强化学习总结

强化学习的故事

强化学习是学习一个最优策略(policy)，可以让本体(agent)在特定环境(environment)中，根据当前的状态(state)，做出行动(action)，从而获得最大回报(G or return)。

有限马尔卡夫决策过程

马尔卡夫决策过程理论定义了一个数学模型，可用于随机动态系统的最优决策过程。
强化学习利用这个数学模型将一个现实中的问题变成一个数学问题。
强化学习的故事1：找到最优价值

> 强化学习就是：追求最大回报G
> 追求最大回报G就是：找到最优的策略π_*。
> 策略π_*告诉在状态s，应该执行什么行动a。
> 最优策略可以由最优价值方法$v_*(s)$或者$q_*(s,a)$决定。

故事1的数学版

$$
\begin{aligned}
&\text{Reinforcement Learning} \doteq \pi_* \qquad (1)\\
&\quad \updownarrow \\
&\pi_* \doteq \{\pi(s)\},\ s \in \mathcal{S} \\
&\quad \updownarrow \\
&\begin{cases} \pi(s) = \underset{a}{argmax}\ v_\pi(s'|s,a),\ s' \in S(s), \quad \text{or} \\ \pi(s) = \underset{a}{argmax}\ q_\pi(s,a) \end{cases} \\
&\quad \updownarrow \\
&\begin{cases} v_*(s), \quad \text{or} \\ q_*(s,a) \end{cases} \\
&\quad \updownarrow \\
&\text{approximation cases:} \\
&\begin{cases} \hat{v}(s,\theta) \doteq \theta^T \phi(s), \quad \text{state value function} \\ \hat{q}\,(s,a,\theta) \doteq \theta^T \phi(s,a), \quad \text{action value function} \end{cases} \\
&where \\
&\theta \text{ - value function's weight vector}
\end{aligned}
$$

有限马尔卡夫决策过程的基本概念：

state 状态
action 行动
reward 奖赏
G_t 回报
$p(s'|s,a)$ 表示在状态s下，执行行动a，状态变成s'的可能性。
$p(s',r|s,a)$ 表示在状态s下，执行行动a，状态变成s'，并获得奖赏r的可能性。
$r(s,a)$ 在状态s下，执行行动a的期望奖赏。

$$r(s,a) \doteq \mathbb{E}[R_{t+1}|S_t=s,A_t=a] = \sum_{r\in\mathcal{R}} r \sum_{s'\in\mathcal{S}} p(s',r|s,a) \tag{2}$$

$r(s,a,s')$ 在状态s下，执行行动a，状态变成s'的期望奖赏。

$$r(s,a,s') \doteq \mathbb{E}[R_{t+1}|S_t=s,A_t=a,S_{t+1}=s'] = \frac{\sum_{r\in\mathcal{R}} rp(s',r|s,a)}{p(s'|s,a)} \tag{3}$$

π 策略$\pi

$$\pi = [\pi(s_1),\cdots,\pi(s_n)] \tag{4}$$

$\pi(s)$ 策略π ，在状态s下，选择的行动。
π_* 最优策略
$\pi(a|s)$ 随机策略在在状态s下，选择行动a的可能性。
$v_\pi(s)$ 策略π的状态价值方法。

$$v_\pi(s) \doteq \mathbb{E}[G_t|S_t=s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}|S_t=s\right] \tag{5}$$

$where$
π - polity
$\mathbb{E}_\pi[\cdot]$ - the expected value of a value follows policy π

$q_\pi(s,a)$ 策略π的行动价值方法。

$$q_\pi(s,a) \doteq \mathbb{E}[G_t|S_t=s,A_t=a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}|S_t=s,A_t=a\right] \tag{6}$$

$v_*(s)$ 最优状态价值方法。

$$v_*(s) \doteq \max_{\pi} v_\pi(s), \forall s \in \mathcal{S} \tag{7}$$

$q_*(s,a)$ 最优行动价值方法。

$$q_*(s,a) \doteq \max_{\pi} q_\pi(s,a),\ \forall s \in \mathcal{S}\ and\ a \in \mathcal{A}(s) \tag{8}$$
$$q_*(s,a) = \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a]$$

强化学习的术语

学习任务可分为两类：

- 情节性任务(episodic tasks)
  指（强化学习的问题）会在有限步骤下结束。比如：围棋。
- 连续性任务(continuing tasks)
  指（强化学习的问题）有无限步骤。比如：让一个立在指尖上的长棍不倒。（不知道这个例子好不好，我瞎编的。）

学习的方法：

- online-policy方法(online-policy methods)
  评估的策略和优化的策略是同一个。
- offline-policy方法(offline-policy methods)
  评估的策略和优化的策略不是同一个。意味着优化策略使用来自外部的模拟数据。

