RL NOTE

1 Introduction

- 强化学习就是学习“做什么（即如何把当前的情景映射成动作）才能使得数值化的收益信号最大化。”

- 强化学习与监督学习、无监督学习的区别：

1. 监督学习是从外部监督者提供的带标注训练集中进行训练，而强化学习中没有监督者，自然也没有标注，只有（延时的）收益信号；
2. 无监督学习的目标是寻找未标注数据中隐含结构的过程，而强化学习的目标是最大化收益信号，尽管找到隐含结构是有益的，但并不能其最终目的。

- 强化学习要素

1. 策略：智能体在特定时间的行为方式，即可以说是环境状态到动作的映射。
2. 收益信号：定义了强化学习问题中的目标。
3. 价值函数：表示了从长远的角度看什么是好的。
4. 模型：对环境的反应模式的模拟/对外部环境的行为进行推断，从而用于规划（planning）

基于上述要素，会产生对于智能体的一些分类标准：基于策略、基于价值、Actor-Critic；有模型、无模型。

- Pairs of concepts

1. 学习 & 规划

学习：智能体与最初未知的环境实际交互并改善策略；

规划：智能体与已知的模型发生交互并改善策略；

2. 试探 & 开发

开发：指利用经验中可以获得最大收益的方法；

试探：指发现是否还有更好的利益获得方式；

3. 预测 & 控制

预测：给定一个策略，对未来进行评估；

控制：找到最优的策略，对未来进行优化；

2 马尔可夫决策过程

- MDP 框架

智能体 - 环境交互产生一个序列：$S_0, A_0, R_1, S_1, A_1, R_2, \ldots$，在此基础上定义如下函数：

$$p(s^{'}, r|s, a) = Pr(S_t = s, R_t = r|S_{t-1} = s, A_{t-1} = a)$$

$$p(s^{'}|s, a) = \mathbf{E}(R_t|S_{t-1} = s, A_{t-1} = a) = \sum_{r \in R} p(s^{'}, r|s, a)$$

$$r(s, a) = \mathbf{E}(R_t|S_{t-1} = s, A_{t-1} = a) = \sum_{r \in R} r \sum_{s^{'} \in S} p(s^{'}, r|s, a)$$

$$r(s, a, s^{'}) = \mathbf{E}(R_t|S_{t-1} = s, A_{t-1} = a, S_t = s^{'}) = \sum_{r \in R} \frac{p(s^{'}, r|s, a)}{p(s^{'}|s, a)}$$

回报：

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

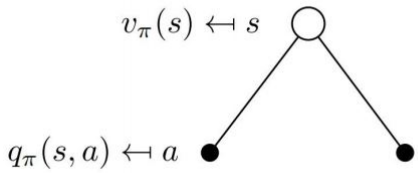
- 价值和策略函数

$$v_\pi(s) = \mathbf{E}_\pi(G_t|S_t = s), 对于所有 s \in S$$

$$q_\pi(s, a) = \mathbf{E}_\pi(G_t|S_t = s, A_t = a)$$

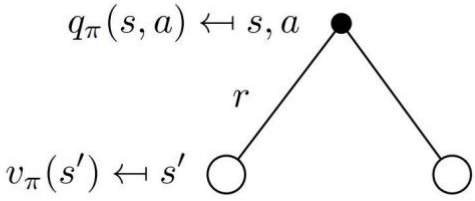
- 贝尔曼期望方程和回溯图

关于 v_π 的贝尔曼期望方程



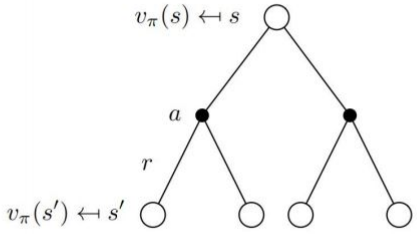
$$v_\pi(s) = \sum_{a \in A} \pi(a \mid s) q_\pi(s,a)$$

