DQN往后的发展与改进



目录

[DQN往后的发展与改进 1](#_Toc51752394)

[1 Double DQN 2](#_Toc51752395)

[2 Prioritized Replay 3](#_Toc51752396)

[3 Dueling Network 4](#_Toc51752397)

[4 改进探索状态空间的方式 6](#_Toc51752398)

[4.1 Bootstrapped DQN 6](#_Toc51752399)

[4.2 Incentivizing Exploration in RL with Deep Predictive model 6](#_Toc51752400)

[5 改进网络结构 7](#_Toc51752401)

[5.1 Deep Recurrent Q-learning for Partially Observable MDPS 7](#_Toc51752402)

[6 DQN的多任务和迁移学习 8](#_Toc51752403)

[6.1 Policy Distillation 8](#_Toc51752404)

[6.2 Actor Mimic Deep Multitask and Transfer Reinforcement Learning 9](#_Toc51752405)

[7 加快DQN的训练速度 9](#_Toc51752406)

[7.1 Asynchronous Methods for Deep Reinforcement Learning 9](#_Toc51752407)

[8 DQN在连续控制上的应用 9](#_Toc51752408)

[8.1 Continuous Deep Q-learning with NAF 9](#_Toc51752409)

## 1 Double DQN

为了解决过度估计问题(over estimation)。

因为DQN的Q值是由下一个状态的Q值的最大值来估算的，而下一个状态的Q值也是一个估算值，也依赖于再下一个状态的Q值，这就导致了Q值往往会有偏大的情况出现（over estimation）。

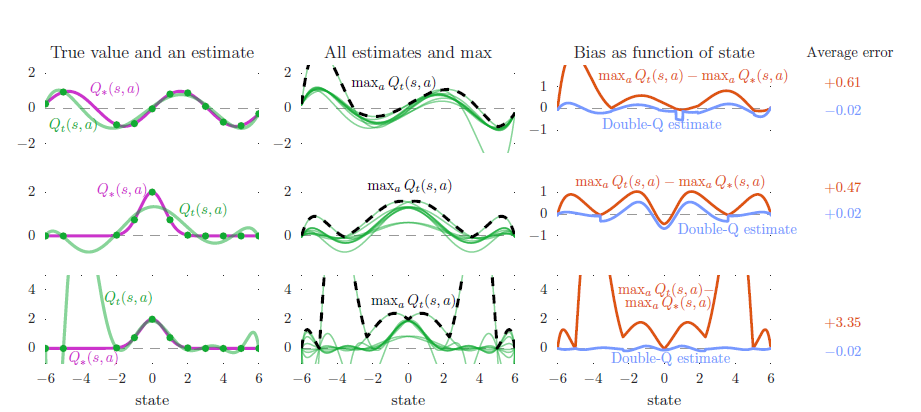


Figure 1 illustration of overestimations

Double DQN就是为了解决这个误差而创造的。

与DQN直接选取目标网络（Target Q Network）中下一个状态各个action对应的Q值最大的那一个Q值不同，Double DQN先使用预测网络（Predict Q Network）计算下一个状态对应的各个action的Q值，然后选取最大的那个Q值对应的action，再使用目标网络计算这个action对应各个状态的Q值，然后选取预测网络中给定action对应的Q值，而此时Q值可能并不是最大的一个。

两个神经网络：Q\_target，Q\_predict

DQN:

图片包含 游戏机

描述已自动生成

Double DQN:

图片包含 游戏机

描述已自动生成

地图上有字

描述已自动生成

Figure 2 Results of DQN and Double DQN

## 2 Prioritized Replay

经验池回放（Experience Replay）使得强化学习的agent可以记住并重复使用过去的经验，并且打破了数据之间的关联性，使数据满足独立分布。但是在之前，在经验池中选取其中一条记录（transition，元组形式，包含state, action, reward, discount factor, next state），是通过均匀随机采样获取，即所有经验池中包含的记录被获取的概率是相同的。为此，提出了一种优先回放的结构，使得重要的经验被回放的概率变大，从而使得学习更加有效率。

经验池中不同记录的重要性是由，RL agent处于当前状态时，可以从记录中学习的数量决定的。具体来说，就是记录（transition）的TD误差的大小，即目标Q值与当前Q值的差距。TD误差大的样本，拥有更高的优先级。

TD error:



