**强化学习理论学习与代码实现**

# 深度强化学习-深度Q网络

2013年Mnih等[1]提出了第一个使用强化学习从高维感官输入直接学习控制策略的深度学习模型。该模型是一个卷积神经网络，使用带有逼近器的Q-learning的进行训练，输入为原始像素，输出未用来估计未来奖励的值函数。DQN在架构或学习算法不需调整的情况下，应用于7个来自街机学习环境的雅达利2600游戏，实现结果表明，它在6个游戏上的表现超过了之前的所有方法，并且在3个游戏上也超过了人类专家。DQN的提出可以说代表着深度强化学习时代的到来。

随后业界开始了一股深度强化学习的热潮，数据源Web of Science中标题包含“Deep Reinforcement Learning”的文献，自2013年以来共有751篇，平均3.4天就产生一篇。从图1可以看出，论文发表数量逐渐递增，到2018年达到顶峰。从图2可知，发表最多论文的出版物是IEEE Access，还有不少IEEE的国际会议。图3中中国遥遥占据领先位置，美国和英国分列二三位。图4列出了主要的研究领域，集中在计算机科学、工程和电讯等领域。北邮、中科院和加州大学伯克利分校是发表文章最多的前三甲（图5）。

图1 每年发表论文数量



图2 主要发表的出版物

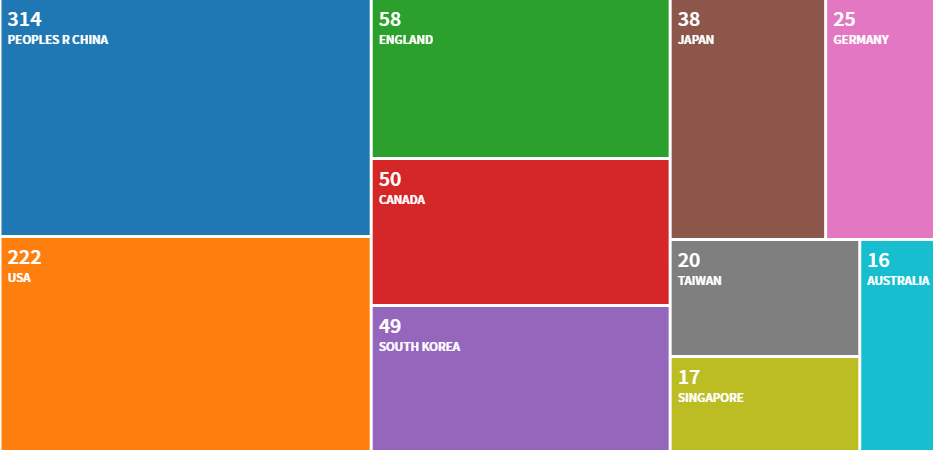


图3 发表国家和地区

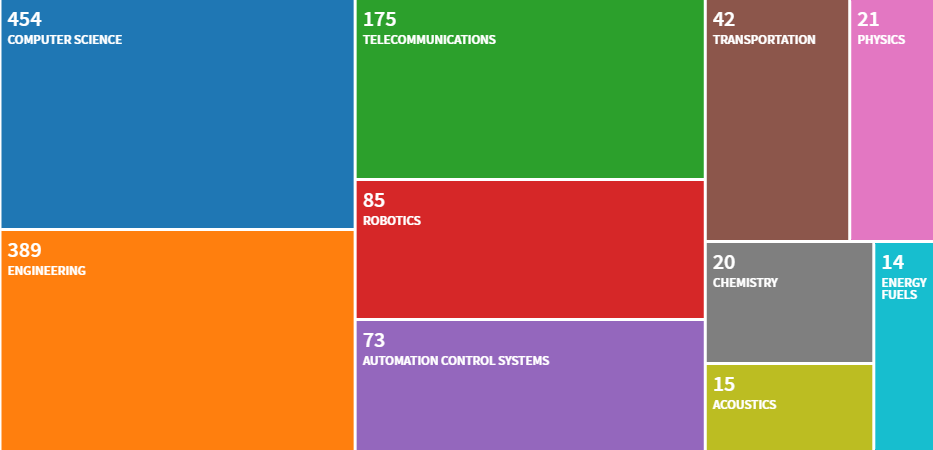


图4 研究领域



图5 研究机构

本节内容将从DQN开始，逐步了解学习各种经典的深度强化学习算法，针对不同算法的不足，研究者们都做了哪些改进。

## 学习目标

* 理解Deep Q-Learning（DQN）算法；
* 理解为什么经验回放和目标网络对深度DQN在实践中起作用是必要的；
* 理解Double DQN；
* 理解优先级经验回放。