关于 q_π 的贝尔曼期望方程



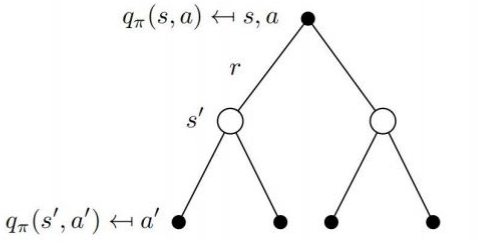
$$q_\pi(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s')$$

关于 v_π 的贝尔曼期望方程（2）



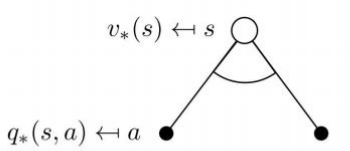
$$v_\pi(s) = \sum_{a \in A} \pi(a \mid s) \left(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s') \right)$$

关于 q_π 的贝尔曼期望方程（2）



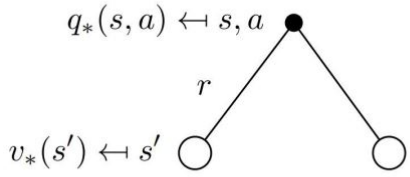
$$q_\pi(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \sum_{a' \in A} \pi(a' \mid s') q_\pi(s',a')$$

- 最优方程



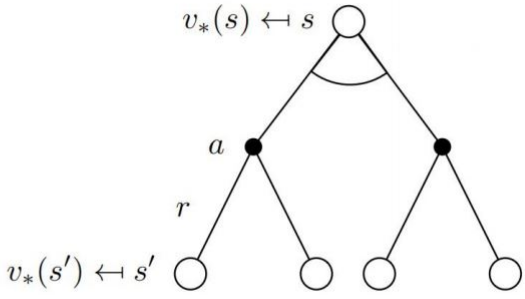
$$v_*(s) = \max_a q_*(s,a)$$

- q_* 的贝尔曼最优方程



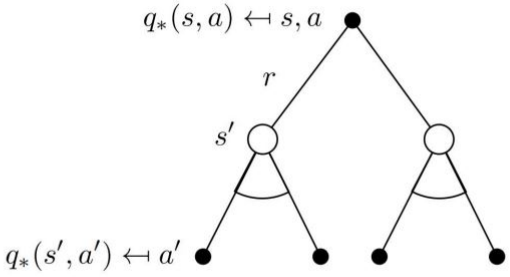
$$q_*(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_*(s')$$

- v_* 的贝尔曼最优方程（2）



$$v_*(s) = \max_a R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_*(s')$$

- q_* 的贝尔曼最优方程（2）



$$q_*(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \max_{a'} q_*(s',a')$$

3 动态规划

- 策略评估：针对任意一个策略 π，计算其状态价值函数 v_π

即用数值方法解：

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')]$$

用回溯图理解记忆。

- 策略改进

因为评价在状态 s 采用动作 a 的价值是：

$$q_\pi(s,a) = \sum_{s',r} p(s',r|,s,a)[r + \gamma v_\pi(s')]$$

根据一些证明，则可以将策略 π 改进为更好的策略 π'：

$$\pi' = argmax_a q_\pi(s,a)$$

- 价值迭代

即将上面两部融合在一起：

$$v_{k+1}(s) = max_a \sum_{s',r} p(s',r|s,r)[r + \gamma v_k(s')]$$

- 广义策略迭代

这个词代指策略评估和策略改进相互作用的一般思路，与这两个流程的粒度和其他细节无关。两者竞争与合作。

4 蒙特卡洛方法

4.1 预测：

- 首次访问和每次访问：比较简单，伪代码暂略；

- 该方法与 DP 方法的区别：通过回溯图可以说明，即只需要根据生成的每一幕采样算平均值即可，不需要下一状态 s' 的所有的情况和转移概率。

- 异策预测 + 增量实现

增量实现即一种通过记录状态出现次数快速计算平均数的方法。

异策预测：

对每一幕：

b = 任何能包括 π 的策略

根据 b 生成一幕序列：$S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$

$G = 0$

$W = 1$

对幕中每一步循环：$t = T-1, T-2, \ldots, 0$：

$G = \gamma G + R_{t+1}$

$C(S_t, A_t) += W$

$Q(S_t, A_t) += \frac{W}{C(S_t.A_t)}[G - Q(S_t, A_t)]$

$W \times = \frac{\pi(A_t|S_t)}{b(A_t|S_t)}$

若 $W = 0$，则退出循环

4.2 控制

- 策略评估采用动作价值函数而不是状态价值函数

因为在无模型的情况下，必须要显示地确定每个动作的价值。

- 同策首次访问 + $\epsilon - greedy$

同侧指用于生成采样数据序列的策略和用于实际决策的待评估和改进的策略是相同的。

伪代码:

