

实验 3. 强化学习实践

MG1733098, 周华平, zhp@smail.nju.edu.cn

2017 年 12 月 31 日

综述

实验二.

实验三.

DQN 实现

DQN 在基本的 Deep Q-Learning 算法的基础上使用了 Experience Replay 经验池, 通过将训练得到的数据储存起来然后随机采样的方法来降低数据样本的相关性, 进而提升了性能。

在本实验中, 我选择使用 PyTorch 来实现 DQN。在定义 Q 值网络时我使用了 MLP, 其中网络结构由 3 层 Linear 构成; 激活函数使用 PReLU, 同时每个隐层中增加 Batch Norm 来对相应的 activation 做规范化操作。另外, 在 DQN 中我采用 optim.Adam 作为优化函数, 用 nn.MSELoss() 来计算均方误差。其余的算法实现细节详见 myDQN.py。

Q 值网络的定义如下所示:

```
class DQN(nn.Module):
    def __init__(self, input_dim, output_dim, hidden_dim):
        super(DQN, self).__init__()
        self.layer1 = nn.Sequential(
            nn.Linear(input_dim, hidden_dim),
            nn.BatchNorm1d(hidden_dim),
            nn.PReLU(),
        )
        self.layer2 = nn.Sequential(
            nn.Linear(hidden_dim, hidden_dim),
            nn.BatchNorm1d(hidden_dim),
            nn.PReLU(),
        )
        self.out = nn.Linear(hidden_dim, output_dim)
```

```

def forward(self, x):
    x = self.layer1(x)
    x = self.layer2(x)
    return self.out(x)

```

DQN 训练

DQN 的超参数设置如表 1 所示。其中 ϵ 的衰减公式同公式 (abcd)。

表 1: DQN 超参数设置

超参数	参数意义	CartPole	MountainCar	Acrobot
memory_size	Replay Memory 的大小	10000	10000	5000
batch_size	mini-batch 的大小	128	128	128
hidden_dim	DQN 的隐层维度	50	50	50
discount	DQN 算法中的 γ	0.99	0.99	0.99
learning_rate	DQN 算法中的 α	0.001	0.001	0.001
eps_start	ϵ 的初始值	0.9	0.9	0.9
eps_end	ϵ 的结束值	0.05	0.05	0.05
eps_decay	ϵ 的衰减权重	200	50	200

DQN 在 CartPole 上的实验结果如图 1 所示。可以观察到 Loss 在超过 450 轮后达到收敛的状态。由于 ϵ 的最小值被设置为 0.05，因此即使 Training 了较多轮数，DQN 依旧会以 5% 的概率随机选择 Action。而 CartPole 似乎对于错误的 Action 比较敏感，当随机到错误的 Action 时，可能会导致该轨迹提前结束。因此在 Training 阶段 Reward 似乎并没有收敛到一个固定值，然而我们可以观察到随着 Training 轮数的增加，Reward 的上限也在不断提高，这也从侧面体现出了训练是有效果的。

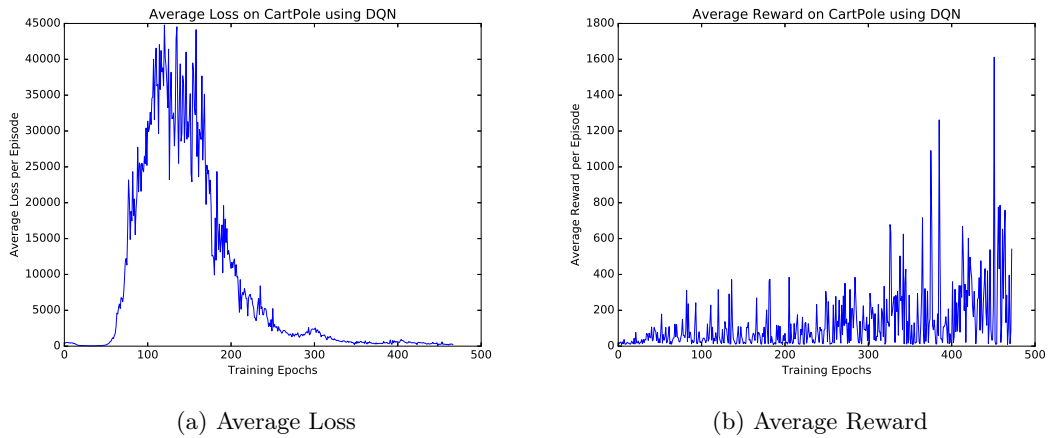


图 1: Training Result of CartPole using DQN

DQN 在 MountainCar 上的实验结果如图 2所示。其中 Reward 在超过 200 轮之后达到收敛的状态，而 Loss 也在超过 200 轮之后达到了基本稳定的状态。