## 深度学习和强化学习的区别

深度学习是一种典型的监督学习方式，基于大量带有标签的数据进行预测（回归问题）或分类（分类问题），而强化学习则是通过与环境不断地交互获得奖励，并基于这些奖励调整学习过程以获得全局最优的行为策略。可以看出，两者在训练数据和学习过程等方面存在巨大差异，主要表现在以下几个方面：

1. 深度学习有固定明确的监督信号即标签，而强化学习缺乏监督信号且只有环境反馈的奖励信号，奖励本身也可能存在噪声、延迟和稀疏性等问题。
2. 深度学习中的样本一般满足独立同分布，而强化学习的状态和行为分布存在前后相关性。
3. 深度学习的网络结构可以用于刻画非线性函数关系，但在实际任务中，采用非线性的深度学习网络结构表示值函数时，可能会引起强化学习算法的损失值设置损失函数都无法收敛。

## DQN原理

接下来一起看一下DQN是如何将深度学习和强化学习结合在一起的。前面也提到过，强化学习从标量奖励中学习，而这些奖励往往稀疏、有噪声、延迟，状态序列也是高度相关的，同时随着强化学习的进行，数据分布还会发生变化。为了克服这些问题，Mnih等采用了一个卷积神经网络，在复杂的RL环境中，采用随机梯度下降直接从原始的视频数据中学习控制策略。为缓和数据相关和非平稳分布的问题，还引入了经验回放（experience replay）机制，从先前的状态转移中随机采样，从而使得在已经历行为上训练分布平稳。

在玩Atari游戏中，环境就是Atari模拟器，在每一时间步，强化学习代理从可选行为集合中选择一个动作，该动作被输入到模拟器中，然后改变内部状态和游戏得分。通常环境是随机的，代理无法感知模拟器的内部状态，而是观察到一幅图像，该图像是表达当前屏幕的由原始像素值构成的向量。此外代理还会收到一个奖励来表示游戏得分的变化，通常游戏得分和之前的整个动作和观察都有关系。

由于代理只观察当前屏幕的图像，因此学习过程是部分可观的，仅从当前屏幕不可能完全理解当前情况，为此作者考虑了动作和观察序列，根据这些序列来学习游戏策略。

代理的目标就是通过选择动作与模拟器交互，以最大化未来奖励。文中做了一个标准假设，即未来奖励是每时间步按γ折扣的，定义t时刻的未来折扣回报为 $R\_{t}=\sum\_{t^{\prime}=t}^{T} \gamma^{t^{\prime}-t} r\_{t^{\prime}}$，其中T为游戏终止的时间。最优动作值函数为Q\*(s,a)，则Q\*(s,a)为遵循某一策略所能得到的最大期望回报，即 $Q^{\*}(s, a)=\max \_{\pi} \mathbb{E}\left[R\_{t} | s\_{t}=s, a\_{t}=a, \pi\right]$。

最优动作值函数遵循一个重要的等式，即贝尔曼方程：

 （1）

$$

Q^{\*}(s, a)=\mathbb{E}\_{s^{\prime} \sim \mathcal{E}}\left[r+\gamma \max \_{a^{\prime}} Q^{\*}\left(s^{\prime}, a^{\prime}\right) | s, a\right]

$$

强化学习的基本思想就是使用贝尔曼方程作为迭代更新来估计动作值函数， $Q\_{i+1}(s, a)=\mathbb{E}\left[r+\gamma \max \_{a^{\prime}} Q\_{i}\left(s^{\prime}, a^{\prime}\right) | s, a\right]$，当迭代次数趋于∞时，Q收敛于Q\*。但是在实际中，这种方法是完全不可行的，因为动作值函数是为每个序列单独估计的，没有任何泛化、因此需要通过函数逼近器来估计动作值函数 $Q(s, a ; \theta) \approx Q^{\*}(s, a)$。可以将具有权重θ的神经网络叫作Q网络，通过最小化损失函数Li(θi)就可以训练Q网络。