对每一幕：

根据 π 生成一幕序列：$S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$

$G = 0$

对幕中每一步循环：$t = T-1, T-2, \ldots, 0$：

$G = \gamma G + R_{t+1}$

(S_t, A_t)在$S_0, A_0, S_1, A_1, \ldots, S_{t-1}, A_{t-1}$中没有出现过：

$Returns(S_t, A_t) += G$

$Q(S_t, A_t) = ave(Returns(S_t, A_t))$

$A^* = argmax_a Q(S_t, a)$

对所有 $a \in \mathbf{A}(S_t)$：

如果 $a = A^*$：

$\pi(a|S_t) = 1 - \epsilon + \frac{\epsilon}{|\mathbf{A}(S_t)|}$

否则：

$$\pi(a|S_t) = \frac{\epsilon}{|\mathbf{A}(S_t)|}$$

- 异策 + 重要性采样

为了解决的矛盾：希望学到的动作可以使随后智能体的行为是最优的，但是为了搜索所有的动作，他们需要采取非最优的行动。异策则是一个用来学习并成为最优策略（目标策略），另一个更有试探性，产生行动样本（行动策略）。

而重要性采样则是修正了由行动策略中得到回报期望的值：

$$\rho_{t:T-1} = \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}$$

$$v_\pi(s) = \mathbf{E}(\rho_{t:T-1} G_t | S_t = s)$$

异策控制：

对每一幕：

b = 任何软性策略（即在所有状态下选择所有动作的概率不为零）

根据 b 生成一幕序列：$S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$

$G = 0$

$W = 1$

对幕中每一步循环：$t = T-1, T-2, \ldots, 0$：

$G = \gamma G + R_{t+1}$

$C(S_t, A_t) += W$

$Q(S_t, A_t) += \frac{W}{C(S_t, A_t)}[G - Q(S_t, A_t)]$

$\pi(S_t) = argmax_a Q(S_t, a)$

如果 $A_t \neq \pi(S_t)$，则推出内层循环，处理下一幕

$$W\times=\frac{1}{b(A_t|S_t)}$$

5 时序差分方法

该方法结合了蒙特卡洛方法和动态规划方法的思想，（MC）TD 可以直接从与环境互动的经验中学习策略，不需要构建关于环境动态的模型；（DP）TD无须等待交互的最终结果，而是可以基于已得到的其他状态的估计值来更新当前状态的价值函数。

5.1 预测

- 计算价值

$$V(S_t)=V(S_t)+\alpha(R_{t+1}+\gamma V(S_{t+1})-V(S_t))$$

- n 步预测

$$G_t^{(n)}=R_{t+1}+\gamma R_{t+2}+\ldots+\gamma^{n-1}R_{t+n}+\gamma^n V(S_{t+n})$$

$$V(S_t)=V(S_t)+\alpha(G_t^{(n)}-V(S_t))$$

5.2 控制

- Sarsa 同策控制

预测同样使用动作价值函数：

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(terminal\text{-}state, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

n 步 Sarsa 就把上面 n 步预测的公式中的价值函数替换为动作价值函数即可。

- Q 学习 异策控制

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(terminal\text{-}state, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
    $S \leftarrow S'$;
  until S is terminal

5.3 与资格迹结合

资格迹是一种将 MC 方法和 TD 算法统一的思想之一。其优势在于只需要追踪一个迹向量，不需要存储最近的 n 个特征向量；第二点优势在于它不需要像 n 步收益思想一样依赖于未来的状态，而是采用后向视角的方式来更新价值函数。

