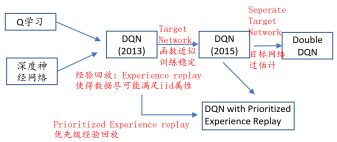
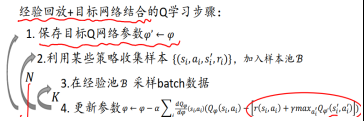
深度强化学习：使用深度神经网络作为强化学习的函数近似器。

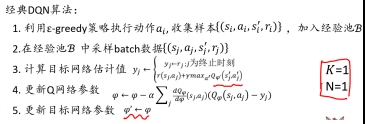


传统在线 Q 学习问题：1，样本不满足 iid；2，无法证明收敛，更新并非梯度下降。



4. 更新参数 $\varphi \leftarrow \varphi - \alpha \sum_i \frac{d Q_\varphi(s_i,a_i)}{d\varphi}(Q_\varphi(s_i,a_i) - [r(s_i,a_i) + \gamma \max_{a_i'} Q_{\varphi'}(s_i', a_i')])$

Batch 训练降低方差



多步回报得更低的方差。Q 值估计不准，TD 目标值中的 max 引入正向偏差，导致下一时刻的目标值过估计，解:**DDQN** 使用不同的网络来分别计算目标 Q 网络值和选择动作；当前 Q 选动作；目标 Q 值估计目标值。

标准DQN $y_j \leftarrow r(s_j,a_j) + \gamma Q_{\varphi'}(s_j', \text{argmax} Q_{\varphi'}(s_i', a_i'))$

Double DQN $y_j \leftarrow r(s_j,a_j) + \gamma Q_{\varphi'}(s_j', \text{argmax} Q_{\varphi}(s_i', a_i'))$

DQN 中经验池均匀采样的效率不高，不能反映样本的价值:带优先级的经验回放，利用 TD 误差去衡量优先级，大优先级高。**优先级经验回放带来的问题**：TD 误差对噪声敏感；TD 误差小的得不到更新；过分关注 TD 误差大的样本，丧失样本多样性。引入重要性采样权重来平衡“有偏”问题。前期注重优先级高后期注重无偏性。

Dueling-DQN：将 Q 函数分解成不依赖动作的值函数$V(s)$依赖动作的优势函数$A(s,a)$。

分解的原因：对于很多状态并不需要估计每个动作的值，增加了 V 函数的学习机会；V 函数的泛化性能好，当有新动作加入时，并不需要重新学习；减少了 Q 函数由于状态和动作维度差导致的噪声和突变**模仿学习**

适用情况 1.提供多演示轨迹：状态-动作序列 2.很容易演示，轨迹容易收集。**目标**：智能体需要找到期望策略，使得该策略下状态-动作轨迹分布尽可能匹配专家演示样本。**输入**：S,A, $P(s'|s,a)$；无R：专家演示样本集 $s0, a0, s1, a1, \ldots$。**利用监督学习**获得期望策略(状态到专家动作的映射)输入为演示样本集中的状态，标签为演示样本集中的动作。**行为克隆：**专家演示样本集$\{St, at|t \in T\}$,在专家演示样本集训练得到期望策略$\hat{\pi}$最小化误差$\sum_{t \in T}(\hat{\pi}(St) - at)$。**监督学习成功的基本假设：**训练集和测试集的分布一致，而行为克隆存在的问题：复合误差，最终导致分布漂移**样本增广：**累计误差的解决办法。**数据聚合：**

1. 训练$\pi_\theta(a_t|s_t)$基于专家演示数据集 $\{(s_t, a_t|t \in \mathcal{T}\}$
2. 执行$\pi_\theta(a_t|s_t)$得到新轨迹$\mathcal{T}_\pi = \{s_1, s_2, \ldots s_M\}$
3. 专家再次标记新轨迹$\mathcal{T}_\pi$的标签 $\{a_1, a_2, \ldots a_M\}$
4. 数据聚合: $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{T}_\pi$
5. 执行步骤1，循环

