假设数据样本是独立的。通常会遇到高度相关的状态序列。 此外，在RL中，随着算法学习新的行为，数据分布也会改变，这对于深度学习可能是有问题的。假定基础分布固定的方法。

## 三、背景

深度强化学习的成功案例不算特别多,但是每个都太有名了,比如AlphaGo通过左右互搏等方式在围棋上碾压人类;大大降低Google能源中心耗能. Atari是美国诺兰·布什内尔在1972年成立的电脑公司,街机、家用电子游戏机和家用电脑的早期拓荒者。不少诸如《乓》、《爆破彗星》等的经典早期电脑游戏的发行，使雅达利在电子游戏历史上举足轻重。经典游戏主机为1977年发行的雅达利2600.

未来无人驾驶的发展也十分依赖深度学习和强化学习的综合应用,对DRL研究的推进将对人类社会产生深远影响,这些技术毫无疑问像电力一样会渗透到我们生活中的方方面面.

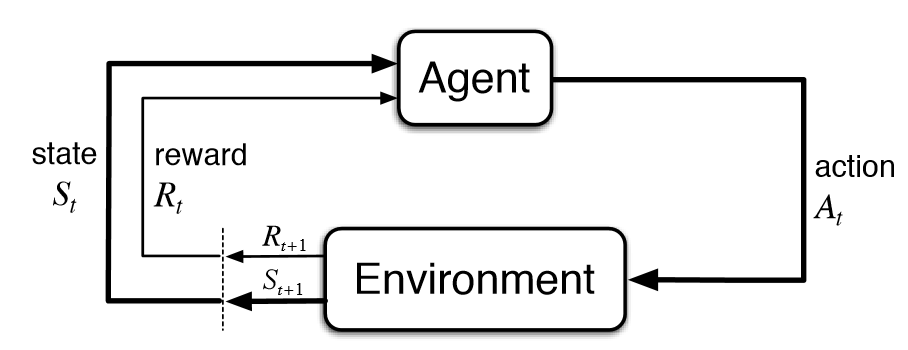
## 四、相关工作

在DRL领域,DeepMind和OpenAI等组织或实验室做的比较出色.(to do) OpenAI创立的gym项目是一个可以模拟大量强化学习/游戏环境的库.它为DRL提供了丰富的测试环境,其中就包含了多种Atari游戏环境,如CartPole,Breakout,SpaceInvaders. 而DeepMind 在2013年首次提出了DQN算法及其应用-----让机器自己学习以提升在各种复杂环境中的表现,是DRL领域的先驱.战胜诸多围棋棋手的享誉全球的AlphaGo就出自DeepMind之手,DRL也因此更加引发人们的关注.

## 五、实验

### 5.1 DQN算法简述

强化学习的本质是给定一个马尔科夫决策过程MDP,寻找最优策略实现奖励最大化.



强化学习系统的主要要素包括智能体,环境,策略,即时奖励信号,值函数和可选的奖励模型.

上图形象地描述了MDP的大致流程:智能体在t时刻做出某个决策使得环境状态转变到状态St时收获的即时奖励是Rt,之后采取决策At,使得环境状态转变到了St+1,在这个状态转变的过程中收获的即时奖励是Rt+1.这是一个循环迭代的过程.

理解题设以后,接下来首先我们要明确我们的目标,然后找出方案去实现它.

我们的目标是最大化总奖励Gt,也就是最大化下面的式子

下面公式展示了处于状态s时有限步决策后续总收益.

但是现实情况中我们会遇到很多问题都会持续不断地无限制进行,比如使用寿命长的机器人的应用程序,如果仍然按照有限步决策的思路,寻找下面公式的最大值是一件不可能完成的事情,因为这个求和序列是发散的.

因此要是想将上面发散问题转变成可收敛问题,需要引入的额外概念是折扣因子γ

折扣因子要求0<=γ<1,依据之一是在这种情况下等比数列之和收敛.

折扣率是未来奖励对现在决策的影响权重的体现.

关于研究目标还有一点要说明的是最优解有时候是完全找不到的,典型地,引入了随机性(在状态s下采取了行为a后进入的下一个状态不唯一)的决策问题大多都没法找到一个确定的最优解;此外还有一些理论上可以找到最优解的情景,但是通过计算获得最优解的计算成本过于昂贵以至于无法承受.

鉴于上述两种情况,很多时候找到一个近似解就已经是很理想的结果.

明确了我们的研究目标以后,接下来我们就要思考如何实现目标了(换句话就是制定策略).

策略直观上很好理解,就是指导智能体在环境处于某个状态时应当采取的行为.

数学上理解就是一个从环境St到行为A­t的映射,At=π(St),但关键是如何找到这个映射的具体数学表达式呢?