- λ 回报（离线 λ 回报算法）

$$G_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G_t^{(n)}$$

$$V(S_t) = V(S_t) + \alpha(G_t^\lambda - V(S_t))$$

- TD(λ)

资格迹：

$$E_0(s) = 0$$

$$E_t(s) = \gamma\lambda E_{t-1}(s) + \mathbf{I}(S_t = s)$$

利用资格迹更新价值函数：

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$$

$$V(s) = V(s) + \alpha\delta_t E_t(s)$$

当 λ = 1，TD (1) 大致相当于每次访问 MC，如果只在每一幕结束时更新，则总更新和 MC 完全一样。

当 λ = 0，算法退化成简单时序差分方法。

值得注意的是以上两种算法在离线更新的情况下是等价的。

- 后续还有 “在线 λ 回报算法” 和 “真实在线 TD (λ)”

这两者在在想更新时是等价的，不过单纯的在线 TD (λ) 只在 λ = 0 时跟在线 λ 回报算法等价，其余情况不等价。

- Sarsa (λ)

类似的，先定义动作价值函数的 n 步回报 q_t^n，然后定义 λ 回报 q_t^λ。

前向视角就把价值函数均替换成动作价值函数即可，G_t^λ 换成 q_t^λ。

后向视角，资格迹表示变成：$E_t(s, a)$，然后更新时也将价值函数替换成动作价值函数即可。

6 价值函数近似

由于存储限制，可以从已知的策略 π 生成的经验来近似一个价值函数，而该函数不再生成一个表格，而是一个具有权值向量 $\mathbf{w} \in \mathbb{R}^d$ 的参数化函数，记作 $\hat{v}(\mathbf{w}, s) \approx v_\pi(s)$. $\mathbf{w}$ 是近似函数的参数，例如线性函数中的特征向量、神经网络中层之间的权重等。

- 随机梯度下降（SGD）

目标是最小化近似函数和实际函数的均方差，采用思想是对近似函数的梯度采样并更新$\mathbf{w}$：

$$\mathbf{w} = \mathbf{w} + \alpha[v_\pi(S) - \hat{v}(\mathbf{w}, s)]\nabla_{\mathbf{w}}\hat{v}(S, \mathbf{w})$$

状态价值函数：

MC 方法：将价值函数替换为 G_t

TD (0) 方法：将价值函数替换为 $R_{t+1} + \gamma\hat{v}(S_{t+1}, \mathbf{w})$

TD (λ) 方法：将价值函数替换为 G_t^λ，若是后向视角则需要将指示函数替换做近似函数的梯度

动作价值函数：

将近似函数替换为$\hat{q}(S_t, A_t, \mathbf{w})$即可，其余类似

- 线性函数近似

由于线性函数可以表示为：

$$\hat{v}(S, \mathbf{w}) = \mathbf{w}^T\mathbf{x}(S)$$

再利用 SGD 方法可以得到近似函数的梯度为：$\mathbf{x}(S)$ 这个比较简单的形式。

- DQN

算法 14.5: 带经验回放的深度 Q 网络

输入: 状态空间 $\mathcal{S}$，动作空间 $\mathcal{A}$，折扣率 γ，学习率 α

1 初始化经验池 $\mathcal{D}$，容量为 N;
2 随机初始化 Q 网络的参数 ϕ;
3 随机初始化目标 Q 网络的参数 $\hat{\phi} = \phi$;
4 **repeat**
5 初始化起始状态 s;
6 **repeat**
7 在状态 s，选择动作 $a = \pi^{\epsilon}$;
8 执行动作 a，观测环境，得到即时奖励 r 和新的状态 s';
9 将 s, a, r, s' 放入 $\mathcal{D}$ 中;
10 从 $\mathcal{D}$ 中采样 ss, aa, rr, ss';
11 $y = \begin{cases} rr, & ss' \text{为终止状态}, \\ rr + \gamma \max_{a'} Q_{\hat{\phi}}(\mathrm{ss}', \mathrm{a}'), & \text{否则} \end{cases}$;
12 以 $\left(y - Q_{\phi}(\mathrm{ss}, \mathrm{aa})\right)^2$ 为损失函数来训练 Q 网络;
13 $s \leftarrow s'$;
14 每隔 C 步，$\hat{\phi} \leftarrow \phi$;
15 **until** s 为终止状态;
16 **until** $\forall s, a$，$Q_{\phi}(\mathrm{s}, \mathrm{a})$ 收敛;