 （2）

$$

L\_{i}\left(\theta\_{i}\right)=\mathbb{E}\_{s, a \sim \rho(\cdot)}\left[\left(y\_{i}-Q\left(s, a ; \theta\_{i}\right)\right)^{2}\right]

$$

其中$y\_{i}=\mathbb{E}\_{s^{\prime} \sim \mathcal{E}}\left[r+\gamma \max \_{a^{\prime}} Q\left(s^{\prime}, a^{\prime} ; \theta\_{i-1}\right) | s, a\right]$为第i次迭代的目标，ρ(s,a)是序列s和动作a上的概率分布。当最优化损失函数Li(θi)时前一次迭代的参数θi-1保持不变。注意这里的目标是和权重有关的，与监督学习中使用的固定目标不同。对损失函数求权重上的导数可得以下梯度：

 （3）

$$

\nabla\_{\theta\_{i}} L\_{i}\left(\theta\_{i}\right)=\mathbb{E}\_{s, a \sim \rho(\cdot) ; s^{\prime} \sim \mathcal{E}}\left[\left(r+\gamma \max \_{a^{\prime}} Q\left(s^{\prime}, a^{\prime} ; \theta\_{i-1}\right)-Q\left(s, a ; \theta\_{i}\right)\right) \nabla\_{\theta\_{i}} Q\left(s, a ; \theta\_{i}\right)\right]

$$

该算法是无模型的：因为它直接利用模拟器中的样本来解决强化学习任务，而没有明确地构造模拟器的估计。该算法也是离策略的：它在遵循一种行为分布以保证充分探索状态空间的同时，学习了一个贪婪策略 $a=\max \_{a} Q(s, a ; \theta)$。

## DQN算法

DQN将强化学习算法与深度神经网络连接起来，这种深度神经网络直接作用于RGB图像，并利用随机梯度更新有效地处理训练数据，该算法借鉴了Tesauro的TD- Gammon结构，但不同于TD-Gammon的是，DQN采用了经验回放技术，将代理在每一时间步的经验 $e\_{t}=\left(s\_{t}, a\_{t}, r\_{t}, s\_{t+1}\right)$储存于数据集 $\mathcal{D}=e\_{1}, \dots, e\_{N}$中。算法的内循环过程中，从样本池中随机抽取经验样本 $e \sim \mathcal{D}$，采用Q学习更新或最小批量个更新。执行完经验回放后，代理根据ε-贪婪策略选择并执行一个动作，由于使用任意长度的历史记录作为神经网络的输入是困难的，因此Q函数是作用于由函数Φ生成的固定长度的历史表达。该完整算法在文中被称为深度Q学习（Deep Q-learning），下面给出其算法伪代码。