数据聚合的问题: 1.不安全/部分训练的策略，2. 需要大量专家标记,与任务相关。**模仿学习经常面临**：数据分配不匹配；专家非马尔可夫行为。**专家多模态行为一些提升的办法：**1.增广 2. 从一个稳定的轨迹分布中采样 3.聚合,4 改进模型提升精度 5,多的数据。**逆强化学习**假设专家策略π_E是最优的，从专家演示样本中恢复出奖赏函数$R*$

价值函数逼近

好处:更少的参数;泛化能力：更少样本学精确价值函数;多样性:多种特征表示和逼近器结构。**价值函数逼近的类型：**1. V(s);2. Q(s,a)3.Q(s)输出所有的 Q(s, a), ∀a

线性函数逼近:

用特征向量 (Feature Vector)代表某一状态:

$$\mathbf{x}(s)=[x_1(s), x_2(s), x_3(s) \ldots x_n(s)]^T$$

$$\hat{V}(s, \mathbf{w}) = \mathbf{x}(s)^T \mathbf{w} = \sum_{j=1}^{n} \mathbf{x}_j(s) \mathbf{w}_j$$

均方误差(MSE):逼近函数和真实价值函数之间的误差(期望损失) d 智体在 π 的状态分布

$$J(w) = \int s\ d(s)(V\pi(s) - V^(s,w))^2$$

对线性逼近器，用样本近似期望损失:

$$\frac{1}{M} \sum_{m=1}^{M} (V_\pi(s_m) - \mathbf{x}(s_m)^T \mathbf{w})^2$$

Π 代表从 f(x)到 g(x|w)最佳匹配。满足:

$$\hat{V}(\mathbf{w}^*) = \Pi(V_\pi)$$
$$s.t.\ \ \mathbf{w}^* = \arg\min_{\mathbf{w}} \mathbb{E}_\pi \left[(V_\pi(s) - \mathbf{x}(s)^T \mathbf{w})^2 \right]$$

$$\mathbf{w}^* = \left(\sum_{m=1}^{T} \mathbf{x}(s_m) \mathbf{x}(s_m)^T \right)^{-1} \sum_{m=1}^{T} \mathbf{x}(s_m) V_\pi(s_m)$$

最小二乘法优点：计算过程简单，一次性求解最佳权重.求逆,O(n3);当特征空间比较大时，计算量大，矩阵可能不满秩，易有误差

常见的特征表示方法

查表法;离散化法;粗糙编码:圆形区域特征可以重叠.**径向基函数**泛化特征值可以是 [0,1]，代表属于该特征的程度，

价值迭代 + 离散化方法

1: 将状态空间离散化成状态子集 $\{\mathcal{S}_k\}, k = 1, \ldots, K$
2: 定义有限动作集 $\{\mathcal{A}_l\}, l = 1, \ldots, L$
3: 初始化 $Q_{k,l}^{(0)}$ (e.g. $Q_{k,l}^{(0)} = 0$), $i = 0$
4: **repeat** {在第 i 次迭代}
5: 更新 Q 值

$$Q_{k,l}^{(i+1)} = \mathcal{R}_{k,l} + \gamma \max_{l'} Q_{k',l'}^{(i)}, \quad \forall k, l$$

P, R为确定型

6: $i \leftarrow i + 1$
7: **until** $\|Q^{(i)} - Q^{(i-1)}\| \leq \varepsilon$

收敛性;离散化程度越高，结果越精确,但是存储空间和计算量增加；维数灾(指数增)