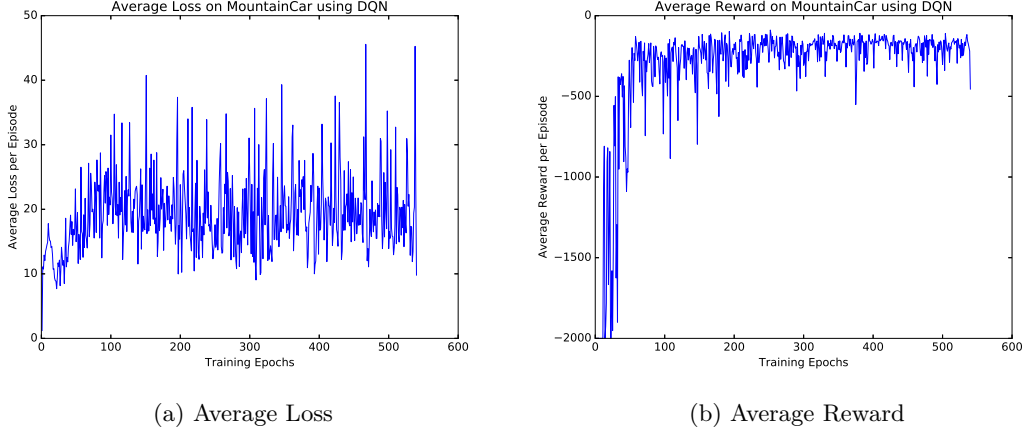


图 2: Training Result of MountainCar using DQN

DQN 在 Acrobot 上的实验结果如图 3所示。在超过 40 轮之后，Loss 达到了较低的水平，并且 Reward 也趋近于收敛。

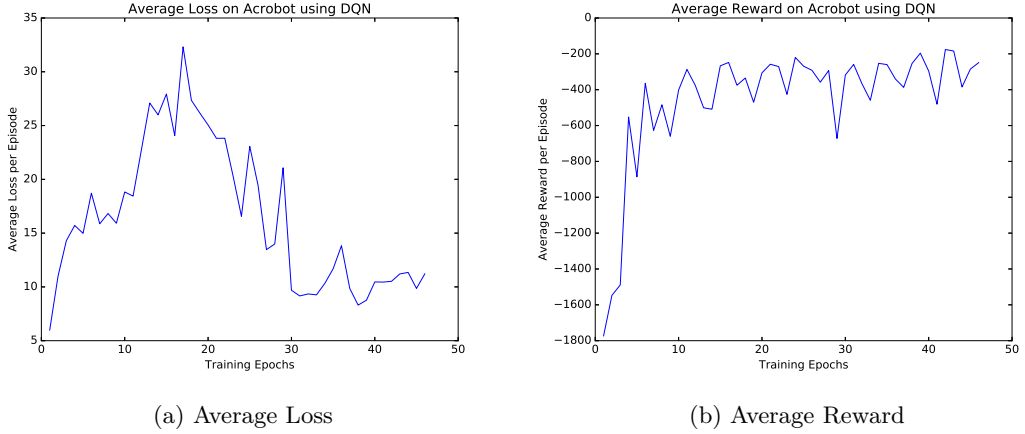


图 3: Training Result of Acrobot using DQN

对于每个任务，我将多次训练中得到的最好的策略进行测试。对于 CartPole 我测试了 50 条轨迹，对于 MountainCar 和 Acrobot 我分别测试了 100 条轨迹。最终得到 reward 的均值和标准差如表 2所示。

表 2: DQN Reward 结果

	CartPole	MountainCar	Acrobot
$mean \pm std$	20000 ± 0	$-183.44 \pm 26.6669533318$	-86.84 ± 18.1117199625

实验四.