具体实现方法：经验池容量为N，采用二进制堆数据结构（binary heap data structure）来存储优先级序列。这样找到最高优先级的记录的复杂度是O(1)，更新优先级的复杂度是O(logN)。

但是，greedy TD-error prioritization算法有几个问题：

1. 为了避免由于扫描整个经验池带来的昂贵代价，只会更新被重放的记录的TD误差值。这样会造成：第一次见到过的一个TD误差值很小的记录存放到经验池后，在很长时间都不会被回放。
2. TD误差值对于噪声（当reward是随机的）很敏感，容易将源于噪声的估计误差加入。
3. 这个算法只专注于经验的一小部分，TD误差减小慢，特别是当在使用函数近似的时候。这意味着开始高的TD误差的记录会重放频繁，导致多样性丧失，使得模型过拟合。

为解决这个问题，又提出了一个叫stochastic sampling的方法，一种介于纯粹的greedy prioritization的算法与原始uniform random sampling算法之间的一种方法。用这种方法保证了记录被抽取的概率是按照优先级单调的，同时又保证了对于低的优先级的记录，不会被0概率抽取。具体来说，抽取标号为i的记录的概率定义为：

手机屏幕的截图

描述已自动生成

这里，是记录i的优先级，均大于0，决定了多少优先级被使用，如果=0，就只用uniform random sampling被使用，优先级没有起到任何作用。

对于的两种表示：

1. Proportional prioritization, , 是一个很小的常数，使得一些TD误差为0的特殊例子也能够被抽取.
2. Rank-based prioritization, ，这里的是按照排序时记录i的等级。

地图的截图

描述已自动生成

Figure 3 Result of prioritized replay

## 3 Dueling Network

提出了一种全新的神经网络结构，用于model-free的RL。

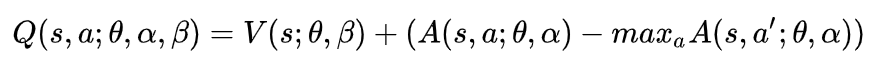
图片包含 游戏机, 文字

描述已自动生成

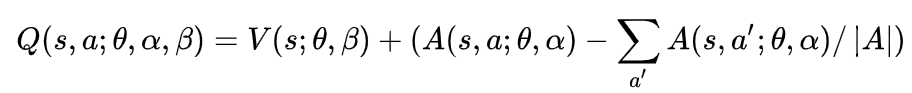
Figure 4 architecture of dueling network (the bottom one)

可以从上图看出，普通的DQN只有一个输出—每个动作的Q值；Dueling DQN则拆成了state的Value和每个动作的advantage。









Dueling DQN将Q值拆分成V值（当前state的值）与加上每个动作在这个state上的advantage。（因为在某些state，无论做什么动作，都对下一个state没有影响）。

地图的截图

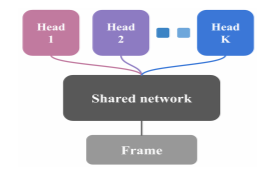
描述已自动生成

Figure 4 Performance of Dueling Network(green) VS Single-stream Network(red)

## 4 改进探索状态空间的方式

### 4.1 Bootstrapped DQN

用DQN实现posterior sampling探索的方法，代替-greedy, 加快学习速度。提出的新的DQN架构支持从若干个互不相同的Q函数中选择动作，这些Q函数通过训练Bootstrap Data得到，架构如下：



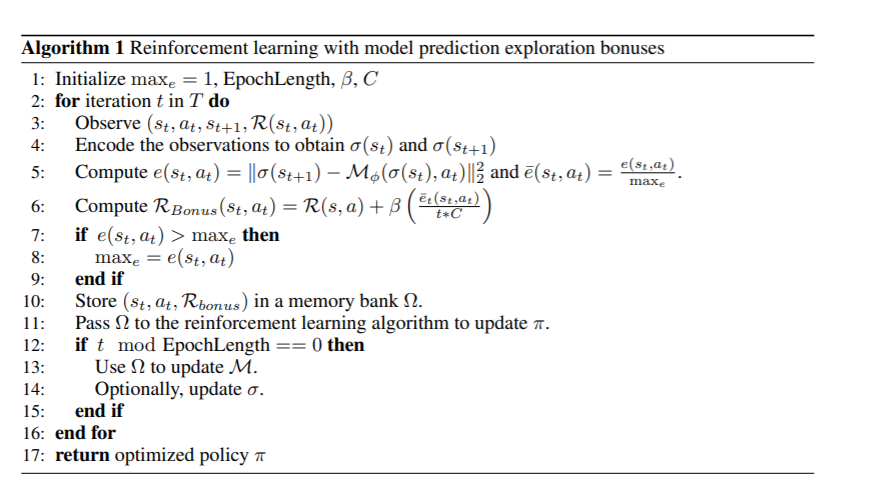
* 最上层是K个端口（head），也就是K个互不相同的Q函数。这K个Q函数在数据集的一个subset上进行训练。在每个episode开始之时，该算法会随机抽取一个Q函数，这个被抽出来的Q函数后面会用到。
* 中间层时共享层，共享层从全部数据中学习联合特征表示（joint data representation），可以视为data-dependent dropout。
* 最下层是输入的数据帧。（所有的sample保存在一个replay buffer中，每个replay buffer都需要记录它从哪个函数中来）