输出: Q 网络 $Q_{\phi}(\mathrm{s}, \mathrm{a})$

注意到 DQN 的两个特点：

1. 经验回放

把一段时期内的经验重新过一遍，更新参数，体现在9、10行中。

2. 目标网络（$\hat{\phi}$）和行为网络（ϕ）

奖励由目标网络获得，然后与行为网络的估计值比较得到目标值用于更新行为网络，行为网络的值固定一段时间传递给目标网络。即动作是由行为网络得到的，但是价值是由目标网络得到的。

- *最小二乘方法

预测：

线性最小二乘法预测(2)

求解思路是逆向思维，假设已经找到这个参数，那么他应该满足最小LS(w)，通过把LS展开，可以直接得到w：

$$\mathbb{E}_{\mathcal{D}}[\Delta \mathbf{w}] = 0$$

$$\alpha \sum_{t=1}^{T} \mathbf{x}(s_t)(v_t^{\pi} - \mathbf{x}(s_t)^{\top}\mathbf{w}) = 0$$

$$\sum_{t=1}^{T} \mathbf{x}(s_t) v_t^{\pi} = \sum_{t=1}^{T} \mathbf{x}(s_t)\mathbf{x}(s_t)^{\top}\mathbf{w}$$

$$\mathbf{w} = \left(\sum_{t=1}^{T} \mathbf{x}(s_t)\mathbf{x}(s_t)^{\top}\right)^{-1} \sum_{t=1}^{T} \mathbf{x}(s_t) v_t^{\pi}$$

LSMC

$$0 = \sum_{t=1}^{T} \alpha(G_t - \hat{v}(S_t, \mathbf{w}))\mathbf{x}(S_t)$$

$$\mathbf{w} = \left(\sum_{t=1}^{T} \mathbf{x}(S_t)\mathbf{x}(S_t)^{\top}\right)^{-1} \sum_{t=1}^{T} \mathbf{x}(S_t) G_t$$

LSTD

$$0 = \sum_{t=1}^{T} \alpha(R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w}) - \hat{v}(S_t, \mathbf{w}))\mathbf{x}(S_t)$$

$$\mathbf{w} = \left(\sum_{t=1}^{T} \mathbf{x}(S_t)(\mathbf{x}(S_t) - \gamma \mathbf{x}(S_{t+1}))^{\top}\right)^{-1} \sum_{t=1}^{T} \mathbf{x}(S_t) R_{t+1}$$

LSTD(λ)

$$0 = \sum_{t=1}^{T} \alpha \delta_t E_t$$

$$\mathbf{w} = \left(\sum_{t=1}^{T} E_t(\mathbf{x}(S_t) - \gamma \mathbf{x}(S_{t+1}))^{\top}\right)^{-1} \sum_{t=1}^{T} E_t R_{t+1}$$

线性最小二乘法预测算法的收敛性

On/Off-Policy	Algorithm	Table Lookup	Linear	Non-Linear
On-Policy	MC	✓	✓	✓
	LSMC	✓	✓	-
	TD	✓	✓	✗
	LSTD	✓	✓	-
Off-Policy	MC	✓	✓	✓
	LSMC	✓	✓	-
	TD	✓	✗	✗
	LSTD	✓	✓	-

控制：

下面是线性Q学习的更新式

$$\delta = R_{t+1} + \gamma \hat{q}(S_{t+1}, \pi(S_{t+1}), \mathbf{w}) - \hat{q}(S_t, A_t, \mathbf{w})$$
$$\Delta \mathbf{w} = \alpha \delta \mathbf{x}(S_t, A_t)$$