为了找到最优解对应的最优策略或者相对理想的近似解对应的近似策略,我们需要引入一个概念 **状态-动作值函数**作为策略的依据. **状态-动作值函数**是一种相对抽象的非即时评价体系,它用来估计在st状态下采取动作at时未来可以收获的最大奖励,我们把这种预估值称为**价值.** 我们强化学习的目标就是最大化奖励值,**围绕未来奖励值上下波动的状态-动作值函数q(at,st)**正适合作为制定策略的依据之一----**利用:状态st有行为空间A(St),采取使得q(at,st)能取取最大值的at**.

值函数的思想很好理解, 假设St可选行为的集合为A(St),那么遍历A(St)的所有行为,对每个行为a∈A(St)进行价值评估,记为q(St,a).

然后比较所有q(St,a)找到最大值对应的a即可.

这种想法很好理解但是要想抽象成具体的数学表达式有一些问题亟待解决:

问题1:强化学习的挑战之一就是当前行为不仅仅影响即时奖励和接下来的状态(**即时奖励**指的是在s­t采取at时收获的奖励值,同一状态采取不同的行为除了导致获得的即时奖励r­t不同,还会导致下一时间序列的状态st+1不同. 状态也分好坏,不同的状态具有不同的得分潜力,这是很自然的一件事情.但是这种得分潜力并不会即时反馈给智能体,不像即时奖励一样所见即所得),还会对后续行为产生难以预估的影响,就是典型的"蝴蝶效应"(蝴蝶效应是指在一个动态系统中，初始条件的微小变化，将能带动整个系统长期且巨大的链式反应，是一种混沌的现象。“蝴蝶效应”在混沌学中也常出现 )!如何保证q(St,a)的表达式能够合理地包含未来奖励和现下行为a的关系?

解决方案:表达式中必然包含R­t,Rt+1,Rt+2等因子.

客观存在的"蝴蝶效应"让我们不能忽略现在行为和未来奖励的关联,同时我们关注的最优化问题应该是一个收敛问题&&直觉和经验上来看离当前行为时间尺度上越近的未来奖励反馈在其它条件一致的情况下包含更多的当前行为的价值信息.

我们还要复用之前折扣因子的概念.

一个状态下某个行为的实际后续奖励依赖我们后续的行为决策.后续行为决策不同,该实际未来奖励也不同.

因此需要引入数学期望

综上所述,

问题2:如何获取上式的数据Rt+k+1进行计算?因为Rt+k+1这些数据是做出未来决策以后才能获取的数据,未来如何做决策仅仅按照前面讲的"依据q(St,a)最大值对应的a"完全不够充分,因为训练较少的情况下可能进入一个agent之前一直没有进入过的状态,极端情况下,初始状态下agent所有状态都未曾进入过,也就意味着对于A(St)中的a, q(St,a)反映的价值都一样.比如在某个新状态下的行为空间大小为3,可以采取的行为包括a0,a1,a2,因为是新状态,所以q­π(st,a0)= q­π(st,a1)= q­π(st,a2).如果随机选择了一个决策行为比如a0,在后续过程q­π(st,a0)更新以后增大了,这会使得q­π(st,a0)开始比qπ(st,a1)和qπ(st,a2)都要大.

从此以后a1和a2再也没有被选择的机会,一切竟然只是因为之前的随机选择.而事实上a1和a2很可能是更优选. 因此为了防止明珠蒙尘,有必要不能一味地根据来选择行为,而要引入随机机制来保障每个行为都有被选择的可能性,这就是探索机制.

探索机制使智能体不断探索,在训练的过程中可以找到更优解,有助于agent提升智能.

利用机制则是发挥agent的已有智能,在实际应用场合发挥至关重要的作用.

总而言之,值函数终究是一种近似和预估最大奖励值的手段,而不是最大奖励值本身,通过值函数获得的对环境和状态的认知就有局限性,容易获得局部步长最优解,哪怕引入动态变化的折扣因子γ也无济于事.**为了追求全局最优解,必须引入探索机制**,与利用机制交互发挥作用!

问题3:探索机制和利用机制冲突,如何解决兼容?

这个比较好办,设置一个比例参数使得两种机制按照一定比例交替发挥作用即可!

上面问题都解决以后,回到究竟如何制定策略的话题上来------将策略数学化表达出来???

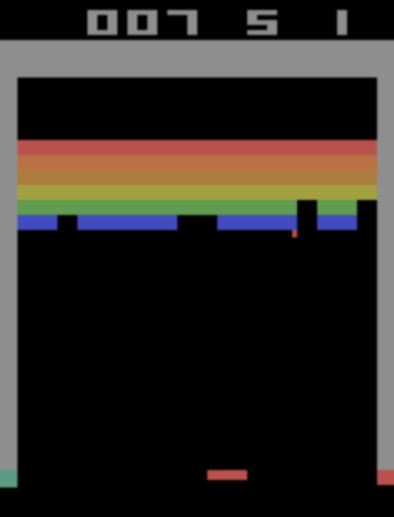
策略的形成依赖两种机制:探索和利用.这两种机制交互进行,尤其是探索机制使得同一状态下可能采取不同的行为,因此策略π是随机的,它指代的映射关系at=π(st)大部分时候不是一输入对一输出的,而是一输入对应多个可能的输出,这些输出按照概率分布的规律进行取值!用数学形式表达的话,π(at|st)=p(at|st),依照我的理解:每个状态st下的策略是一个概率向量,策略整体是一个状态-行为 值矩阵.