学习的算法：

- 预测算法(predication algorithms)
  计算每个状态的价值$v(s)$。然后预测(可以得到最大回报的)最优行动。
- 控制算法(predication algorithms)
  计算每个状态下每个行动的价值$q(s,a)$。

学习的算法：

- 列表方法(tabular methods)
  指使用表格存储每个状态（或者状态-行动）的价值。
- 近似方法(approximation methods)
  指使用一个函数来计算状态（或者状态-行动）的价值。
- 模型(model)
  环境的模型。可以模拟环境，模拟行动的结果。
  Dynamic Programming need a model。
- 基于模型的方法(model-base methods)
  通过模型来模拟。可以模拟行动，获得（状态或者行动）价值。

  > 注：这个模拟叫做模型模拟。

- 无模型的方法(model-free methods)
  使用试错法(trial-and-error)来获得（状态或者行动）价值。

  > 注：这个模拟叫做试错、试验、模拟等。
  > 无模型的方法，可以用于有模型的环境。

- 引导性(bootstrapping)
  （状态或者行动）价值是根据其它的（状态或者行动）价值计算得到的。

- 取样性(sampling)
  （状态或者行动）价值，或者部分值（比如：奖赏）是取样得到的。
  引导性和取样性并不是对立的。可以是取样的，并且是引导的。

强化学习算法的分类

强化学习的故事2：我们该用哪个方法？

> 如果有一个模型，可以获得价值函数$v(s)$或者$q(s,a)$的值 $\rightarrow$ 动态规划方法
> 如果可以模拟一个完整的情节 $\rightarrow$ 蒙特卡罗方法
> 如果需要在模拟一个情节中间就要学习策略 $\rightarrow$ 时序差分方法
> λ-return用来优化近似方法中的误差。
> 资格迹(Eligibility traces)用来优化近似方法中的，价值函数的微分。
> 预测方法是求状态价值方法$v(s)$或者$\hat{v}(s,\theta)$。
> 控制方法是求行动价值方法$q(s,a)$或者$\hat{(q)}(s,a,\theta)$。
> 策略梯度方法(Policy Gradient Methods)是求策略方法$\pi(a|s,\theta)$。

算法类别	需要模型	引导性	情节性任务	连续性任务
动态规划方法	Y	Y	-	-
蒙特卡罗方法	N	N	Y	N
时序差分方法	N	Y	Y	Y
策略梯度方法	N	Y	Y	Y

算法列表

在每个算法中，后面的算法会更好，或者更通用一些。

4 动态规划(Dynamic Programming)

动态规划是基于模型的方法。
注：一个常见的考虑是将每个action的reward设为-1，期望的结果V(S_T)为1。

- Iterative policy evaluation
  使用随机策略$\pi(a|s)$来迭代计算v(s)$
- Policy iteration (using iterative policy evaluation)
  通过使用迭代策略$\pi(s)$来优化了计算v(s)$部分。但是，还是使用了期望值。
- Value iteration
  优化了整个流程，直接用行动的最大回报作为$v(s)$的值。

5 蒙特卡罗方法(Monte Carlo Method)

- First-visit MC policy evaluation (returns $V \approx v$)
  在每个情节中，记录状态s第一个G。$v(s) = avg(G(s))$
- Monte Carlo ES (Exploring Starts)
  从一个特定起始点的蒙特卡罗方法。
  变成了计算$q(s,a)$。
- On-policy first-visit MC control (for ϵ-soft policies)
  在探索中使用了ϵ-soft策略。
- Incremental o♪ff-policy every-visit MC policy evaluation
  支持off-policy。
- Off-policy every-visit MC control (returns $\pi \approx \pi_*$)
  使用了贪婪策略来支持off-policy。

6 时序差分方法(Temporal-Difference Learning)

时序差分方法的思想是：

1. 在一个情节进行过程中学习。
   比如：计算到公司的时间问题。早上晚起了10分钟，可以认为会比以往晚到10分钟。而不用完成从家到公司整个过程。
2. 视为蒙特卡罗方法的通用化。蒙特卡罗方法是步数为完成情节的TD算法。

- Tabular TD(0) for estimating v_π
  计算$v(s)$的单步TD算法。
- Sarsa: An on-policy TD control algorithm
  计算$q(s,a)$的单步TD算法。
- Q-learning: An off♪-policy TD control algorithm
  是一个突破性算法。但是存在一个最大化偏差(Maximization Bias)问题。
- Double Q-learning
  解决了最大化偏差(Maximization Bias)问题。

