DQN2013

Abstract：应用Q-learning变种训练的CNN模型（DQN），能够玩游戏（输入游戏画面/原始像素，输出动作），效果很好。

Introduction：

1 原先的RL很难用高维输入（比如图像）控制agent

2 DL可以从高维输入中提取信息，所以我们打算用DL优化RL

3 但是有一些问题

4 这篇paper将展示CNN可以解决这些问题

5 我们用Atari 2600 去测试我们的方法。我们的目标是只要训练一个agent就能学会玩尽量多的游戏。Agent知道的只有游戏画面、reward、终端信号、动作空间。结果很好

Backward：

1 流程：agent输入代表游戏画面的像素向量和reward，输出动作种类

需要注意的是总游戏得分可能会依赖于整个动作和画面（observation）的顺序；一个动作的feedback可能很久以后才能收到

2 Agent很难只通过当前的画面就了解情况，我们考虑使用一系列动作和observation来训练 “st = x1, a1, x2, ..., at−1, xt”,序列在有限的时间后结束。这个序列可以看作是一个MDP，所以我们可以用MDP的方法去做这玩意。

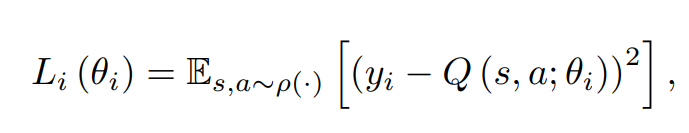
3 

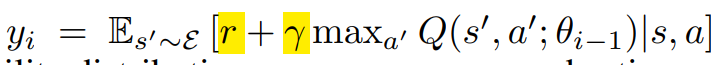


Rt是衰减的、未来奖励的和。 最佳S-A函数

4 Q函数遵守贝尔曼方程

5 利用贝尔曼方程更新Q函数在实践中是不行的，因为Q函数只估计一个s的，没有归纳一堆s的。常见的作法是用一个function approximator（函数近似器）来估计Q，可以用线性也可以用非线性的（比如神经网络）。我们用神经网络作为Q-network。





Loss函数定义，i代表每次迭代，yi是i的target，p（s，a）是s，a的概率分布，作为动作（行为）分布。注意target依赖于nn的参数θ，这和监督学习不一样。

6 用SGD优化。如果weights每个时间戳都更新，并且expectation被动作分布和emulator各自替代，这就很像是Q-learning

7注意算法不依赖于模型，同时也是off-policy的。（∈-greedy strategy没看懂）

Related work：

1介绍 TD-Gammon算法，这是一项应用在西洋双陆棋上的方法，使用时间差分学习(Temporal-Difference Learning)

2 TD算法在国际象棋，围棋和跳棋上的应用不成功

3 TD结合无模型算法、非线性函数近似器，off-policy可能会使Q函数发散。后来更多的用线性近似器

4最近，结合DL和RL的工作又火了。Deep nn被用于评估环境，restricted Boltzmann machines被用于评估Value函数或policy。另外，Q函数发散问题一定程度上被gradient temporal-difference解决