以上就是DQN算法的基本思想,接下来按照这些公式结合具体的Atari问题就可以去实现机器对游戏的自动学习了.

### 5.2 预处理和模型架构

为了感受DRL的自适应学习,不会采取对某一游戏的特殊处理,而是统一采取所有Atari游戏的共同点:像素. 接下来以BreakOut为例来说明这一点.

暂停一个游戏,看看我们可以得到什么?

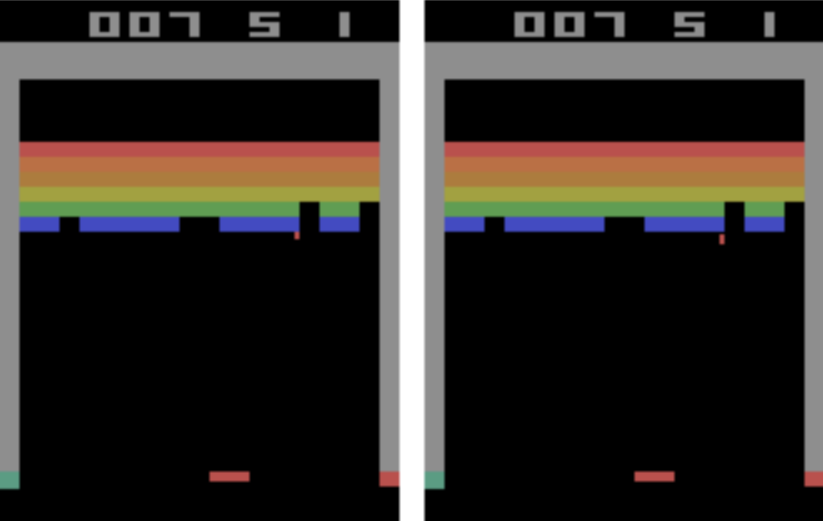


我们可以看到球拍的位置，我们可以看到球和积木。

但是我们是否拥有所有的信息来确定到底发生了什么？

不，我们没有任何关于游戏物体运动的信息，然而这在游戏世界中是至关重要的。

我们怎样才能克服这个问题呢？需要为每个观测再叠加一个连续的帧构成两帧。

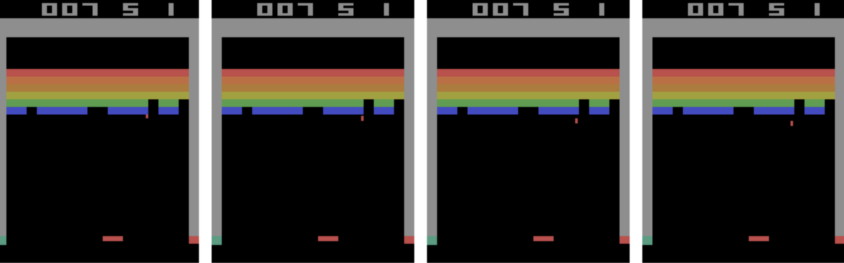


现在我们可以看到球的方向和速度。

但是我们知道它的加速度吗？ 不，那我们叠三张吧。



现在我们可以推导出运动物体的方向、速度和加速度，但由于不是每个游戏都以同样的速度进行渲染，我们将保留4帧——只是为了确保我们有所有必要的信息。

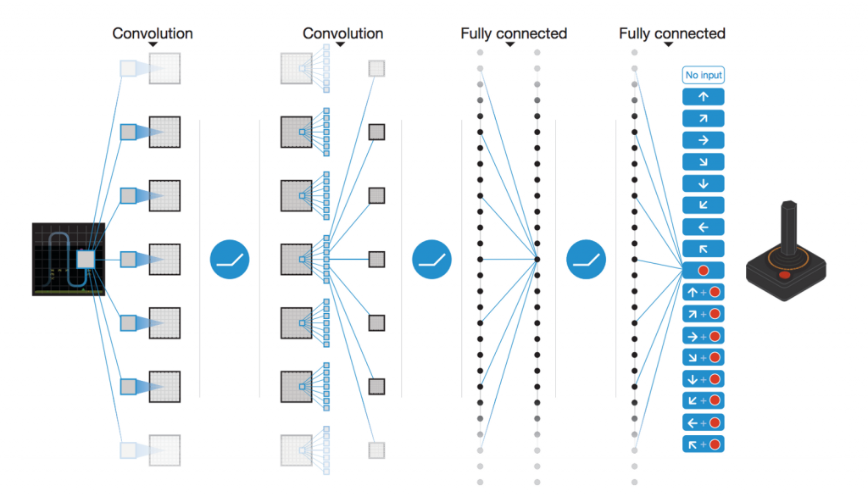


以上定义了连续的4帧作为一个状态来表达所有必要的信息，现在让就可以把精力都集中在图像的预处理上了。Atari 环境输出210x160 RGB 数组(210x160x3)。 这样的像素太多了，绝对超出了实验的需要。 首先，让我们裁剪和缩减我们的图像到84x84方块(这在程序中可以很容易按照自己的想法修改)。 这种处理方式使得我们将使用更少的内存，同时仍然保留所有必要的信息。 然后可以将颜色转换为灰度，我们将观察矢量从84x84x3减少到84x84x1。 假设我们要存储最后4个帧，那么我们的输入形状将是84x84x4。 请注意，每一帧都是由4个连续帧构成状态整体的一部分，因此我们可以使用内存中单一点的引用，而不是每次都复制它。我们将显著减少我们的内存占用(可达4倍)，这对后续进行体验回放时至关重要，我们将在 RAM 中存储多达100万个观察结果。 以上所有的优化都可以手动实现，但是不需要重造轮子，可以随意使用 OpenAI 的包装器，这些包装器经过测试并证明是有效的。

通过上述预处理过程,大大缩减了计算量. 将RGB图像进行灰度化处理和减采样处理为更少像素的灰度图像.对程序运行效率有显著提升.

### 5.3 卷积神经网络

传统RL算法使用Q\_table来记录状态值,DRL则使用CNN来代替Q-table,这也就是DRL之所以叫DRL的原因了.



图片来源：[https://medium.com/@awjuliani/simple-reinforcement-learning-with-tensorflow-part-4-deep-q-networks-and-beyond-8438a3e2b8df](https://medium.com/@awjuliani/simple-reinforcement-learning-with-tensorflow-part-4-deep-q-networks-and-beyond-8438a3e2b8df" \t "_blank)