LSTDQ算法：求解 总更新=零

$$0 = \sum_{t=1}^{T} \alpha(R_{t+1} + \gamma \hat{q}(S_{t+1}, \pi(S_{t+1}), \mathbf{w}) - \hat{q}(S_t, A_t, \mathbf{w}))\mathbf{x}(S_t, A_t)$$
$$\mathbf{w} = \left(\sum_{t=1}^{T} \mathbf{x}(S_t, A_t)(\mathbf{x}(S_t, A_t) - \gamma \mathbf{x}(S_{t+1}, \pi(S_{t+1})))^{\top}\right)^{-1} \sum_{t=1}^{T} \mathbf{x}(S_t, A_t) R_{t+1}$$

7 策略梯度

直接学习参数化策略，动作的选择不再直接依赖于价值函数，价值函数可以用于学习策略的参数。学习方法都基于某种性能度量 $J(\theta)$ 基于该策略参数的梯度，目标是最大化性能指标，更新近似于 J 的梯度上升：

$$\theta_{t+1} = \theta_t + \alpha \widehat{\nabla J(\theta)}$$

其中 $\widehat{\nabla J(\theta)}$ 是一个随机估计，其期望是性能指标对它的参数的梯度的近似。

- 优势与劣势

优势：具有更好的收敛性、对于高维度或者连续状态空间来说更有效、能够学习到一些随机策略；

劣势：收敛到局部最优而不是全局最优、效率低而且方差大；

- 策略梯度定理

- 蒙特卡洛策略梯度（REINFORCE）

输入：一个可到的参数化策略$\pi(a|s,\theta)$

算法参数：步长 $\alpha > 0$

初始化参数策略 $\boldsymbol{\theta} \in \mathbb{R}^d$

循环（对于每一幕）：

根据 $\pi(\cdot|\cdot, \boldsymbol{\theta})$，生成一幕序列

对于每一步循环，$t = 0, 1, \ldots, T-1$:

$G = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$

$\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha \gamma^t G \nabla ln \pi(A_t|S_t, \boldsymbol{\theta})$

暂时忽略 γ 来看，每一个增量的更新都正比 G_t 和一个向量的乘积，该向量是选取该动作的概率的梯度除以这个概率本身。该向量是参数空间中使得将来在状态 S_t 下重复选择动作 A_t 的概率增加最大的方向。正比于回报：促进更新；反比概率：避免频繁选取的动作影响结果的最优性。

- 带有基线的 REINFORCE

采用这个方法可以减小方差

输入：一个可微的参数化策略$\pi(a|s, \theta)$

输入：一个可微的参数化状态价值函数 $\hat{v}(s, \mathbf{w})$

算法参数：步长 $\alpha^{\boldsymbol{\theta}} > 0, \alpha^{\mathbf{w}} > 0$

初始化参数 $\boldsymbol{\theta} \in \mathbb{R}^d, \mathbf{w} \in \mathbb{R}^{d'}$

循环（对于每一幕）：

根据 $\pi(\cdot|\cdot, \boldsymbol{\theta})$，生成一幕序列

对于每一步循环，$t = 0, 1, \ldots, T-1$:

$G = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$

$$\delta = G - \hat{v}(S_t, \mathbf{w})$$

$$\mathbf{w} = \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S_t, \mathbf{w})$$

$$\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \gamma^t \delta G \nabla ln \pi(A_t | S_t, \boldsymbol{\theta})$$

- Actor-Critic 方法

该方法的提出是为了利用时序差分，来消除蒙特卡洛学习缓慢和不利于在线实现和连续性模型的问题。因为自举操作（用后继各个状态的价值估计来更新当前某个状态的价值估计值）带来的偏差以及状态表示上的依赖降低了方差并加快了学习。同时，也可以灵活的选择自举操作的程度来优化结果和过程。