\*5 和我们工作最相似的是NFQ，这是一种用于解决MDP过程的算法。

6使用Atari 2600 emulator作为强化学习的测试平台这一想法被xx文章最初提到。

DRL：

1 和上面的内容基本一致

2 TD-Gammon是本方法的出发点

3 和TD不同，我们使用经验回放技术，即存储过去的经历



我们用Q-learning进行更新。Agent通过挑选动作

4这个方法比standard online Q-learning要好：

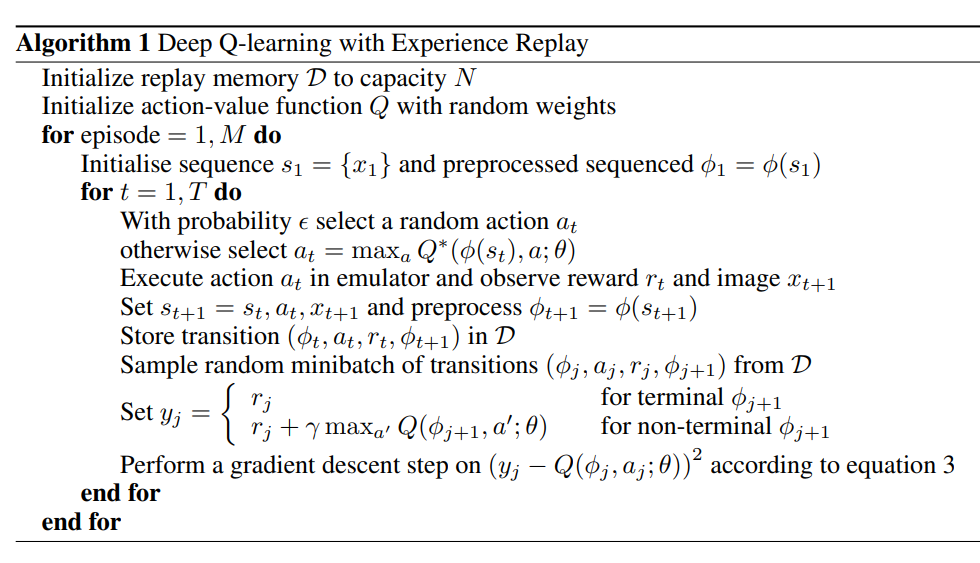
首先，一次经验可应用在多次权重更新上，这可以提高数据利用效率。

其次，直接用连续的样本学习效率不高，因为样本之间的关联性比较高（样本很多信息之间是重复或者相似的，所以说学习相似的样本就相当于重复学习，不如相似度较低的样本那样有效）

第三，经验回放是off-policy的，不会像on-policy那样训练完样本就不能用了，效率会高很多。

实践中，DQN算法只在缓冲区中存储了最后一个N经验，并且当更新时只从D中随机的抽取一项经验，这是因为缓冲区大小有限，并且不能分辨出更为重要的translation（给予相同的重要性）。如果采用一种更加复制的采样（sampling）策略也许就能辨别出更重要的translation。

DQN算法的伪代码见下：



初始化 经验回放的缓冲区D #以存储N

初始化 Q函数 ；Q函数中的权重随机化

For episode=1，M do

初始化列表s={x}和预处理后的列表 

For t=1,T do

根据概率ε随机选择一个动作a

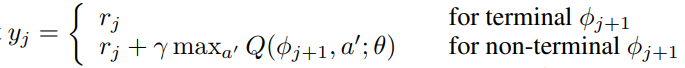
否则选择

在模拟器（环境）中执行a并返回reward r和image x

使并作预处理

把转换四元组储存到D中

在D中随机抽一小批转换组

使

#terminal，最后一个预处理后的状态

用前面提到的公式做梯度下降

结束循环

结束循环

预处理部分：

210x160的图片with128种颜色=》110x84 的图片with灰度图=》84x84的灰度图（二维卷积层需要方阵作为输入）。预处理函数Φ会把最近的四帧图像连起来作为Q函数的输入。

架构：

输入层 84x84x4（预处理后）

Hid1 16个8x8的卷积核，stride=4

ReLU

Hid2 8个4x4的卷积核，stride=2

ReLU

Hid3 全连接，有256个ReLU单元（到256维的线性层+ReLU ？）

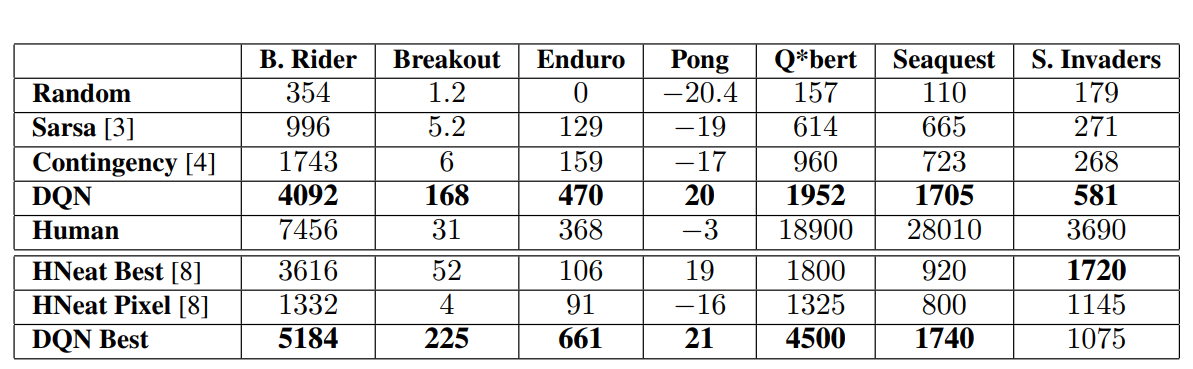
输出层 全连接，输出维数为动作的种类数目。

这就是DQN

Experiment：

采用7种游戏，奖励统一为1 0 -1（正奖励 0 负奖励），采用RMSprop和ε-greedy算法，训练了10M帧，经验回放缓冲区为1M

和其他算法及人类的结果比较。



Mnih2015

Abstract：

RL提供了一种规范性说明，但是agent在处理高维信息时遇到了问题。人和动物使用RL和多级感官系统来学习。RL只在低维状态空间中取得好效果。我们用Deep nn来创建一个新的模型，Deep Q-learning，它可以处理高维信息，并在多个游戏中取得很好的效果。

DQN部分和DQN2013内容基本一样，只做了一些优化，例如微调了模型架构，误差裁剪，预处理部分做了微调等。