可是为什么非要用NN替代Q-table呢?

这是因为在普通的Q-Learning中，当状态和动作空间是离散且维数不高的时候可以使用Q-Table来存储每个状态动作对应的Q值，而当状态和动作空间是高维连续时，使用Q-Table不现实。一是因为当问题复杂后状态太多，所需内存太大；二是在这么大的表格中查询对应的状态也是一件很耗时的事情。

通常的做法是把Q-Table的更新问题变成一个函数拟合问题，相近的状态得到相近的输出动作。如： Q(s,a;θ)≈Q′(s,a)，通过更新参数θ使Q函数逼近最优Q值，而深度神经网络可以自动提取复杂特征，因此，面对高维且连续的状态使用深度神经网络再合适不过。我们可以想象，神经网络接受外部的信息，相当于眼睛鼻子耳朵收集信息，然后通过大脑加工输出每种动作的值，最后通过强化学习的方式选择动作。

DRL是将深度学习（DL）和强化学习（RL）结合，直接从高维原始数据学习控制策略。而DQN是DRL的其中一种算法，它将卷积神经网络（CNN）和Q-Learning结合起来，CNN的输入是原始图像数据（作为状态state），输出则是每个动作Action对应的价值评估Value Function(Q值)。(或者输入状态和动作，通过神经网络输出对应的Q值)

深度学习与强化学习结合的问题:

1. 深度学习需要大量带标签的样本进行监督学习；强化学习只有reward的返回值，并且伴随着噪声，延迟（过了几十毫秒才返回0，稀疏（很多state的reward是0）等问题。
2. 深度学习的样本独立；强化学习前后的state状态相关。
3. 深度学习目标分布固定；强化学习的分布一直变化，比如你玩一个游戏，一个关卡和下一个关卡的状态分布是不同的，所以训练好了前一个关卡，下一个关卡又要重新训练。
4. 过往的研究表明，使用非线性网络表示值函数时会出现不稳定的问题。

三、DQN解决结合出现问题的办法

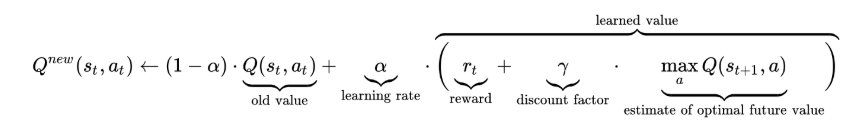
1. 通过Q-Learning使用reward来构造标签（对应问题1）
2. 通过experience replay(经验池）的方法来解决相关性及非静态分布问题（对应问题2,3）
3. 使用一个CNN（MainNet）产生Q值，使用另外一个CNN（Target）产生Target Q值（对应问题4）

我这里构建的神经网络的输入由预处理图生成的84 × 84 × 4的图像组成，第一隐藏层卷积32个8 × 8的滤波器，与输入图像步长为4，采用整流非线性。 第二隐藏层卷积了64个4 x 4的滤波器，跨度为2，其次是整流非线性滤波器。 接着是第三个卷积层，卷积64个3 x 3的滤波器，步幅1后跟一个整流器。 最后的隐藏层是完全连接，并由512整流单元组成。 输出层是一个完全连接的线性层，每个有效动作只有一个输出。

