

RLChina第三次习题课参考教程

题目要求:

动手:

1 训练DQN:

2 提交

submission.py

算法理解

相关文章:

## RLChina第三次习题课参考教程

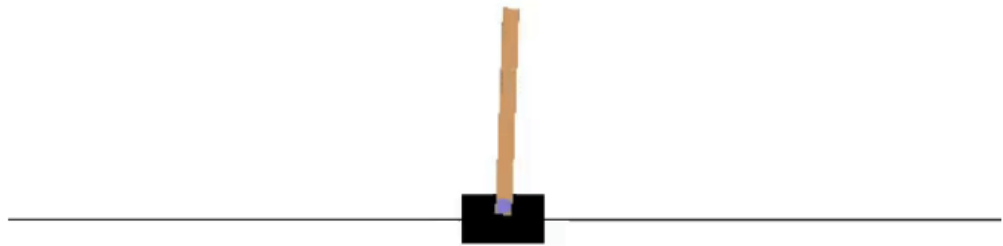
---

嘻嘻~~ 我又来了~

这次的作业和第二次基本一致

游戏的环境使用的状态空间连续的场景:

及第科目 <http://www.jidiai.cn/cartpole>



换了一个算法, 算法使用适合连续状态空间的DQN算法

### 题目要求:

作业的目的:

理解DQN算法并且动手训练提交

作业要求

- 训练车杆游戏的DQN算法
- 将homework里的submission.py填写完整
- 将submission.py, critic.py, critic\_\*.pth提交到及第平台

### 动手:

下载和第二次作业一样不说了, 直接到训练 上传吧

然后本文将给出提交代码和DQN的讲解以及其衍生算法

## 1 训练DQN:

### 方法1: 直接修改

```
if __name__ == '__main__':
    # set env and algo
    parser = argparse.ArgumentParser()
    # parser.add_argument('--scenario', default="gridworld", type=str)
    parser.add_argument('--scenario', default="classic_CartPole-v0", type=str)
    parser.add_argument('--algo', default="dqn", type=str,
                        help="tabularq/sarsa/dqn/ppo/ddpg/ac/ddqn/duelingq/sac/pg/sac/td3")

    parser.add_argument('--reload_config', action='store_true') # 加是true; 不加为false
    args = parser.parse_args()

    print("===== args: ", args)
```

### 方法2: 命令行(我没试 应该行)

在main.py的目录下执行(course3的啊)

```
1 python main.py --scenario classic_CartPole-v0 --algo dqn
```

```
1 # 打印部分结果
2 i_epoch: 495 Gt: 200.00
3 i_epoch: 496 Gt: 200.00
4 i_epoch: 497 Gt: 200.00
5 i_epoch: 498 Gt: 200.00
6 i_epoch: 499 Gt: 154.00
7 i_epoch: 500 Gt: 114.00
```

### Tensorboard可视化训练过程

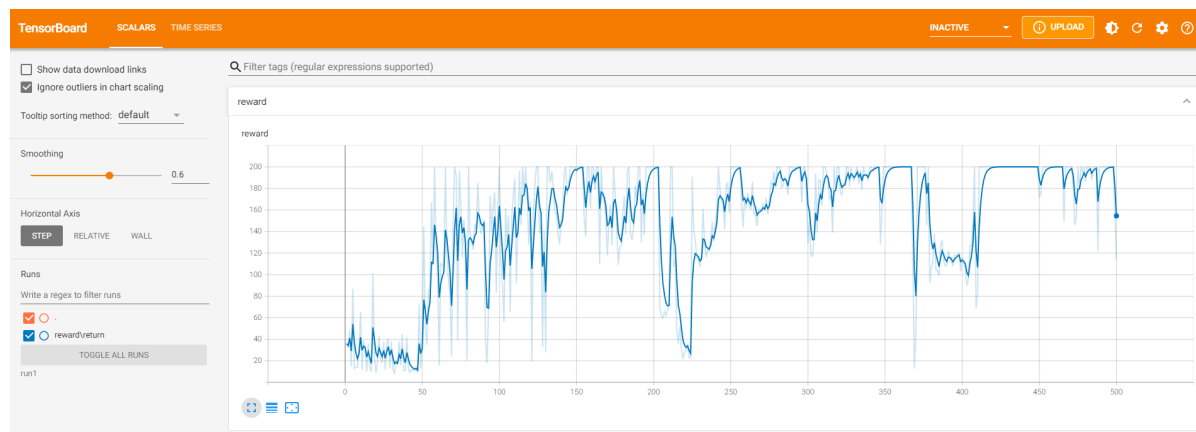
在这个路径下执行

```
1 tensorboard --logdir run1
```

```
SummerCourse2021\course3\examples\models\classic_CartPole_v0\dqn>tensorboard --logdir run1
TensorFlow installation not found - running with reduced feature set.
Serving TensorBoard on localhost; to expose to the network, use a proxy or pass --bind_all
TensorBoard 2.6.0 at http://localhost:6006/ (Press CTRL+C to quit)
```