Fitted Q Iteration（Q_learning）

Least-Squared FQI

1: 定义特征向量 $\mathbf{x}$, 和线性 Q 逼近器 $Q(s,a,\mathbf{w}_0) = \mathbf{x}^T \mathbf{w}_0$, 初始化重 $\mathbf{w}_0$, $i = 0$
2: 采样一组数据 $\{(s_k, a_k, r_{k+1}, s_{k+1})\}, k = 1, \ldots, K$
3: **repeat**
4: **for all** $k = 1, \ldots, K$ **do**
5: 计算 $q_k = r_{k+1} + \gamma \max_{a'} \hat{Q}(s_{k+1}, a', \mathbf{w}_i)$
6: **end for**
7: 更新 $\mathbf{w}_{i+1} = \arg\min_{\mathbf{w}} \frac{1}{K} \sum_{k=1}^{K} (q_k - \mathbf{x}(s_k, a_k)^T \mathbf{w})^2$
8: $i \leftarrow i + 1$
9: **until** $\mathbf{w}_i$ 收敛或迭代一定次数

策略迭代 + 最小二乘（闭解）

根据 pi 采样?如果 π 是确定策略，需要探索的动作,提泛化能力，如果 π 是随机策略,并且 π(a|s) > 0，可以 at ~ π(st)

最小二乘策略迭代 LSPI：

1: 给定策略 π_0, 定义特征向量 $\mathbf{x}(s,a)$, 初始化 $i = 0$
2: **repeat**
3: 采样一组数据集 $\{(s_t, a_t, r_{t+1}, s_{t+1})\}$
4: (策略评估:) 针对策略 π_i 使用 LSTD-Q 计算权重 $\mathbf{w}_i$, 得到策略的线性 Q 函数逼近器 $\hat{Q}_i(s,a,\mathbf{w}_i) = \mathbf{x}^T(s,a)\mathbf{w}_i$
5: (策略提升:) 提取贪心策略 $\pi_{i+1}(s) = \arg\max_a \hat{Q}_i(s,a,\mathbf{w}_i)$
6: $i \leftarrow i + 1$
7: **until** π_i 收敛或迭代一定次数

approximate VI/PI 优缺点

优点:收敛;缺点:每次迭代的计算量大,线性最小二乘法求样本特征矩阵的逆;通常 PI 的迭代次数要比 VI 迭代次数少

预测学习 + 随机梯度下降法

梯度下降 MC 预测算法

1: 给定策略 π, 定义价值逼近器 $\hat{V}(s,\mathbf{w})$, 初始化 $\mathbf{w}$
2: **repeat** {对每个 episode}
3: 根据 π 采样轨迹 $\{s_0, a_0, r_1, s_1, \ldots, s_T\}$
4: **repeat** {对 episode 中每个首次访问的 s_t}
5: 计算回报 $G_t = r_{t+1} + \gamma r_{t+2} + \ldots$
6: 计算更新量 $\Delta\mathbf{w} = \alpha(G_t - \hat{V}(s_t,\mathbf{w}))\nabla_{\mathbf{w}} \hat{V}(s_t,\mathbf{w})$
7: 更新权重 $\mathbf{w} \leftarrow \mathbf{w} + \Delta\mathbf{w}$
8: **until**
9: **until**

梯度下降 TD 预测学习，

1: 给定策略 π, 定义价值逼近器 $\hat{V}(s,\mathbf{w})$, 初始化 $\mathbf{w}$, $s_t = s_0$, $t = 0$
2: **loop**
3: 选择执行动作 $a_t \sim \pi(s_t)$, 观测 r_{t+1}, s_{t+1}
4: 计算更新量 $\Delta\mathbf{w} = \alpha(r_{t+1} + \gamma \hat{V}(s_{t+1},\mathbf{w}) - \hat{V}(s_t,\mathbf{w}))\nabla_{\mathbf{w}} \hat{V}(s_t,\mathbf{w})$
5: 更新权重 $\mathbf{w} \leftarrow \mathbf{w} + \Delta\mathbf{w}$
6: $t \leftarrow t + 1$
7: **end loop**

对于前向 TD(λ)：上式为

$$\Delta\mathbf{w} = \alpha(G_t^\lambda - \hat{V}(s_t,\mathbf{w}))\nabla_{\mathbf{w}} \hat{V}(s_t,\mathbf{w})$$