在我们考虑的游戏中，每个整体可以采取的有效动作的数量在4到18个之间。

### 5.4 体验回放

经验回放是一个受生物启发的过程，它统一地(减少后续行动之间的相关性)从记忆中采集经验，并且对于每个条目更新其 q 值。



对于每次运行，我们初始化新环境并设置其初始状态。

对于直到终止的每个步骤，基于给定的状态，我们从代理获得一个操作。 然后我们在一个环境中执行它，接收一个新的状态以及相关的奖励。然后，我们记住这些信息(状态，行动，奖励，状态下一步，终端)并执行经验回放。

这里将会占用大量的存储资源尤其是内存资源!这就是记忆的代价!

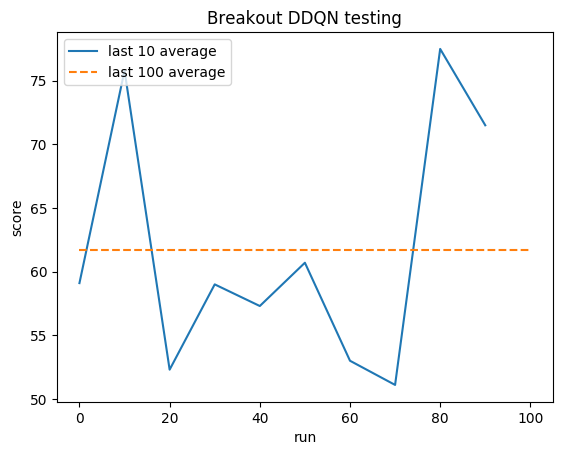
### 5.5 DDQN改进

常规 DQN 倾向于高估给定状态下潜在动作的 q 值。 如果所有的动作都被同样高估了，这不会造成任何问题，但是情况是，一旦一个特定的动作被高估了，它更有可能在下一次迭代中被选择，这使得代理很难统一地探索环境并找到正确的策略。

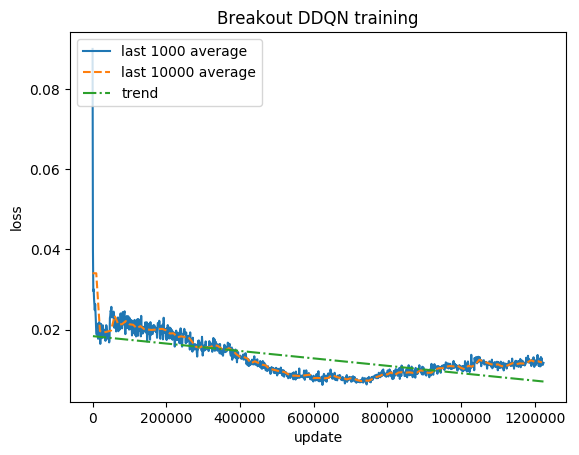
克服这个问题的方法很简单。 我们将动作选择与目标 q 值生成解耦。 为了做到这一点，我们将有两个独立的网络，因此名为双 q 学习。 我们将使用我们的主要网络来选择一个行动和一个目标网络生成一个 q 值的行动。 为了同步我们的网络，我们将从主网络复制权重到目标网络每 n (通常约10k)训练步骤。

Q-Target = r + γQ(s’,argmax(Q(s’,a,ϴ),ϴ’))

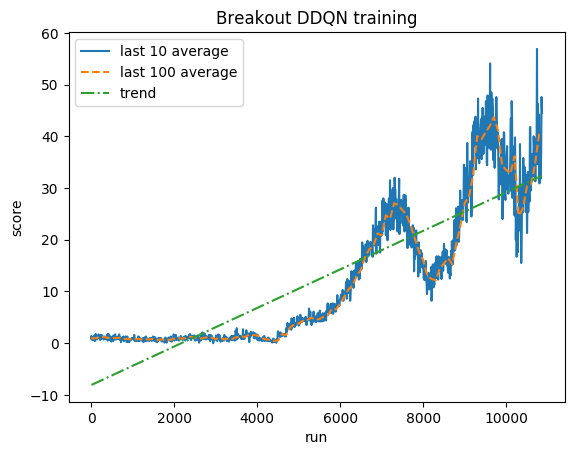
### 5.6 实验结果可视化



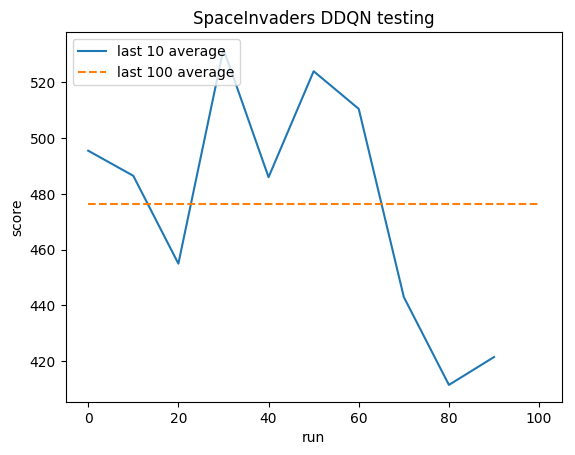
上图是agent训练完成后的测试结果.