7 多步时序差分方法

- n-step TD for estimating $V \approx v_\pi$
  计算$v(s)$的多步TD算法。
- n-step Sarsa for estimating $Q \approx q_* \triangleright$, or $Q \approx q_\pi$ for a given π
  计算$q(s,a)$的多步TD算法。
- O♪ff-policy n-step Sarsa for estimating $Q \approx q_* \triangleright$, or $Q \approx q_\pi$ for a given π
  考虑到重要样本，把ρ带入到Sarsa算法中，形成一个off-policy的方法。
  ρ - 重要样本比率(importance sampling ratio)

$$\rho \leftarrow \prod_{i=\tau+1}^{min(\tau+n-1,T-1)} \frac{\pi(A_t|S_t)}{\mu(A_t|S_t)} \qquad (\rho_{\tau+n}^{(\tau+1)}) \tag{9}$$

- n-step Tree Backup for estimating $Q \approx q_* \triangleright$, or $Q \approx q_\pi$ for a given π
  Tree Backup Algorithm的思想是每步都求行动价值的期望值。
  求行动价值的期望值意味着对所有可能的行动a都评估一次。

- Off♪-policy n-step $Q(¤\sigma)$ for estimating $Q \approx q_* \triangleright$, or $Q \approx q_\pi$ for a given π
  $Q(\sigma)$ 结合了Sarsa(importance sampling), Expected Sarsa, Tree Backup算法，并考虑了重要样本。
  当$\sigma = 1$时，使用了重要样本的Sarsa算法。
  当$\sigma = 0$时，使用了Tree Backup的行动期望值算法。

8 基于模型的算法

这里的思想是：通过体验来直接优化策略和优化模型（再优化策略）。

- Random-sample one-step tabular Q-planning
  通过从模型中获取奖赏值，计算$q(s,a)$。

- Tabular Dyna-Q
  如果$n = 0$，就是Q-learning算法。Dyna-Q的算法的优势在于性能上的提高。
  主要原因是通过建立模型，减少了执行行动的操作，模型学习到了$Model(S,A) \leftarrow R, S'$。

- Prioritized sweeping for a deterministic environment
  提供了一种性能的优化，只评估那些误差大于一定值θ的策略价值。

9 近似预测方法

预测方法就是求$v(s)$。

$$\hat{v}(s,\theta) \doteq \theta^T \phi(s), \quad \text{state value function} \tag{10}$$
$$where$$
$$\theta \text{ - value function's weight vector}$$

- Gradient Monte Carlo Algorithm for Approximating $\hat{v} \approx v_\pi$
  蒙特卡罗方法对应的近似预测方法。
- Semi-gradient TD(0) for estimating $\hat{v} \approx v_\pi$
  单步TD方法对应的近似预测方法。
  之所以叫半梯度递减的原因是TD(0)和n-steps TD计算价值的公式不是精确的（而蒙特卡罗方法是精确的）。
- n-step semi-gradient TD for estimating $\hat{v} \approx v_\pi$
  多步TD方法对应的近似预测方法。
- LSTD for estimating $\hat{v} \approx v_\pi$ (O(n2) version)

10 近似控制方法

控制方法就是求$q(s,a)$。

$$\hat{q}(s,a,\theta) \doteq \theta^T \phi(s,a), \quad \text{action value function} \tag{11}$$
$$where$$
$$\theta \text{ - value function's weight vector}$$

- Episodic Semi-gradient Sarsa for Control
  单步TD的近似控制方法。（情节性任务）
- Episodic semi-gradient n-step Sarsa for estimating $\hat{q} \approx q_* \triangleright$, or $\hat{q} \approx q_\pi$
  多步TD的近似控制方法。（情节性任务）

- Dif♪ferential Semi-gradient Sarsa for Control
  单步TD的近似控制方法。（连续性任务）
- Di♪fferential semi-gradient n-step Sarsa for estimating $\hat{q} \approx q_* \triangleright$, or $\hat{q} \approx q_\pi$
  多步TD的近似控制方法。（连续性任务）

12 λ⍉-return和资格迹(Eligibility traces)

求权重向量θ是通过梯度下降的方法。比如：

$$\begin{aligned} \delta_t &= G_t - \hat{v}(S_t, \theta_t) \\ \theta_{t+1} &= \theta_t + \alpha \delta_t \nabla \hat{v}(S_t, \theta_t) \end{aligned} \tag{12}$$