复制返回的链接到网页打开

可以看到训练趋于稳定



这里有个问题:

选择模型的时候。是要最终的模型好，还是在前面一点选择能拿到200分的模型好。。。。

关于这个问题的讨论，我后面做个考虑，先完成一下作业

## 2 提交

还是和第二次作业一样，但是这次把critic 智能体分开放了。

最后提交三个文件即可(submission.py 自己写的 不正确 可以去参考附录)

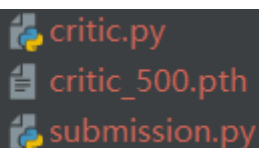
把这部分完成即可：就是把学习的部分扔掉，只考虑执行部分

```
# TODO: Complete DQN algo under evaluation.
class DQN:
    def __init__(self):
        pass

    def choose_action(self, observation):
        pass

    def load(self, file):
        pass
```

最后要提交的文件是这么几个, homework下(没有的话去github找最新版)



critic.py  
critic\_500.pth  
submission.py

- critic.py 是保存的神经网络，被submission.py 调用
- pth文件是训练完的模型
- submission.py 是要自己写的哈

之前之所以没有想着发第三次作业，是想着这次作业和第二次基本差不多

但是从后台提问的同学来看，可能其实大家是对如何如何写submission函数不是很清楚。

其实蛮简单，但是逻辑不太好用图文表达。不然我在**公众号发起一个投票**吧，有超过10人想知道这里逻辑怎么写怎么调试的同学，我就开个直播给大家稍稍讲一下。(害羞。。)

投票链接：<https://mp.weixin.qq.com/s/FwEVjAfjhBbafxatamFdZw>

## submission.py

```
1  ## This is homework.
2  ## Load your model and submit this to Jidi
3
4  import torch
5  import os
6
7  # load critic
8  from pathlib import Path
9  import sys
10 base_dir = Path(__file__).resolve().parent
11 sys.path.append(str(base_dir))
12 from critic import Critic
```

```

13
14
15 # TODO: Complete DQN algo under evaluation.
16 class DQN:
17     def __init__(self):
18         # pass
19         self.state_dim = 4
20         self.action_dim = 2
21
22         self.hidden_size = 64
23         self.critic_eval = Critic(self.state_dim, self.action_dim,
self.hidden_size)
24     def choose_action(self, observation):
25         # pass
26         observation = torch.tensor(observation, dtype=torch.float).view(1,
-1)
27         action = torch.argmax(self.critic_eval(observation)).item()
28         return action
29
30     def load(self, file):
31         # pass
32         self.critic_eval.load_state_dict(torch.load(file))
33
34
35 def action_from_algo_to_env(joint_action):
36     joint_action_ = []
37     for a in range(n_player):
38         action_a = joint_action
39         each = [0] * action_dim
40         each[action_a] = 1
41         joint_action_.append(each)
42     return joint_action_
43
44
45 n_player = 1
46 state_dim = 4
47 action_dim = 2
48 hidden_size = 64
49
50 # TODO: Once start to train, u can get saved model. Here we just say it is
critic.pth.
51 critic_net = os.path.dirname(os.path.abspath(__file__)) + '/critic_500.pth'
52 agent = DQN()
53 agent.load(critic_net)
54
55
56 # This function dont need to change.
57 def my_controller(observation, action_space, is_act_continuous=False):
58     obs = observation['obs']
59     action = agent.choose_action(obs)
60     return action_from_algo_to_env(action)

```

这次内容不多，可以顺道把算法实现提一嘴，完事也是建议大家可以好好看看

## 算法理解

DQN算法部分，在整个代码中其实就是这两个文件

- `course3\examples\algo\dn\dn.py` : 算法核心实现
- `course3\examples\networs\critic.py` : 神经网络

从submission也可以看到，如果训练完成后，用来给输入一个动作输出的code也很少

最主要的就是

- 神经网络如何更新？
- 本算法中引入了什么训练技巧？

这个时候，我们需要理解的代码量就非常小了

**首先第一个问题：神经网络如何更新？**

- 价值更新：最小化均方误差MSE

$$\theta \leftarrow \arg \min_{\theta} \frac{1}{N} \sum_{i=1}^N \left[ Q_{\theta}(s_i, a_i) - \left( r_i + \gamma \max_{a'} Q_{\theta}(s'_i, a') \right) \right]^2$$