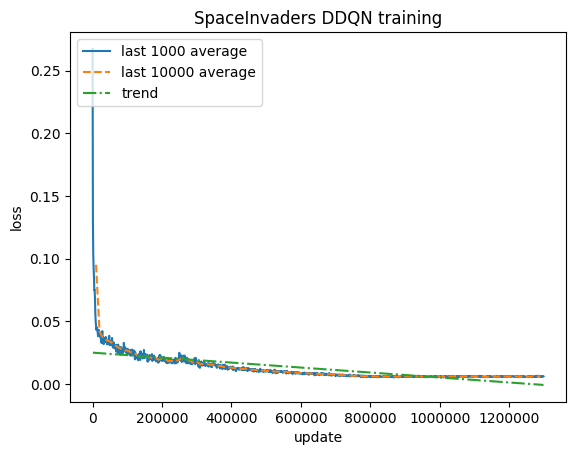


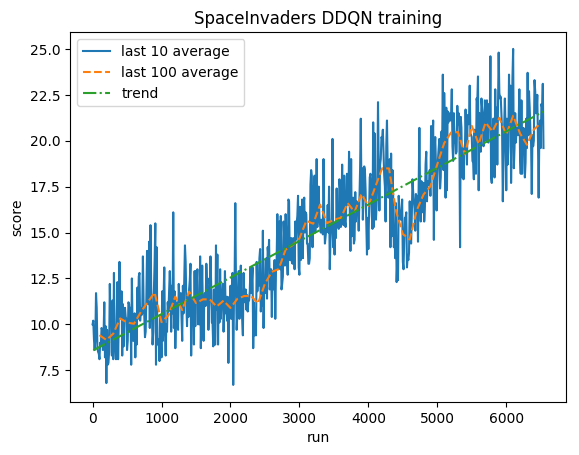
从以上图像可以看到损失函数在趋势上总体遵循随着训练的进行不断下降的规律!



从上图可以观察到智能体游戏得分在趋势上总体遵循随着训练的进行不断上升的规律!







### 5.7 实验评估

[file:///D:/code/Python\_WorkSpace/Deep-Reinforcement-Learning-with-Double-Q-learning/paper/1509.06461.pdf](file:///D:\\code\\Python_WorkSpace\\Deep-Reinforcement-Learning-with-Double-Q-learning\\paper\\1509.06461.pdf)

Breakout 的玩家得分平均27.9 DDQN平均得分62

SpaceInvaders 的玩家得分平均1464.9 DDQN平均得分479

## 六、结论

我只训练了两个游戏,在我的i5-8350U(四核八线程1.70GHz)上使用了两百小时以上,说明机器学习是一件极其占用计算资源的事情.

在这么长的时间里,Breakout性能表现出色,远超人类玩家平均水平,但是SpaceInvaders则不如人意.

DRL的缺陷还包括:

1. 它的样本利用率非常低。换言之为了让模型的表现达到一定高度需要极为大量的训练样本。

2.DRL成功的关键离不开一个好的奖励函数（reward function），然而这种奖励函数往往很难设计。

3.不稳定性,由于探索机制的存在,运气比较差的时候,不好的模型曲线将会维持很久,换言之,获得了一个糟糕的局部最优解(与全局最优解相去甚远)而难以极低概率通过探索机制逃离出去.

上面的原因都导致了对计算资源的疯狂需求.

但是抛开上面的缺陷不谈,我们仍应看到它是可以实现和被利用的一个上限很高的算法.

不过不难料想如果给予足够长的时间,它能做的越来越好.

由此可见DQL的潜力巨大,但是仍然有很多可以改进的地方, DeepMind并没有止步于一开始的DQN，在它上面做了各种各样的改进，性能不断提升。于是，有一天，他们就想，那是不是把各种改进方法都合在一起看看效果呢。然后，Rainbow就横空出世了.

然而以上都只是单机算法,如果拓展成分布式算法,也许可以进一步给DQN带来质的提升.实际上DeepMind也真的做了这件事: Ape-X出来了:Ape-X直接把性能翻了一倍，而且，更关键是还学的更快，快太多了！看上图右边的具体游戏示例，Pong乒乓游戏在大概半小时内就训练到极致了，而这个在初代DQN里面要训练好几天呀！整个算法相对于Rainbow也就是训练架构上发生改变，算法实质并没有变化。

但是使用分布式充分发挥硬件优势,使得DRL可以更快得到验证和落地应用! 这让我们要意识到RL的分布式计算是非常重要的一个环节，不仅仅是加快训练速度，还可以更好的提升效果！只懂得RL算法，不懂得分布式编程恐怕是不够的。特别是未来，对于机器人学习，会有越来越多的仿真环境出来，那么极其需要分布式训练，因此值得我们投入更多的精力在分布式上！

**七、引用文献和资料**

[1]Sutton R S, Barto A G. Reinforcement learning: An introduction[M]. MIT press, 2018.