每个Q值都有一个bootstrap mask，该mask用于决定是否应该根据在时刻t产生的sample进行训练。每个独立的sample都被赋予一个随机mask的。

随后，当网络在replay buffer的minibatch上进行训练时，mask就决定了某个特定的Q函数是否要参与这个训练过程。

### 4.2 Incentivizing Exploration in RL with Deep Predictive model

通过在原有的reward function上增加Bonus的方法，刺激agent进行探索，从而提高-greedy探索策略。



## 5 改进网络结构

### 5.1 Deep Recurrent Q-learning for Partially Observable MDPS

因为DQN存在的两个缺陷，一个是内存限制，第二个在每个决策点需要依靠完整的游戏界面。为解决这两个问题，本文尝试用LSTM单元替换掉全连接层，这样就产生了Deep Recurrent Q-Network（DQRN），能处理部分观测的情况，处理信息丢失的问题。用部分状态进行训练，略完整的状态评估，DQRN的性能随评估图像的完整性而提高。用完整状态训练，部分状态评估，DQRN性能低于DQN。

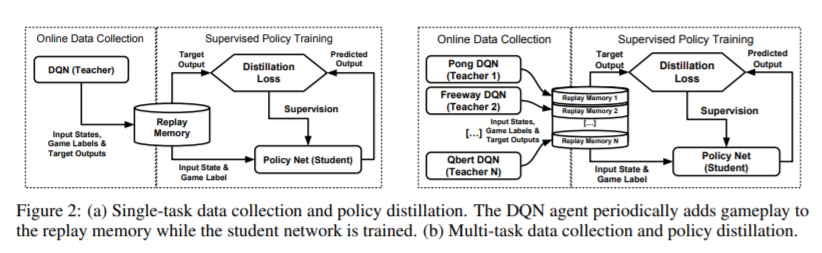
一些文字和图案

描述已自动生成

## 6 DQN的多任务和迁移学习

### 6.1 Policy Distillation

虽然DQN可以学习复杂视觉任务的策略，但是网络结果复杂需要大量训练。本文提出policy distillation的方法，利用较小的网络结构，更高的效率，使之前的任务达到专家的水平。此外，可以用相同的方法将多个特定于任务的策略整合为一个策略。



### 6.2 Actor Mimic Deep Multitask and Transfer Reinforcement Learning

本文解决的问题是如何把之前在多个任务上面学习到的知识迁移到一个单一的策略神经网络上。假设有N个source game，每个游戏都有一个训练好的DQN网络。现在需要训练一个策略网络，使得该策略网络能够在每一个游戏中都有比较好的表现。文中的方法是，先把这些DQN网络转化为一个相应的Boltzmann Policy，然后最小化这些policy和所需要得到的策略网络之间的差距。

## 7 加快DQN的训练速度

### 7.1 Asynchronous Methods for Deep Reinforcement Learning

Experience Replay存在一些缺点：每一次交互都会耗费更多的内存和计算；需要off-policy的学习算法从更老的策略中产生的数据上进行更新。本文提出不用experience replay，而是用异步的方法，并行执行多个agent，可以获取环境不同方面的信息。该框架可以实现On-Policy训练，减少计算资源。

## 8 DQN在连续控制上的应用

### 8.1 Continuous Deep Q-learning with NAF

提出了NAF算法，利用Q-learning和经验回放来处理连续的任务，并降低了深度强化学习中连续控制任务的样本复杂度。