梯度下降 TD(λ)（后向）算法：

4: 计算 TD 误差 $\delta_t = r_{t+1} + \gamma \hat{V}(s_{t+1},\mathbf{w}) - \hat{V}(s_t,\mathbf{w})$
5: 更新资格迹 $e_t = \gamma\lambda e_{t-1} + \nabla_{\mathbf{w}} \hat{V}(s_t,\mathbf{w})$
6: 更新权重 $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta_t e_t$

梯度下降 MC 控制算法

1: 定义 Q 函数逼近器 $\hat{Q}(s,a,\mathbf{w})$, 初始化 $\mathbf{w}$
2: 提取出 ϵ 贪心策略 $\pi = \epsilon\text{-greedy}(\hat{Q}(\mathbf{w}))$
3: **repeat** {对每个 episode}
4: 根据 π 采样轨迹 $\{s_0, a_0, r_1, s_1, \ldots, s_T\}$
5: **repeat** {对 episode 中每个首次访问的 (s_t, a_t)}
6: 计算回报 $G_t = r_{t+1} + \gamma r_{t+2} + \ldots$
7: 计算更新量 $\Delta\mathbf{w} = \alpha(G_t - \hat{Q}(s_t,a_t,\mathbf{w}))\nabla_{\mathbf{w}} \hat{Q}(s_t,a_t,\mathbf{w})$
8: 更新权重 $\mathbf{w} \leftarrow \mathbf{w} + \Delta\mathbf{w}$
9: **until**
10: 更新策略 $\pi = \epsilon\text{-greedy}(\hat{Q}(\mathbf{w}))$
11: **until**

梯度下降 Sarsa 算法

1: 定义 Q 函数逼近器 $Q(s,a,\mathbf{w})$, 初始化 $\mathbf{w}$, $\pi = \epsilon\text{-greedy}(\hat{Q}(\mathbf{w}))$, $s_t = s_0$, $t = 0$
2: 采样动作 $a_t \sim \pi(s_t)$ 并执行, 观测 (r_{t+1}, s_{t+1})
3: **loop**
4: 采样动作 $a_{t+1} \sim \pi(s_{t+1})$ 并执行, 观测 r_{t+2}, s_{t+2}
5: 根据 $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$ 计算 TD 误差

$$\delta_t = r_{t+1} + \gamma \hat{Q}(s_{t+1}, a_{t+1}, \mathbf{w}) - \hat{Q}(s_t, a_t, \mathbf{w})$$

6: 更新权重 $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta_t \nabla_{\mathbf{w}} \hat{Q}(s_t, a_t, \mathbf{w})$
7: 更新策略 $\pi = \epsilon\text{-greedy}(\hat{Q}(\mathbf{w}))$
8: $t \leftarrow t + 1$
9: **end loop**

梯度下降 Sarsa(λ) 算法

6: 更新资格迹 $e_t = \gamma\lambda e_{t-1} + \nabla_{\mathbf{w}} \hat{Q}(s_t, a_t, \mathbf{w})$
7: 更新权重 $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta_t e_t$

梯度下降的 Q 学习算法

$$\delta_t = r_{t+1} + \gamma \max_{a'} \hat{Q}(s_{t+1}, a', \mathbf{w}) - \hat{Q}(s_t, a_t, \mathbf{w})$$

5: 更新权重 $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta_t \nabla_{\mathbf{w}} \hat{Q}(s_t, a_t, \mathbf{w})$

策略梯度

策略 RL 好处:更好的收敛性;有效解决大规模动作集或连续动作空间问题;能够学习随机策略。缺点:收敛到局部最优;策略评估费力且方差大.**策略逼近器不同形式:** p(a|s)=pi(a|s,θ), a=pi(s, θ), $a \sim \mathcal{N}(\mu(s,\theta), \sum(s,\theta))$.**不同优化目标。优化问题**:无梯度优化 (爬山法);梯度优化.**有限差分法(FD):**对策略参数θ每个维度增加微小扰动计算.