Q-Learing算法简明教程(5间房子) <https://blog.csdn.net/itplus/article/details/9361915>

[2]小鸟过管道(q\_table) <https://www.zhihu.com/question/26408259>

[3]通过下面这篇文章指出DQN使用NN代替q\_table <https://zhuanlan.zhihu.com/p/33255409>

[4]Q-Learning&&Sarsa&&DQN <http://wulc.me/2018/05/09/%E5%BC%BA%E5%8C%96%E5%AD%A6%E4%B9%A0%E7%AC%94%E8%AE%B0(2)-%E4%BB%8E%20Q-Learning%20%E5%88%B0%20DQN/>

附录：

主要代码：

**from** statistics **import** mean  
**import** matplotlib  
  
matplotlib.use(**'Agg'**)  
**import** matplotlib.pyplot **as** plt  
**from** collections **import** deque  
**import** os  
**import** csv  
**import** numpy **as** np  
  
*# 数据分析文件存放的路径*SCORES\_CSV\_PATH = **"./scores/scores.csv"**SCORES\_PNG\_PATH = **"./scores/scores.png"**SOLVED\_CSV\_PATH = **"./scores/solved.csv"**SOLVED\_PNG\_PATH = **"./scores/solved.png"***# def "solving" in cartpole:连续100次的平均分数在195以上*AVERAGE\_SCORE\_TO\_SOLVE = 195  
CONSECUTIVE\_RUNS\_TO\_SOLVE = 100  
  
  
**class** ScoreLogger: *# 分数日志记录者* **def** \_\_init\_\_(self, env\_name):  
 self.scores = deque(maxlen=CONSECUTIVE\_RUNS\_TO\_SOLVE)  
 self.env\_name = env\_name  
  
 **if** os.path.exists(SCORES\_PNG\_PATH):  
 os.remove(SCORES\_PNG\_PATH)  
 **if** os.path.exists(SCORES\_CSV\_PATH):  
 os.remove(SCORES\_CSV\_PATH)  
  
 **def** add\_score(self, score, run):  
 self.\_save\_csv(SCORES\_CSV\_PATH, score)  
 self.\_save\_png(input\_path=SCORES\_CSV\_PATH,  
 output\_path=SCORES\_PNG\_PATH,  
 x\_label=**"runs"**,  
 y\_label=**"scores"**,  
 average\_of\_n\_last=CONSECUTIVE\_RUNS\_TO\_SOLVE,  
 show\_goal=**True**,  
 show\_trend=**True**,  
 show\_legend=**True**)  
 self.scores.append(score)  
 mean\_score = mean(self.scores)  
 print(**"Scores: (min: "** + str(min(self.scores)) + **", avg: "** + str(mean\_score) + **", max: "** + str(  
 max(self.scores)) + **")\n"**)  
 **if** mean\_score >= AVERAGE\_SCORE\_TO\_SOLVE **and** len(self.scores) >= CONSECUTIVE\_RUNS\_TO\_SOLVE:  
 solve\_score = run - CONSECUTIVE\_RUNS\_TO\_SOLVE  
 print(**"Solved in "** + str(solve\_score) + **" runs, "** + str(run) + **" total runs."**)  
 self.\_save\_csv(SOLVED\_CSV\_PATH, solve\_score)  
 self.\_save\_png(input\_path=SOLVED\_CSV\_PATH,  
 output\_path=SOLVED\_PNG\_PATH,  
 x\_label=**"trials"**,  
 y\_label=**"steps before solve"**,  
 average\_of\_n\_last=**None**,  
 show\_goal=**False**,  
 show\_trend=**False**,  
 show\_legend=**False**)  
 exit()  
  
 **def** \_save\_png(self, input\_path, output\_path, x\_label, y\_label, average\_of\_n\_last, show\_goal, show\_trend,  
 show\_legend):  
 x = []  
 y = []  
 **with** open(input\_path, **"r"**) **as** scores:  
 reader = csv.reader(scores)  
 data = list(reader)  
 **for** i **in** range(0, len(data) - 1):  
 x.append(int(i))  
 y.append(int(data[i][0]))  
  
 plt.subplots()  
 plt.plot(x, y, label=**"score per run"**)  
  
 average\_range = average\_of\_n\_last **if** average\_of\_n\_last **is not None else** len(x)  
 plt.plot(x[-average\_range:], [np.mean(y[-average\_range:])] \* len(y[-average\_range:]), linestyle=**"--"**,  
 label=**"last "** + str(average\_range) + **" runs average"**)  
  
 **if** show\_goal:  
 plt.plot(x, [AVERAGE\_SCORE\_TO\_SOLVE] \* len(x), linestyle=**":"**,  
 label=str(AVERAGE\_SCORE\_TO\_SOLVE) + **" score average goal"**)  
  