Improved DQN 实现

相比于 DQN, Improved DQN 进行了一个微小的改动: 增加了目标 Q 网络 \hat{Q} 。在计算目标 Q 值时, 我们使用专门的目标 Q 网络 \hat{Q} 来计算, 而不是直接使用预更新的 Q 网络 Q 。这样做的目的是为了减少目标计算与当前值的相关性。

在 DQN 中, 目标 Q 网络会随着 Q 的更新而动态变化, 这样不利于计算目标 Q 值, 导致目标 Q 值和当前的 Q 值相关性较大, 因此 Improved DQN 提出单独使用一个目标 Q 网络。而 \hat{Q} 中的参数是通过延迟更新的方式从 Q 中获得: 在每训练了 C 步之后, Improved DQN 将当前 Q 的参数值复制给 \hat{Q} 。

Improved DQN 的实现基于 DQN, 因此代码部分的改动比较少。另外在 Improved DQN 中需要新增一个超参数 `target_c`, 用于表示 \hat{Q} 的更新频率。

Improved DQN 训练

Improved DQN 的超参数设置如表 3 所示。

表 3: Improved DQN 超参数设置

超参数	参数意义	CartPole	MountainCar	Acrobot
<code>memory_size</code>	Replay Memory 的大小	10000	10000	10000
<code>batch_size</code>	mini-batch 的大小	128	128	128
<code>hidden_dim</code>	DQN 的隐层维度	50	50	50
<code>target_c</code>	\hat{Q} 的更新频率	10	10	5
<code>discount</code>	DQN 算法中的 γ	0.99	0.99	0.99
<code>learning_rate</code>	DQN 算法中的 α	0.001	0.001	0.001
<code>eps_start</code>	ϵ 的初始值	0.9	0.9	0.9
<code>eps_end</code>	ϵ 的结束值	0.05	0.05	0.05
<code>eps_decay</code>	ϵ 的衰减权重	200	50	200

Improved DQN 在 CartPole 上的实验结果如图 4 所示。可以观察到 Loss 在超过 300 轮后达到收敛的状态。

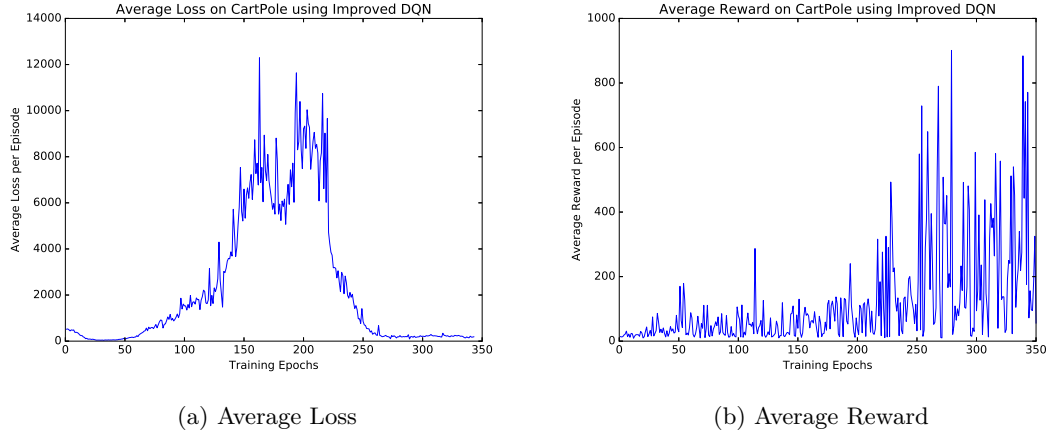


图 4: Training Result of CartPole using Improved DQN

Improved DQN 在 MountainCar 上的实验结果如图 5所示。其中 Reward 在超过 60 轮之后达到收敛的状态，而 Loss 也在超过 60 轮之后达到了基本稳定的状态。

