强化学习：作业二

李龙飞 DZ20370003

2020年11月20日

1 作业内容

在gridworld环境中实现Q-learning算法。

2 实现过程

2.1 预处理阶段

1. 对每个二维位置赋予唯一的状态s用于构建Q值表，我们采取的方法是令$s = pos[0] * grid_num + pos[1]$。
2. 构造Q值表，我们采用了字典结构，如果查询的Q(s,a)未在字典里出现过，直接返回0，这样可以极大的减小Q值表的存储开销，同时也提高了查询效率。

2.2 训练阶段

1. 将Q值表所有值初始化为0
2. for $i = 0, 1, \ldots, T$ do
   (a) 用ϵ贪婪在当前状态s选择动作a, 即$a = \pi_\epsilon(s)$
   (b) 在状态s执行动作a,得到新状态s'和奖励r,即$s', r =$ do action a
   (c) 更新价值函数$Q(s, a)$,即

$$Q(s,a) += \alpha \left(r + \gamma \max_a Q\left(s', a\right) - Q(s,a) \right)$$

   (d) 更新当前状态，即$s = s'$
3. end for

2.3 测试阶段

由于Q-learning算法是off-policy的算法，因此我们在测试阶段关闭了ϵ贪婪的探索，直接使用训练得到的策略,即$\pi(s) = \arg\max_a Q(s, a)$。

3 复现方式

在主文件夹下运行 `python main.py`.

4 实验效果

从图中我们可以看到平均奖励随训练轮数逐渐增加且能够很快收敛到最大值，同时我们对比了测试阶段是否带ϵ的探索对实验结果的影响，可以看出在测试阶段不带ϵ探索，算法会收敛的更好，波动更小。

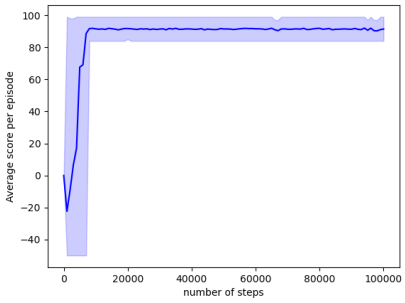

Figure 1: 不带ϵ探索

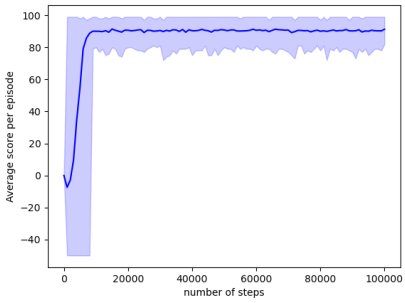

Figure 2: 带ϵ探索

5 小结

在这次实验中，我发现Q-learning和SARSA算法的一些区别和联系：

- Q-learning算法是off-policy的策略，而SARSA在on-policy的策略，因此SASA算法即使在测试阶段ϵ探索也无法关闭，而Q-learning在测试阶段可以将探索关闭，从而提高算法稳定性。
- Q-learning相较SARSA算法更加大胆，每次在更新的时候选取的是最大化Q的方向，而当下一个状态时，再重新选择动作，而SARSA选取的是一种保守的策略，他在更新Q值的时候已经为未来规划好了动作，对错误和死亡比较敏感。这两种算法在实际应用中各有优劣。
- Q-learning和SARSA算法依赖于Q值表的估计，这在小型强化学习问题是非常有效的，但当状态空间和动作空间过大时，Q值表会非常大，甚至超出内存，这一问题限制了时序差分算法的应用场景。