 **if** show\_trend **and** len(x) > 1:  
 trend\_x = x[1:]  
 z = np.polyfit(np.array(trend\_x), np.array(y[1:]), 1)  
 p = np.poly1d(z)  
 plt.plot(trend\_x, p(trend\_x), linestyle=**"-."**, label=**"trend"**)  
  
 plt.title(self.env\_name)  
 plt.xlabel(x\_label)  
 plt.ylabel(y\_label)  
  
 **if** show\_legend:  
 plt.legend(loc=**"upper left"**)  
  
 plt.savefig(output\_path, bbox\_inches=**"tight"**)  
 plt.close()  
  
 **def** \_save\_csv(self, path, score):  
 **if not** os.path.exists(path):  
 **with** open(path, **"w"**): *# 新建一个可写文件* **pass** scores\_file = open(path, **"a"**)  
 **with** scores\_file:  
 *# writer = csv.writer(scores\_file)  
 # writer.writerow([score])* scores\_file.write(str(score) + **"\n"**)

**import** random  
**import** gym  
**import** numpy **as** np  
**from** collections **import** deque  
**from** keras.models **import** Sequential  
**from** keras.layers **import** Dense  
**from** keras.optimizers **import** Adam  
  
**from** gym\_learn.cartpole.scores.score\_logger **import** ScoreLogger  
  
ENV\_NAME = **"CartPole-v1"**GAMMA = 0.95  
LEARNING\_RATE = 0.001  
  
MEMORY\_SIZE = 1000000  
BATCH\_SIZE = 20  
  
EXPLORATION\_MAX = 1.0  
EXPLORATION\_MIN = 0.01  
EXPLORATION\_DECAY = 0.995  
  
  
**class** DQNSolver:  
  
 **def** \_\_init\_\_(self, observation\_space, action\_space):  
 self.exploration\_rate = EXPLORATION\_MAX  
  
 self.action\_space = action\_space  
 self.memory = deque(maxlen=MEMORY\_SIZE)  
  
 self.model = Sequential()  
 self.model.add(Dense(24, input\_shape=(observation\_space,), activation=**"relu"**))  
 self.model.add(Dense(24, activation=**"relu"**))  
 self.model.add(Dense(self.action\_space, activation=**"linear"**))  
 self.model.compile(loss=**"mse"**, optimizer=Adam(lr=LEARNING\_RATE))  
  
 **def** remember(self, state, action, reward, next\_state, done):  
 self.memory.append((state, action, reward, next\_state, done))  
  
 **def** act(self, state):  
 **if** np.random.rand() < self.exploration\_rate:  
 **return** random.randrange(self.action\_space)  
 q\_values = self.model.predict(state)  
 **return** np.argmax(q\_values[0])  
  
 **def** experience\_replay(self): *# 经验回放函数* **if** len(self.memory) < BATCH\_SIZE:  
 **return** batch = random.sample(self.memory, BATCH\_SIZE)  
 **for** state, action, reward, state\_next, terminal **in** batch:  
 q\_update = reward  
 **if not** terminal:  
 q\_update = (reward + GAMMA \* np.amax(  
 self.model.predict(state\_next)[0])) *# crucial Q(s,a)=r+γ\*Q\_max(s',a')* q\_values = self.model.predict(state)  
 q\_values[0][action] = q\_update  
 self.model.fit(state, q\_values, verbose=0)  
 self.exploration\_rate \*= EXPLORATION\_DECAY  
 self.exploration\_rate = max(EXPLORATION\_MIN, self.exploration\_rate)  
  
  
**def** cartpole():  
 env = gym.make(ENV\_NAME) *# 构造环境* score\_logger = ScoreLogger(ENV\_NAME) *# 日志* observation\_space = env.observation\_space.shape[0]  
 action\_space = env.action\_space.n  
 dqn\_solver = DQNSolver(observation\_space, action\_space) *# 根据观察空间和动作空间创建一个代理* run = 0 *# 运行次数* **while True**:  
 run += 1  
 state = env.reset() *#每次运行都初始化新环境* state = np.reshape(state, [1, observation\_space]) *#设置初始状态* step = 0  
 **while True**:  
 step += 1  
 *# env.render()* action = dqn\_solver.act(state)  
 state\_next, reward, terminal, info = env.step(action)  
 reward = reward **if not** terminal **else** -reward  
 state\_next = np.reshape(state\_next, [1, observation\_space])  
 dqn\_solver.remember(state, action, reward, state\_next, terminal)*#要求代理记住这些数据,因为需要经验回放* state = state\_next  
 **if** terminal:  
 print(  
 **"Run: "** + str(run) + **", exploration: "** + str(dqn\_solver.exploration\_rate) + **", score: "** + str(step))  
 score\_logger.add\_score(step, run)*# step就是score!* **break** dqn\_solver.experience\_replay()  
  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 cartpole()

代码已开源，地址为：