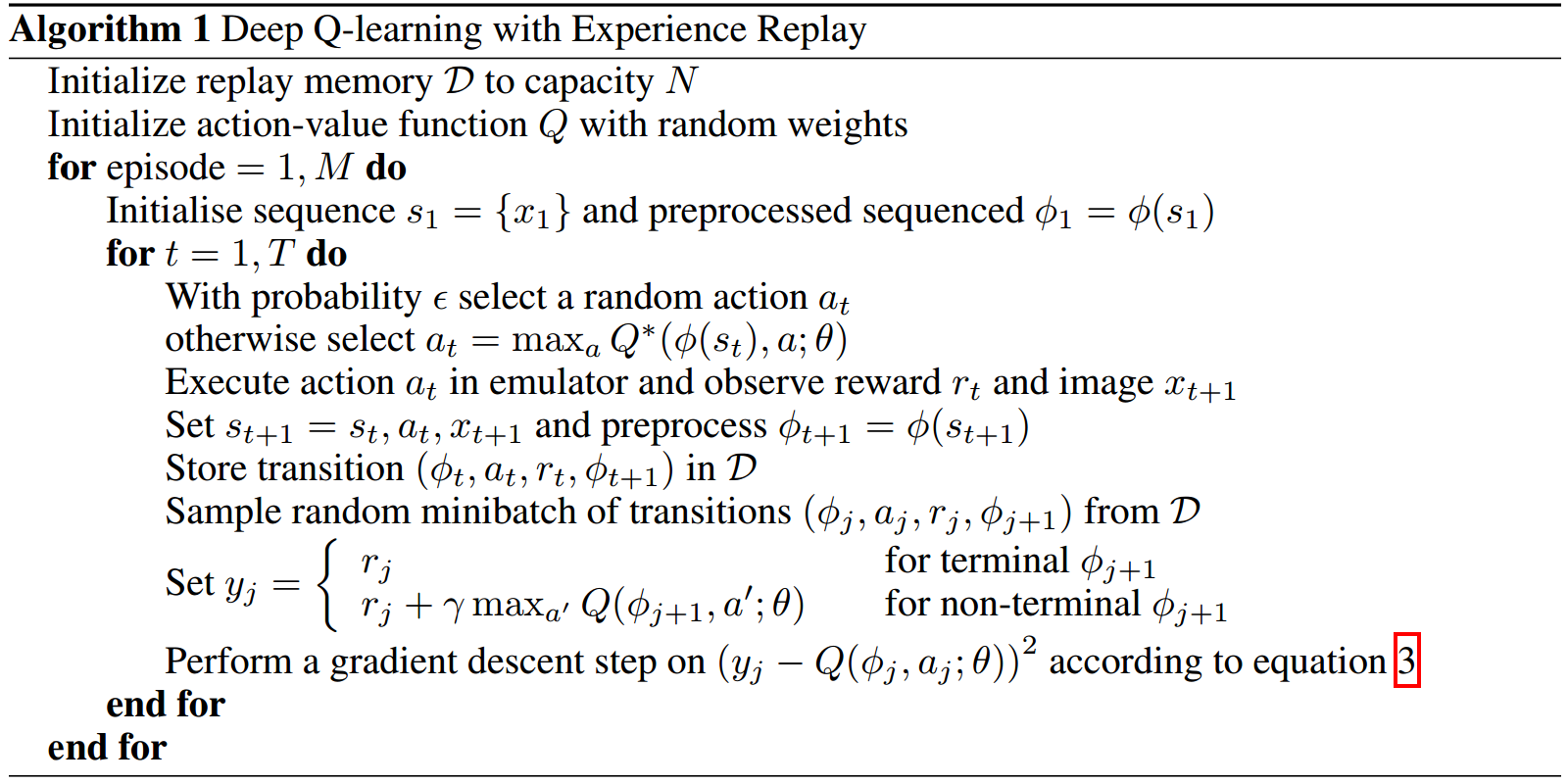


图6 使用经验回放的深度Q学习

算法中各步骤的具体含义如下：

Step1：初始化经验池 $\mathcal{D}$，其容量为N（即可存储N个历史样本），用作历史经验回放。

Step2：使用卷积神经网络作为预测Q值的逼近器，并初始化该网络模型的权重参数θ。

Step3：设定游戏片段总数为M，即代理最多执行游戏次数为M次。

Step4：初始化网络输入，得到系列s1={x1}，并预处理得到Φ1=Φ(s1)。

Step5：以概率ε随机选择代理需要执行的动作at，或以概率1-ε执行网络输出最大Q值所对应的动作。

Step6：代理执行动作at，获得环境反馈的奖励rt和下一时间步的游戏图像xt+1。

Step7：基于新的状态st+1=st，at，xt+1，根据Φt+1=Φ(st+1)计算下一时间步的固定序列游戏帧Φt+1。

Step8：将获得的状态转移(Φt,at,rt, Φt+1)存入经验池 $\mathcal{D}$中。

Step9：代理从 $\mathcal{D}$中随机采样最小批量转移。

Step10：计算每一个状态的目标值，代理通过执行动作后的奖励rj来更新Q值作为Q学习算法的目标值。

Step11：根据式（3），基于最小批量样本采用随机梯度下降法更新Q网络权重参数θ。

相比较于标准的在线Q学习，DQN的优势在于：

1. 每一步产生的经验都可能用于多次权重更新，从而提高了数据利用率。
2. 直接从连续样本中学习效率较低，因为样本之间具有强相关性，随机采样可以打破这种相关性，从而降低更新的方差；
3. 当学习在策略时，当前参数决定了用于参数训练的下一数据样本。

### 预处理

原始的Atari游戏的每一帧是一幅210x160像素的128色图像，直接对该图像进行操作将会需要大量的计算，为此先进行预处理以降低输入维度。首先将RGB转换成灰度图，并下采样成110x84的图像，最后只保留大致可以捕捉游戏范围的84x84区域作为最终的输入。至于为什么图像的宽和高相等，是因为采用了2D卷积的GPU实现，该操作需要方形输入。前面提到为了保证输入相同，函数Φ会得到固定长度的历史表达，这里对最近4幅图像进行预处理，并叠在一起作为Q函数的输入。

