实验三-多臂老虎机

潘林朝

强化学习

与其他机器学习方法相比，RL更注重从互动中进行目标导向学习。强化学习是学习做什么——如何将情境映射到动作——以便最大化数字奖励信号。学习者不会被告知要采取哪些动作，而是必须通过尝试发现哪些动作产生的奖励最大。试错搜索和和延迟奖励这两个特征是强化学习的两个最重要的区别性特征。
强化学习的要素：Agent、环境、策略、奖励信号、值函数。

- 策略：从感知的环境状态到处于这些状态时要采取的动作的映射；
- 奖励信号：奖励信号定义了强化学习问题的目标。在每个时间步，环境都会向强化学习 agent 发送一个称为奖励的单个数值。 Agent 的唯一目标是使其长期获得的总奖励最大化；
- 值函数：奖励信号表明什么是即时好的，而值函数则指明什么是长期好的；

多臂老虎机(Multi-Armed Bandits)

考虑以下学习问题。您反复面临着 k 个不同选项或动作中的一个选择。在每一次选择之后，您会得到一个从平稳概率分布中选出的数值奖励，该奖励取决于您选择的动作。您的目标是在某个时间段内最大化期望的总奖励，例如，超过 1000 个动作选择或时间步[1]。

问题定义

多臂老虎机没有涉及状态转移（概率分布是固定的），是一种特殊的强化学习问题。在我们的 k 臂赌博机问题中，只要选择了该动作， k 个动作中的每个动作都有期望或平均的奖励；称为动作的价值。一般而言，时刻t的动作A_t得到的奖励R_t的期望称为价值(value)。若选择了动作a，则任意动作a的值$q_*(a)$表示该动作下，获得的奖励的期望，即价值。

$$q_*(a) = \mathbb{E}[R_t|A_t = a]$$

采取一些策略π映射的一系列动作a，最大化上述的$q_*(a)$。如果每个动作奖励的期望（概率分布）已知，那这个问题将迎刃而解，每次都选那个期望最大的就行了。

操作流程（重新理解）

你进了一家赌场，假设面前有 K 台老虎机（arms）。我们知道，老虎机本质上就是个运气游戏，我们假设每台老虎机 i 都有一定概率 p_i 吐出一块钱，或者不吐钱（概率$1-p_i$）。假设你手上只有 T 枚代币（tokens），而每摇一次老虎机都需要花费一枚代币，也就是说你一共只能摇 T 次，那么如何做才能使得期望回报（expected reward）最大呢？[2]

贪心求解

根据已经选择的估计的期望，选择最大期望的那个老虎机。

$$A_t = \arg\max_a Q_t(a),$$

其中$Q_t(a)$是在前t次对动作a价值的估计，即奖励的期望。但这样会有问题，估计的期望需要探索出来得到准确的，也即如果期望估计的不准确，那后续贪心选择最大也无法近似最优。这里存在利用和探索的平衡问题。
估计伪代码如下：

Algorithm – Abstract multi-armed bandit

Input: Multi-armed bandit problem $M = \langle \{X_{i,k}\}, A, T \rangle$
Output: Q-function Q

$Q(a) \leftarrow 0$ for all arms $a \in A$
$N(a) \leftarrow 0$ for all arms $a \in A$

$k \leftarrow 1$
while $k \leq T$
    $a \leftarrow \text{select}(k)$
    Execute arm a for round k and observe reward $X_{a,k}$
    $N(a) \leftarrow N(a) + 1$
    $Q(a) \leftarrow Q(a) + \frac{1}{N(a)}[X_{a,k} - Q(a)]$
    $k \leftarrow k + 1$

其中，N(a)表示对动作a（对应某个老虎机）的执行次数，对Q(a)的估计是求和平均的增量式写法，减少内存消耗。而贪心算法的应用在于，$a \leftarrow select(k)$语句选择估计价值最大的。

ϵ-贪心求解意味着每次有ϵ的概率会随机挑选，有$1-\epsilon$的概率选择最大的。

> 但是ϵ-贪心也存在问题...

实验结果/结论

1. 画出迭代过程中收益和懊悔(regret)的变化；
2. 探讨不同的参数对算法的影响；
3. ...

懊悔定义为拉动当前拉杆的动作a与最优拉杆的期望奖励差。通过对懊悔的分析，可以得到算法获得的收益与最佳收益之间的差距。

示例代码

多臂老虎机的定义可以是[3]：

```python
class BernoulliBandit(Bandit):

    def __init__(self, n, probas=None):
        assert probas is None or len(probas) == n
        self.n = n
        if probas is None:
            np.random.seed(int(time.time()))
            self.probas = [np.random.random() for _ in
range(self.n)]
        else:
            self.probas = probas

        self.best_proba = max(self.probas)

    def generate_reward(self, i):
        # The player selected the i-th machine.
        if np.random.random() < self.probas[i]:
            return 1
        else:
            return 0
```

可扩展的地方

1. 使用上置信界、汤普森采样等算法求解多臂老虎机；
2. 考虑泛化情况，比如组合多臂老虎机问题(CMAB)；
3. 在非平稳多臂老虎机情况下求解；

1. https://github.com/YunlianMoon/reinforcement-learning-an-introduction-2nd-edition/tree/master↩
2. https://www.zhihu.com/question/53381093/answer/562235053↩
3. https://github.com/lilianweng/multi-armed-bandit/blob/master/bandits.py↩