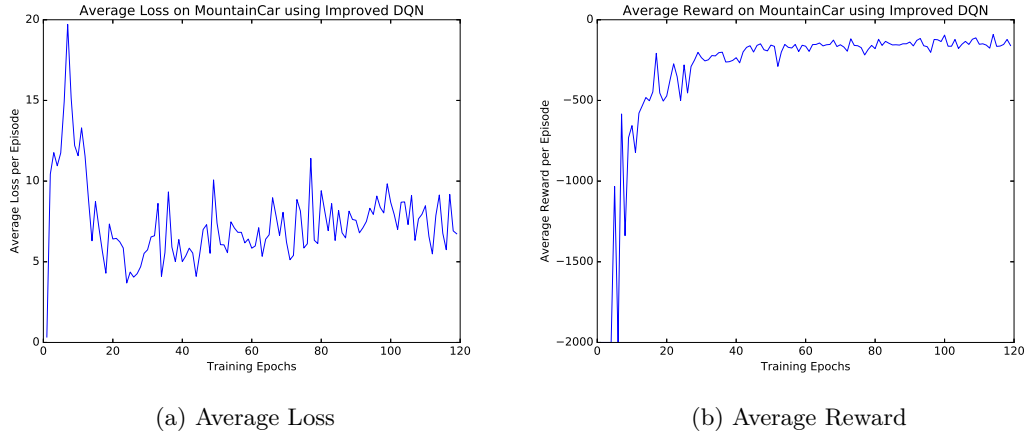


图 5: Training Result of MountainCar using Improved DQN

Improved DQN 在 Acrobot 上的实验结果如图 6所示。在超过 50 轮之后，Loss 达到了较低的水平，并且 Reward 也趋近于收敛。

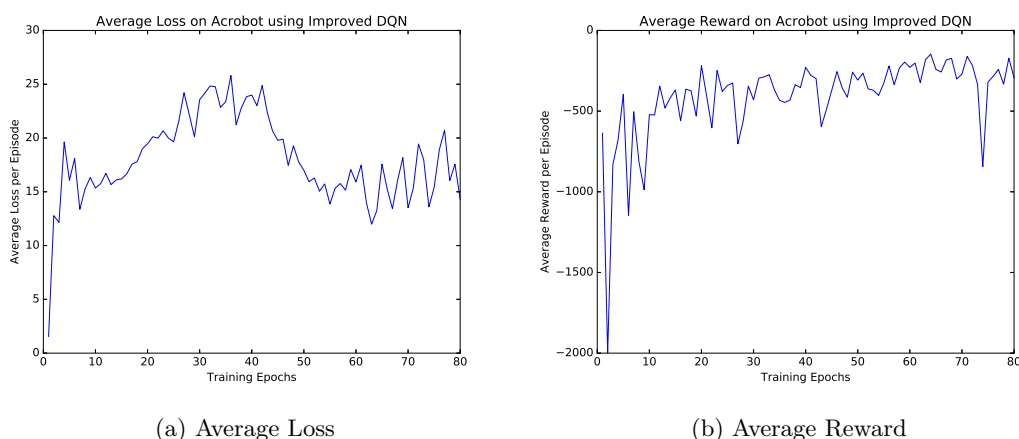


图 6: Training Result of Acrobot using Improved DQN

对于每个任务，我将多次训练中得到的最好的策略进行测试。对于 CartPole 我测试了 50 条轨迹，对于 MountainCar 和 Acrobot 我分别测试了 100 条轨迹。最终得到 reward 的均值和标准差如表 4 所示。

表 4: Improved DQN Reward 结果

	CartPole	MountainCar	Acrobot
$mean \pm std$	20000 ± 0	$-107.54 \pm 13.5273205033$	-84.31 ± 18.8024971746

Improved DQN 与 DQN 的异同

在训练过程中可以明显观察到 Improved DQN 相比于 DQN 表现得更加稳定，收敛速度也更快。以 MountainCar 为例，在使用 DQN 时，很多时候并没有办法训练出一个可行的模型：经过几十轮的训练，每一轮的 Reward 依旧为 -2000。而在前一节中得到的 DQN 在 MountainCar 上的比较好的训练结果似乎带有一定的偶然成分；相比之下，Improved DQN 在 MountainCar 上几乎每次都能训练出一个可行的模型：在前十轮的训练中，Reward 就能突破 -2000，并且一旦突破了 -2000 以后，Reward 能够在之后的每一轮训练中快速增加，直至收敛。

在训练效果上，除了 MountainCar 以外，DQN 和 Improved DQN 在 CartPole 和 Acrobot 上的训练效果比较接近。而 DQN 之所以在 MountainCar 上的训练结果与 Improved DQN 差别较大，我认为主要是由于 DQN 训练出稳定的 MountainCar 模型比较困难：如果能够用 DQN 训练足够多的模型，其中最好的那个模型的效果应该和 Improved DQN 上得到的结果相近。

所以我认为 Improved DQN 在训练阶段比 DQN 表现得更加稳定，收敛速度也更快；而在训练模型收敛后，DQN 与 Improved DQN 的训练效果应该还是比较接近的。