### 结构

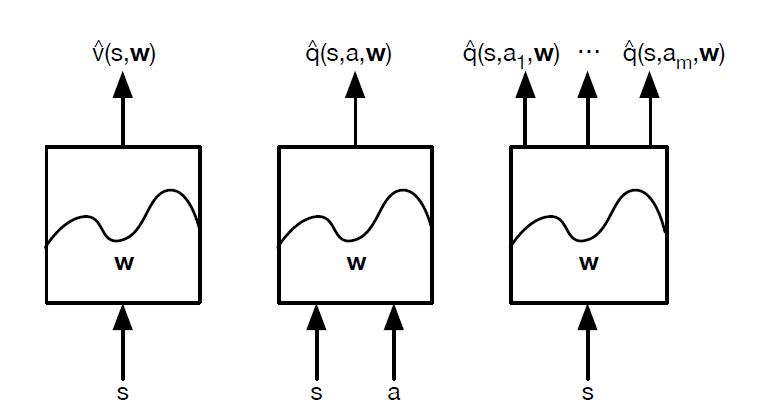


图7 两种不同的神经网路结构

使用神经网络参数化Q可以有多种不同的方法，先前的方法是将状态历史和动作输入到神经网络中，这种结构的缺点在于计算每一个动作Q值都需要一次前向传播，导致计算成本与动作数量成正比。DQN采用了另一种结构，每个可能的动作都有自己单独的输出单元，只需将状态表达输入到神经网络中，其优点是只需一次前向传播就能够计算给定状态下所有可能动作的Q值。

在Atari游戏中，输入到神经网络的是一幅84x84x4的图像，第一个隐藏层采用步长为4大小为168x8的滤波器对输入图像进行卷积，并应用非线性整流器，第二个隐藏层采用步长为2大小为324x4的滤波器进行卷积，同样应用非线性整流器，最后一个隐藏层是一个全连接层，包含256个校正单元。输出层是一个全连接的线性层，每一个有效动作对应一个输出。将采用这种方法训练得到的卷神经网络称为深度Q网络（DQN）。

## 实验结果

文中对7个Atari游戏进行了实验，并在不同的游戏上保持相同的网络结构、学习算法和超参数设置，这就是深度强化学习的优势所在。在使用真实游戏评估代理时，只在训练期间对游戏的奖励结构做了一个改变。由于不同游戏间得分范围变化很大，将所有正奖励设置为1，所有负奖励设置为-1，奖励不变时设置为0，通过这样的裁剪处理就可以限制误差导数的范围，使其更容易在多个游戏间使用相同的学习率。同时也会影响代理的性能，因为代理无法区分不同规模的奖励。

在这些实验中，使用了最小批量为32的RMSProp算法，训练过程中行为策略采用的是ε-贪婪策略，ε在前100万帧线性的从1递减至0.1，然后保持不变，总共训练1000万帧，并将最近的10万帧进行回访存储。

文中还使用了跳帧技术，准确地说，代理每隔k帧才会选择动作，而是每一帧，在跳过的帧中会重复采取上一动作，由于将模拟器向前运行一步所需的计算量要比让代理选择一个动作少得多，因此这种技术允许代理在不显著增加运行时的情况下多玩大约k倍的游戏。

### 训练和稳定性

在这里强化学习的评估指标是多个游戏上的平均片段总奖励。该指标可能会有较多噪声，因为策略权重的微小变化可能导致策略已访问状态分布的巨大变化，图8和图9就可以说明这一点，图中曲线给人的印象是学习算法并没有稳定的收敛。