单步 Actor-Critic 方法：

实际上就是把回报 δ 的计算方法替换为 TD 的计算方法即可（并且不需要生成所有序幕，只要根据策略采取某个动作并观察到下一个状态和奖励即可）：

$$\delta = R + \gamma \hat{v}(S_t, \mathbf{w}) - \hat{v}(S_t, \mathbf{w})$$

对之后的 γ^t 做微调即可。

如果要推广到前向视角的 n 步方法和 λ 回报算法，只需要替换响应的回报形式$G_t^{(n)}$. G_t^{λ}即可。

带有资格迹的 Actor-Critic 方法：

引入两个资格迹，并分别按照 TD(λ) 后向视角更新即可。

- 自然梯度策略

这一点是为了防止固定步长在造成的问题，有时变化太小，有时则变化太大，设置小了无法收敛，设置大了模型崩溃。

自然策略梯度是独立于重参数化的，当针对策略进行一个小的、固定量的改变时，它会找到最接近vanilla梯度的上升方向

$$\nabla_\theta^{nat}\pi_\theta(s,a) = G_\theta^{-1}\nabla_\theta\pi_\theta(s,a)$$

其中 G_θ 是**Fisher信息矩阵**

$$G_\theta = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\nabla_\theta \log \pi_\theta(s,a)^T\right]$$

策略梯度算法的总结

- **策略梯度有许多等价形式**

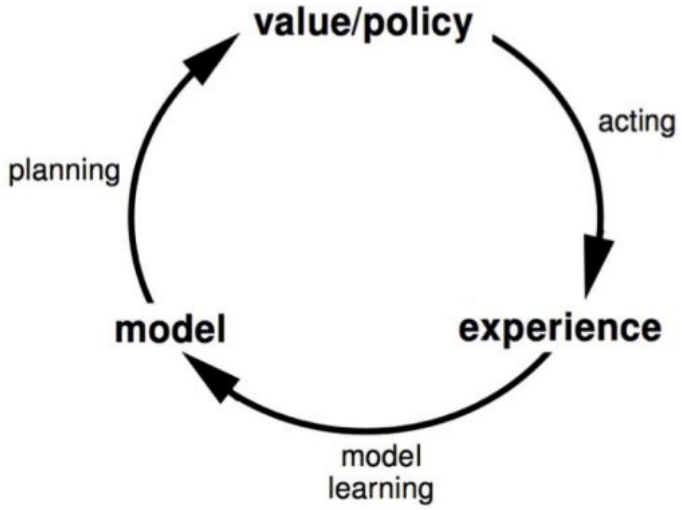
$$\begin{aligned}
\nabla_\theta J(\theta) &= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ v_t\right] && \text{REINFORCE}\\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ Q^w(s,a)\right] && \text{Q Actor-Critic}\\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ A^w(s,a)\right] && \text{Advantage Actor-Critic}\\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ \delta\right] && \text{TD Actor-Critic}\\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ \delta e\right] && \text{TD}(\lambda)\text{ Actor-Critic}\\
G_\theta^{-1}\nabla_\theta J(\theta) &= w && \text{Natural Actor-Critic}
\end{aligned}$$

- **每种形式都对应一个随机梯度上升算法**
- Critic**使用策略评估（例如MC或TD学习）来估计** $Q^\pi(s,a)$, $A^\pi(s,a)$ **或** $V^\pi(s)$

8 整合学习与规划

将基于模型和无模型的强化学习方法结合起来，虽然这两种方法有较大差异，但是也有相似点：核心都是价值函数计算、并且都是基于对未来事件的展望计算一个回溯价值，然后用该价值更新近似价值函数。

