

作业内容

实现Model-based Q-learning算法

作业描述

环境描述

本次作业使用和作业2一样的环境为网格世界(gridworld)，玩家可以通过选择动作来移动人物，走到出口。唯一的区别在于输出的状态包括了额外的一维特征，表示agent是否拿到了钥匙。agent需要先拿到钥匙（坐标在 (0,7) ），然后走到出口才算通关。

###实验描述

实验探究1：实现Dyna-Q 算法，并通过调节参数找到算法可提升的性能极限。

伪代码如下：

```
Require: initialized  $Q(s,a)$  and  $Model(s,a)$  for all  $s \in S$  and  $a \in A$ . A given
reward function  $R$  and terminal function  $D$ .
s = env.reset()
While True:
    while not done:
        a = epsilon-greedy(s,Q)
        s',r, done = env.step(a)
        update Q:  $Q(s,a) \leftarrow \alpha [r + (1 - done) * \gamma \max_{a'} Q(s',a')] - Q(s,a)$ 
        update Model:  $Model(s,a) \leftarrow s'$ 
        s = s'
        if done:
            s = env.reset()
    repeat n times:
        s_m = random previously observed state
        a_m = random action previously taken in s_m
        s'_m = Model(s_m,a_m)
        r_m = R(s_m, a_m)
        update Q:  $Q(s_m,a_m) \leftarrow \alpha [r_m + (1 - done) * \gamma \max_{a'} Q(s'_m,a'_m)] - Q(s_m,a_m)$ 
```

实验要求：

1. 写完代码后，请从 $n=0$ 开始（即纯 model-free 方法），尝试调试不同的参数 n ，记录算法的收敛时

间，和所消耗的样本量。得出一个经验性的 n^* 的粗略估计，表示若 n 的取值 $n > n^*$ 算法收敛所消耗的样本量不再有明显的下降。

2. 请在实验报告中展示你所尝试的参数和对应的实验结果。

Note:

1. 由于环境的转移是确定性的，Model 也可以用table 来进行记录和更新
2. policy 的学习部分，可以使用你在HW2中的实现

实验探究2：用神经网络来预测环境Model， 实现简单的Model-based 算法，完成以下三个探究问题

伪代码如下：

```
initialize Q(s,a) and Model(s,a) for all s \in S and a \in A. A given reward
function R and terminal function D.
s = env.reset()
for iter in T:
    while not done:
        a = epsilon-greedy(s,Q)
        s',r = env.step(a)
        update Q: Q(s,a) = Q(s,a) + \alpha [r + (1 - done) * \gamma max_{a'}
Q(s',a') - Q(s,a)]
        s = s'
    if done:
        s = env.reset()

    repeat m times:
        Model.train_transition()
    if iter > start_planning:
        repeat n times:
            s_m = random previously observed state
            repeat h times:
                a_m = epsilon-greedy(s_m,Q)
                s'_m = Model(s_m,a_m)
                r_m = R(s_m, a_m, s'_m)
                done = D(s_m, a_m, s'_m)
                update Q: Q(s_m,a_m) = Q(s_m,a_m) + (1 - done) * \alpha [r_m +
\gamma max_{a'} Q(s'_m,a'_m) - Q(s_m,a_m)]
            if done:
                break
```

实验1：算法调试

1. 该实验的Model 相关接口及其实现已经写好，调节算法的参数，寻找你能找到的达到最好效果的参数组合
 1. n (采样的轨迹条数),
 2. start_planning (开始使用model based 提高样本利用率),

3. h (一条轨迹执行的长度)
 4. m (转移训练的频率)
 5. ... 其他你发现的有影响的参数
2. 请在实验报告中展示你所尝试的有显著差异的参数组合和实验结果

实验2：改进算法

改进1：尝试改进Model的学习流程，强化对稀疏/奖励变化相关的数据的学习，可参考下面的代码：

```
def store_transition(self, s, a, r, s_):
    s = self.norm_s(s)
    s_ = self.norm_s(s_)
    self.buffer.append([s, a, r, s_])
    # 新增部分
    if s[-1] - s_[-1] != 0:
        self.sensitive_index.append(len(self.buffer) - 1)

def train_transition(self, batch_size):
    s_list = []
    a_list = []
    r_list = []
    s_next_list = []
    for _ in range(batch_size):
        idx = np.random.randint(0, len(self.buffer))
        s, a, r, s_ = self.buffer[idx]
        s_list.append(s)
        a_list.append([a])
        r_list.append(r)
        s_next_list.append(s_)
    # 新增部分
    if len(self.sensitive_index) > 0:
        for _ in range(batch_size):
            idx = np.random.randint(0, len(self.sensitive_index))
            idx = self.sensitive_index[idx]
            s, a, r, s_ = self.buffer[idx]
            s_list.append(s)
            a_list.append([a])
            r_list.append(r)
            s_next_list.append(s_)
    x_mse = self.sess.run([self.x_mse, self.opt_x], feed_dict={
        self.x_ph: s_list, self.a_ph: a_list, self.x_next_ph: s_next_list
    })[:1]
    return x_mse
```

改进2：对策略的学习过程做额外的约束：

```
Q(s,a) = Q(s,a) + \alpha [r + (1 - done) * \gamma max_{a'} Q(s',a') - Q(s,a)]
Q(s,a) = np.clip(Q(s,a), -100, 100)
```

分别尝试两个改进，重新调节该探究问题 中实验1 的参数组合，最优的参数和对应的性能是否发生变化？若有变化，发生了什么变化

(Optional)：可以尝试其他任意的改进，并展示你的改进带来的性能提升

实验探究3：根据以上实验进行分析

根据上面的实验回答以下两个问题（开放问题）

1. 根据上面实验，试讨论不同模型学习方式（table 和 neural network），不同参数对实验结果的影响和背后的原因，从而分析影响model-based 的算法的性能的因素由哪些？有以下两条参考建议
 1. 可打印学习过程的以下中间指标辅助进行分析：Q函数学习情况如何？策略表征如何？模型在各个状态的预测准确度如何？
 2. 可回顾老师上课提到的Model-based 相关的三个问题进行思考，即：how to learn the model efficiently? how to update the policy efficiently? how to combine model learning and policy learning?
2. 回顾HW3的DQN中的replay buffer设置和前面的Dyna-Q 实验，你觉得这两者有什么联系？

提交方式

完成的作业请通过sftp上传提交。上传的格式为一份压缩文件，命名为'学号+姓名'的格式，例如'MG21370001张三.zip'。文件中需包含 'main.py', 'arguments.py', 'algo.py', 'env.py', 'performance.png' 和'Document.pdf'（一份pdf格式的说明文档）

文档模板参见'Document4.tex'和'Document4.pdf'。（也可以使用自己的模板。）