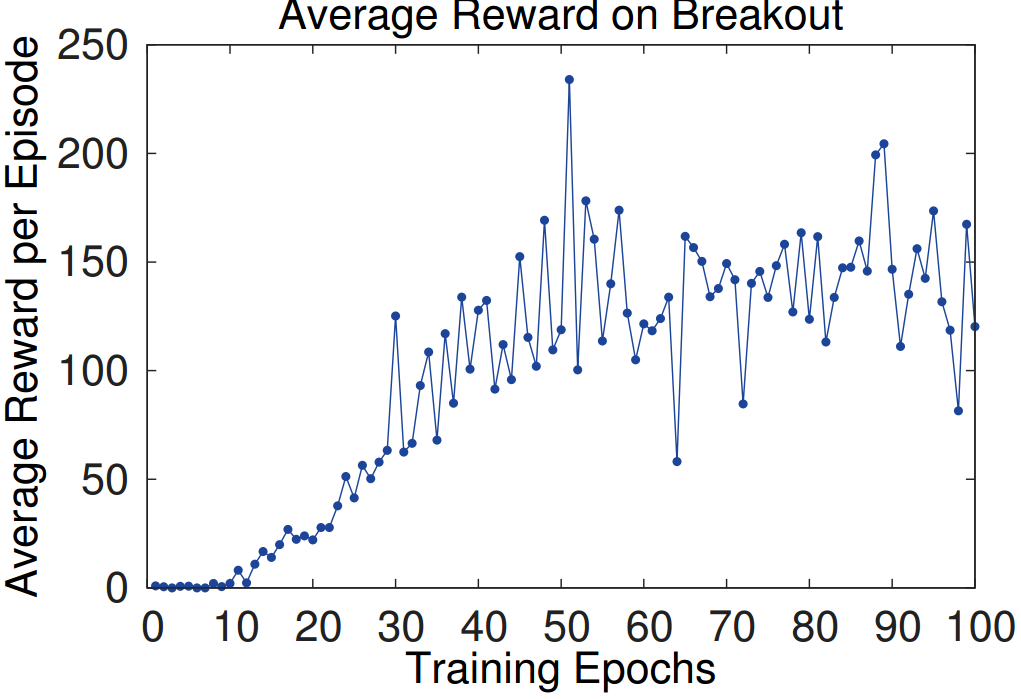


图8 Breakout游戏上的每代平均奖励

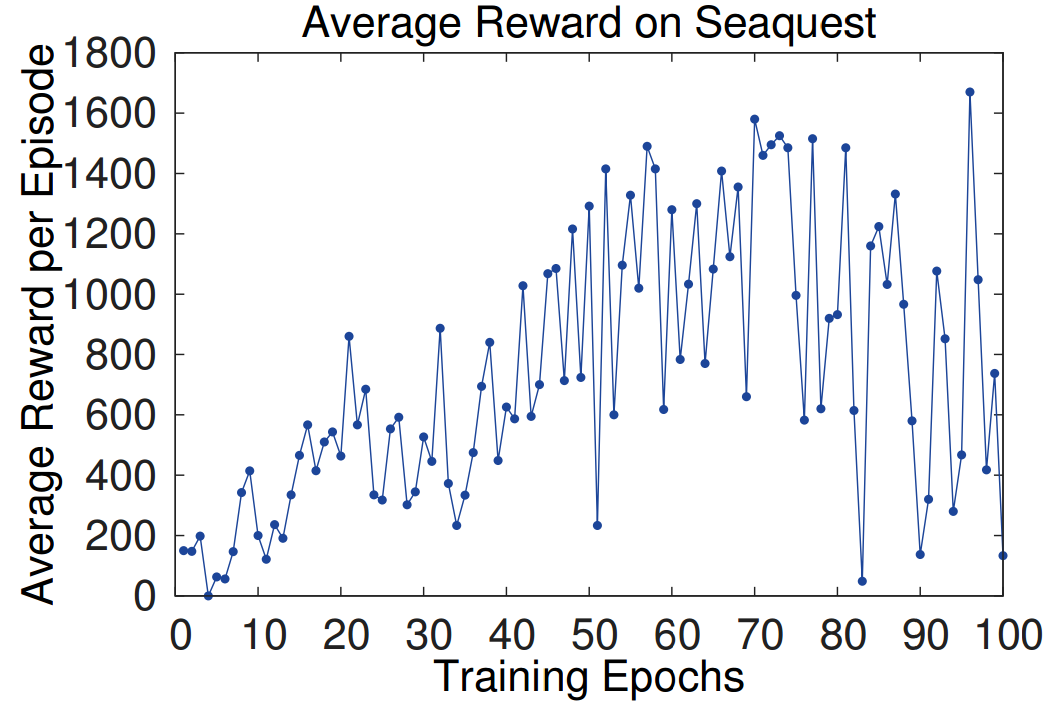


图9 Seaquest游戏上的每代平均奖励

而另一个更加稳定的指标是策略的动作值函数估计值Q，代表从给定的状态开始遵循某一策略代理所能获得的折扣奖励。在开始训练前先收集随机策略下一组固定的状态，并记录这些状态的最大预测Q值的平均值，从图10和图11可以看出该指标更加平滑，也没有发散问题，这说明尽管缺乏任何理论上的收敛性保证，该方法能够非常稳定地利用强化学习信号和随机梯度下降训练大型神经网络。