- 基于模型



优势：可以利用监督学习、当学习价值或策略困难时、能对智能体理解环境提供帮助；

劣势：是对运行环境的近似，引起双重误差。

模型是一个 MDP<S,A,P,R> 的参数化表现形式：M<P′,R′>。

查表模型：把通过经验得到各个状态行为转移概率和奖励存入表中即可：

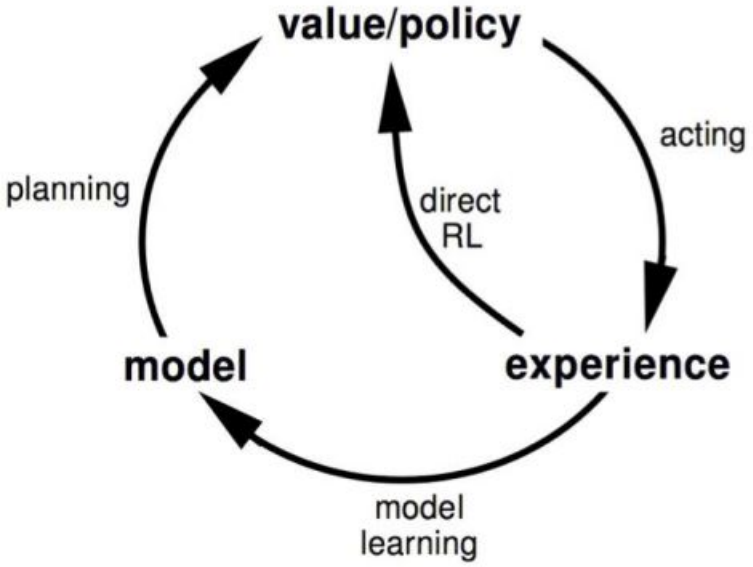
$$\hat{\mathcal{P}}^a_{s,s'} = \frac{1}{N(s,a)} \sum_{t=1}^{T} \mathbf{1}(S_t, A_t, S_{t+1} = s, a, s')$$

$$\hat{\mathcal{R}}^a_s = \frac{1}{N(s,a)} \sum_{t=1}^{T} \mathbf{1}(S_t, A_t = s, a) R_t$$

规划：给定一个M，求解MDP，即找到基于该模型的最优价值函数或者策略（给定一个状态s，找到最优的动作a）。

基于采样的规划：利用模型产生一个时间步长的虚拟经验，之后用无模型的强化学习方法来学习得到价值或策略函数。

- Dyna



dyna-Q 算法：

Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$
Do forever:
(a) $S \leftarrow$ current (nonterminal) state
(b) $A \leftarrow \varepsilon$-greedy(S,Q)
(c) Execute action A; observe resultant reward, R, and state, S'
(d) $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
(e) $Model(S,A) \leftarrow R, S'$ (assuming deterministic environment)
(f) Repeat n times:
$S \leftarrow$ random previously observed state
$A \leftarrow$ random action previously taken in S
$R, S' \leftarrow Model(S,A)$
$Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$

- 基于模拟的搜索

从当前时刻开始，使用基于模拟采样的规划，构建一个关注于短期未来的前向搜索树，做为无模型强化学习方法的资源来寻找最优策略。

蒙特卡洛树搜索

设计思路：

> 根据「棋感」在脑海里大致筛选出了几种「最可能」的走法，然后再想走了这几种走法之后对手「最可能」的走法，然后再想自己接下来「最可能」的走法。

4个概念（阶段）：

1. 模拟

选定一个结点，根据预演策略选择动作进行整幕的模拟，首先由树内确定性策略决定（选择动作最大限度提高Q(s,a)），树外则由预演策略选取（随机）。简单的说就是用随机快速走子的方法下完一盘棋。

2. 选择

选择一个“最有可能会走到的”一个未被评估的局面。评价公式略。

3. 扩展

将刚刚选择的结点加上一个 “0/0” 的结点，然后进入下一步模拟。

4. 回溯

即从子结点沿更新路径回到父节点并更新统计信息。

优点：越好的点访问次数越多，反之亦然、可以动态评估各状态价值但又避免了动态规划产生的维度灾难。

9 探索和利用

- 几个基本探索方法

朴素探索：即ε-greedy，以 1-ε 的概率利用，以 ε 的概率探索。

乐观初始估计：优先选择当前被认为是最高价值的行为。

不确定优先：优先尝试不确定价值的行为，可以利用置信区间上界（UCB）来指导选择动作。

概率匹配：先估计每一个行为可能是最佳行为的概率，然后依据这个概率来选择后续行为。

信息状态搜索：将已经探索的状态作为状态的一部分联合智能体的状态组成新的状态，并以此为基础探索。

依据状态-行为空间的探索：针对每一个当前的状态，以一定的方法尝试之前该状态下没有尝试过的行为。

参数化探索：针对策略的函数近似，尝试不同的参数设置。