**引入了什么训练技巧？**

- 优化1：经验回放Experiment Replay
  - 将环境采样数据 $\langle s, a, r, s' \rangle$ 存放在回放池
  - 每次训练时从回放池中随机采样
  - 作用：增强样本独立性；提高样本利用率
- 优化2：目标网络Target Network
  - 增加一套目标网络，与原训练网络结构相同但使用较旧参数 $\theta'$
  - $\theta \leftarrow \arg \min_{\theta} \frac{1}{N} \sum_{i=1}^N \left[ Q_{\theta}(s_i, a_i) - \left( r_i + \gamma \max_{a'} Q_{\theta'}(s'_i, a') \right) \right]^2$
  - 每隔若干步，同步两个网络参数 $\theta' \leftarrow \theta$
  - 作用：使训练相对稳定

对应代码：

```
main.py x dqn.py x
74 self.memory.insert(k, agent_id, v)
75
76 def learn(self):
77     data_length = len(self.memory.item_buffers["rewards"].data)
78     if data_length < self.buffer_size:
79         return
80
81     data = self.memory.sample(self.batch_size)
82
83     transitions = {
84         "o_0": np.array(data['states']),
85         "o_next_0": np.array(data['states_next']),
86         "r_0": np.array(data['rewards']).reshape(-1, 1),
87         "u_0": np.array(data['action']),
88         "d_0": np.array(data['dones']).reshape(-1, 1),
89     }
90
91     obs = torch.tensor(transitions["o_0"], dtype=torch.float)
92     obs_ = torch.tensor(transitions["o_next_0"], dtype=torch.float)
93     action = torch.tensor(transitions["u_0"], dtype=torch.long).view(self.batch_size, -1)
94     reward = torch.tensor(transitions["r_0"], dtype=torch.float).squeeze()
95     done = torch.tensor(transitions["d_0"], dtype=torch.float).squeeze()
```

优化1: 经验回放

```
# 一直更新的网络最新学习的Q
q_eval = self.critic_eval(obs).gather(1, action) # 将对应动作的q 取出

# 旧的目标网络预测的Q
q_next = self.critic_target(obs_).detach() # 隔离关系 更新critic_eval的时候 不更新critic_target
q_target = (reward + self.gamma * q_next.max(1)[0] * (1 - done)).view(self.batch_size, 1)
# 可以对比下公式看看

# 新网络的预估 和 目标网络的预测差值
loss_fn = nn.MSELoss()
loss = loss_fn(q_eval, q_target)

self.optimizer.zero_grad() # 看明白loss更新哪个网络, 不更新哪个网络 这个技巧就明白了
loss.backward() # 因为目标网络detach()了 所以不更新, 只更新q_eval
self.optimizer.step()

# 当学习了target_replace_iter = 100 次后 将critic_eval参数给目标网络复制一份 -- 硬更新
if self.learn_step_counter % self.target_replace_iter == 0:
    self.critic_target.load_state_dict(self.critic_eval.state_dict())
```

优化2: 目标网络

至于Critic的具体网络结构: 这个大家可以去叠层, 调整隐藏层的个数, 去看看学习效果怎么改变。

大家感兴趣的话, 我也可以把其他几个优化的DQN的算法一起拿出来对比一下。直观比较下差别(可以push我一下)

OK ,That's all

啊 不对, 这个结果不是很完美, 最高可以到200的。后期优化的时候感觉有价值的话我就贴到号里叭, 这里先不说了(因为还没做, (逃~))

>	车杆	dqn	114.20	2021-08-19 15:30:12	通过	山	↓
---	----	-----	--------	---------------------	----	---	---

作者: HandsomeWu(公众号同步)



投票链接: <https://mp.weixin.qq.com/s/FwEVjAfjhBbafxatamFdZw>

也欢迎关注公众号: **RLCN** 在后台提问:



当然 有问题最好是大家在论道板块一起讨论哦~~

**及第**

金榜

科目

秘籍

擂台

论道

赶考

论道 - 及第: <http://www.jidiai.cn/forumlist>

## 相关文章:

(没有的话找我要哈 群里应该都有)

RLChina第一次习题课参考教程

RLChina第二次习题课参考教程

RLChina2021-习题课3 -- 林舒 中科院自动化研究所