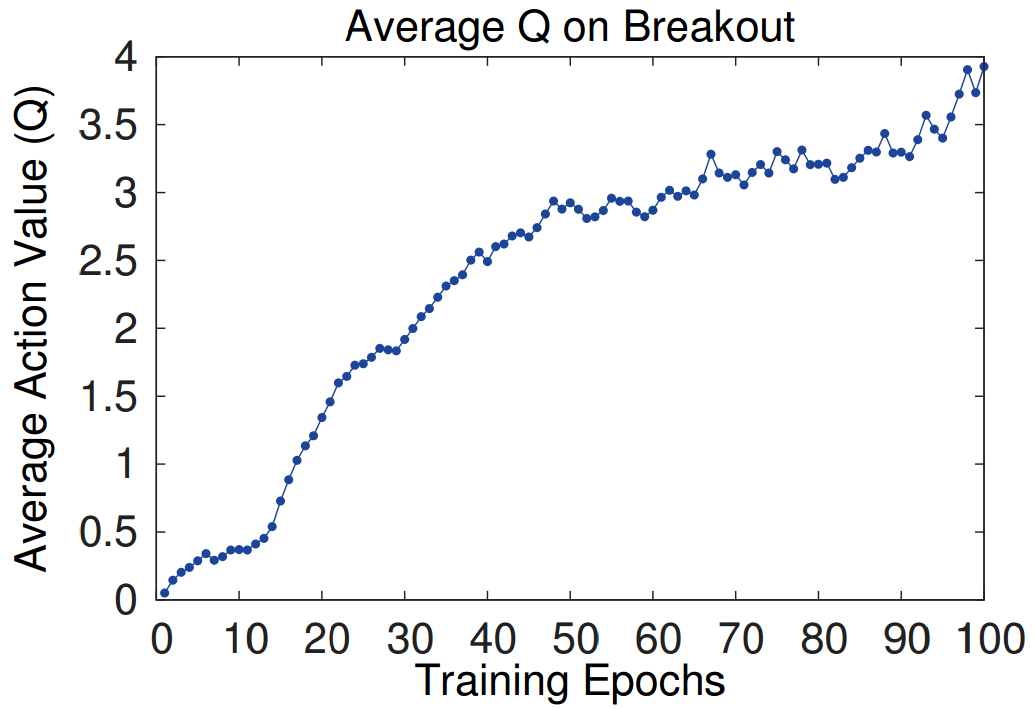


图10 Breakout游戏上的平均最大预测动作值

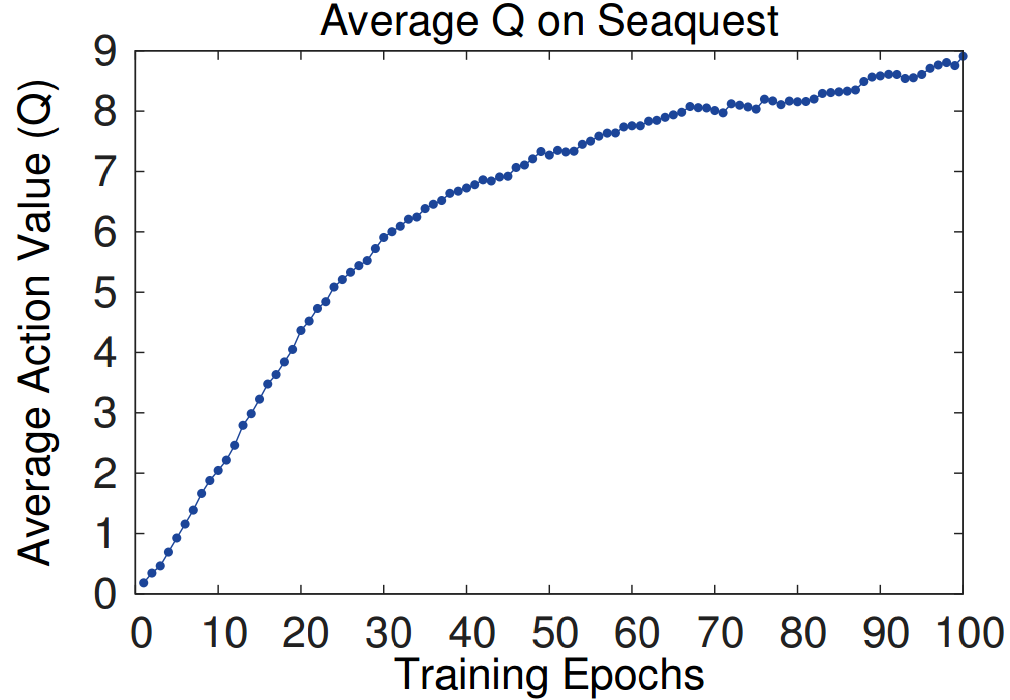


图11 Seaquest游戏上的平均最大预测动作值

### 可视化值函数

图12中展示的是在游戏Seaquest上学习到的值函数，当在屏幕左边有敌人出现时，预测值会跳跃（图13对应A点），然后代理向敌人发射鱼雷，当鱼雷即将击中敌人时，预测值达到峰值（图14对应B点），最后敌人消失后，估计值又会下降到原始值（图15对应C点）。这说明该方法能够学习到在一系列相当复杂的事件中值函数是如何进化的。