这里面，有三个元素：$\alpha, G_t, \nabla \hat{v}(S_t, \theta_t)$。每个都有自己的优化方法。

- α是学习步长
  要控制步长的大小。一般情况下步长是变化的。比如：如果误差δ_t变大了，步长要变小。
- G_t的计算
  可以通过本章的λ - return方法。
- $\nabla \hat{v}(S_t, \theta_t)$
  可以通过资格迹来优化。资格迹就是优化后的函数微分。
  为什么要优化，原因是在TD算法中$\hat{v}(S_t, \theta_t)$是不精确的。
  G_t也是不精确的。
  λ-return用来优化近似方法中的误差。
  资格迹(Eligibility traces)用来优化近似方法中的，价值函数的微分。
- Semi-gradient TD(λ⍉) for estimating $\hat{v} \approx v_\pi$
  使用了λ⍉-return和资格迹的TD算法。

- True Online TD(λ) for estimating $\theta^T\phi \approx v_\pi$
  Online TD(λ)算法

13 策略梯度方法

策略梯度方法就是求$\pi(a|s,\theta)$。

策略梯度方法的新思路(Policy Gradient Methods)

$$
\begin{aligned}
&\text{Reinforcement Learning} \doteq \pi_* \\
&\quad \updownarrow \\
&\pi_* \doteq \{\pi(s)\},\ s \in \mathcal{S} \\
&\quad \updownarrow \\
&\pi(s) = \underset{a}{argmax}\ \pi(a|s,\theta) \\
&where \\
&\pi(a|s,\theta) \in [0,1] \\
&s \in \mathcal{S},\ a \in \mathcal{A} \\
&\quad \updownarrow \\
&\pi(a|s,\theta) \doteq \frac{exp(h(s,a,\theta))}{\sum_b exp(h(s,b,\theta))} \\
&\quad \updownarrow \\
&exp(h(s,a,\theta)) \doteq \theta^T\phi(s,a) \\
&where \\
&\theta \text{ - policy weight vector}
\end{aligned}
\tag{13}
$$

- REINFORCE, A Monte-Carlo Policy-Gradient Method (episodic)
  基于蒙特卡罗方法的策略梯度算法。
- REINFORCE with Baseline (episodic)
  带基数的蒙特卡洛方法的策略梯度算法。
- One-step Actor-Critic (episodic)
  带基数的TD方法的策略梯度算法。
- Actor-Critic with Eligibility Traces (episodic)
  这个算法实际上是：

  1. 带基数的TD方法的策略梯度算法。
  2. 加上资格迹(eligibility traces)

- Actor-Critic with Eligibility Traces (continuing)
  基于TD方法的策略梯度算法。（连续性任务）

参照

- Reinforcement Learning: An Introduction, Richard S. Sutton and Andrew G. Barto c 2014, 2015, 2016
- 强化学习读书笔记 - 00 - 术语和数学符号
- 强化学习读书笔记 - 01 - 强化学习的问题
- 强化学习读书笔记 - 02 - 多臂老O虎O机问题
- 强化学习读书笔记 - 03 - 有限马尔科夫决策过程
- 强化学习读书笔记 - 04 - 动态规划
- 强化学习读书笔记 - 05 - 蒙特卡洛方法(Monte Carlo Methods)
- 强化学习读书笔记 - 06~07 - 时序差分学习(Temporal-Difference Learning)
- 强化学习读书笔记 - 08 - 规划式方法和学习式方法

- 强化学习读书笔记 - 09 - on-policy预测的近似方法
- 强化学习读书笔记 - 10 - on-policy控制的近似方法
- 强化学习读书笔记 - 11 - off-policy的近似方法
- 强化学习读书笔记 - 12 - 资格痕迹(Eligibility Traces)
- 强化学习读书笔记 - 13 - 策略梯度方法(Policy Gradient Methods)
- 强化学习读书笔记 - 14 - 心理学

评论列表

#1楼 2017-04-11 10:40 飞翔南

楼主写的非常好，我准备做RL的研究，最近也在看RL书籍，你这个博客算是有关RL最新的资料了，赞一个！！！

支持(0) 反对(0)

#2楼 2017-04-26 11:38 小茄子

楼主写的非常棒！我有一个小问题，关于连续性任务举得例子，控制倒立摆算么？

支持(0) 反对(0)

#3楼[楼主] 2017-05-08 17:18 SNYang

@ 小茄子

这个问题很有趣。连续性任务的一个特征是这个任务永远不会结束。

而“控制倒立摆”有可能结束，或者也会永远不结束。

从算法的角度来说：可以使用情节性任务的算法。

我查了一下：

[How to distinguish episodic task and continuous tasks?](https://stats.stackexchange.com/questions/271356/how-to-distinguish-episodic-task-and-continuous-tasks)

举了一个连续性学习例子：在internet上学习数学。

我可能也要修改一下这个博文。

支持(0) 反对(0)

#4楼 2017-06-26 10:56 liulizi

楼主这个系列讲的逻辑清晰，通俗易懂，感谢。

支持(0) 反对(0)