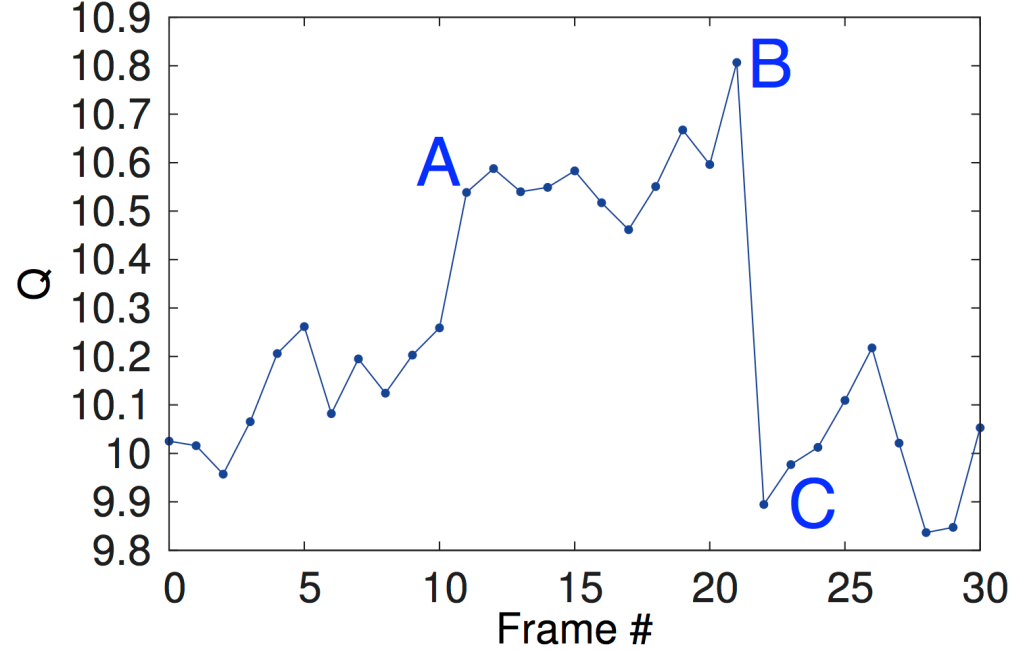


图12 Seaquest游戏在一个30帧的片段上的预测值函数



图13 对应A点的截图



图14 对应B点的截图



图15 对应C点的截图

### 对比

表1 DQN vs 其他学习算法和人类玩家

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **B. Rider** | **Breakout** | **Enduro** | **Pong** | **Q\*bert** | **Seaquest** | **S. Invaders** |
| **Random** | 354 | 1.2 | 0 | *-*20.4 | 157 | 110 | 179 |
| **Sarsa** | 996 | 5.2 | 129 | *-*19 | 614 | 665 | 271 |
| **Contingency** | 1743 | 6 | 159 | *-*17 | 960 | 723 | 268 |
| **DQN** | **4092** | **168** | **470** | **20** | **1952** | **1705** | **581** |
| **Human** | 7456 | 31 | 368 | *-*3 | 18900 | 28010 | 3690 |

表2 DQN vs 进化策略搜索

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **HNeat Best** | 3616 | 52 | 106 | 19 | 1800 | 920 | **1720** |
| **HNeat Pixel** | 1332 | 4 | 91 | *-*16 | 1325 | 800 | 1145 |
| **DQN Best** | **5184** | **225** | **661** | **21** | **4500** | **1740** | 1075 |

## 感悟

目前正在研究的基于深度强化学习的柔性作业车间调度问题根本上也是以这篇文章为出发点，利用深度强化学习针对不同的调度状态自适应地做出较优的决策，但是该方法也有很明显的弊端，自己做实验时也有体会，后面也会介绍针对存在的问题都有哪些改进。

参 考 文 献

[1] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Playing Atari with Deep Reinforcement Learning[J]. Computer Science. 2013.