解析法求策略梯度:采样 N 条轨迹,用样本近似策略梯度: $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) \right) \left(\sum_{t=0}^{T-1} r_{t+1} \right)$

REINFORCE 算法（蒙特卡洛策略梯度算法）: 利用时间连贯性改变解析法的$\nabla J(\theta)$:t 时刻之前的奖励与 t 时刻的策略无关，所以不影响 t 时刻的梯度更新;同时将上述扩展到连续运行场景。实际算法不可能产生无穷长的轨迹,截断计算。

1: 定义策略逼近器 π_θ, 初始化参数 θ, 初始状态分布 d
2: **repeat**
3: 根据 d 和 π_θ 采样 N 条轨迹 $\tau_i = \{s_0^i, a_0^i, r_1^i, s_1^i, \ldots\}$
4: **for all** $i = 1, \ldots, N$ **do**
5: **for all** $t = 0, 1, \ldots$ **do**
6: 计算回报 $G_t^i = r_{t+1}^i + \gamma r_{t+2}^i + \ldots$
7: 计算更新量 $\Delta\theta = \Delta\theta + \alpha\nabla_\theta \log \pi_\theta(s_t^i, a_t^i) G_t^i$
8: **end for**
9: **end for**
10: 更新策略参数 $\theta \leftarrow \theta + \Delta\theta$
11: **until** 达到一定迭代次数或策略无明显的提升

Actor-Critic: Q 函数逼近器$Q_w(s,a)$为 Critic，策略逼近器$\pi_\theta(s,a)$为 Actor.Critic 更新价值函数的权重;Actor 利用 Q(降低方差)更新策略梯度。**Actor** 的策略梯度基于 Critic 定，故让β大一些,Critic 学的快一些.

1: 定义 Q 函数逼近器 $Q_w(s,a)$, 策略逼近器 $\pi_\theta(s,a)$, 初始化 $\mathbf{w}$, θ, $s_t = s_0$, $t = 0$
2: **repeat**
3: 采样动作 $a_t \sim \pi_\theta(s_t, a_t)$ 并执行, 观测 r_{t+1}, s_{t+1}
4: $\delta = r_{t+1} + \gamma Q_{\mathbf{w}}(s_{t+1}, a_{t+1}) - Q_{\mathbf{w}}(s_t, a_t)$
5: $\mathbf{w} \leftarrow \mathbf{w} + \beta\delta\nabla_{\mathbf{w}} Q_{\mathbf{w}}(s_t, a_t)$
6: $\theta \leftarrow \theta + \alpha\nabla_\theta \log \pi_\theta(s_t, a_t) Q_{\mathbf{w}}(s_t, a_t)$
7: $t \leftarrow t + 1$
8: **until**

AC 算法 Q 和 pi 满足条件则收:Q 和 pi 兼容；Q 的权重 w 等于真实Q_{π_θ}的最小均方误差解

策略梯度引入基准：在回报上减去一个 baseline 不改变梯度. b 值可以方差降到最低。对于 episodic:b=V_0.连续 b(s_t)=$V_{\pi_\theta}(s_t)$。**优势函数:** $A_{\pi_\theta}(s,a) = Q_{\pi_\theta}(s,a) - V_{\pi_\theta}(s)$.

1: 定义 V 函数逼近器 $V_w(s)$, 策略逼近器 $\pi_\theta(s,a)$, 初始化 $\mathbf{w}$, θ, $s_t = s_0$, $t = 0$
2: **repeat**
3: 采样动作 $a_t \sim \pi_\theta(s_t, a_t)$ 并执行, 观测 r_{t+1}, s_{t+1}
4: $\delta = r_{t+1} + \gamma V_{\mathbf{w}}(s_{t+1}) - V_{\mathbf{w}}(s_t)$
5: $\mathbf{w} \leftarrow \mathbf{w} + \beta\delta\nabla_{\mathbf{w}} V_{\mathbf{w}}(s_t)$
6: $\theta \leftarrow \theta + \alpha\delta\nabla_\theta \log \pi_\theta(s_t, a_t)$
7: $t \leftarrow t + 1$
8: **until**

自然梯度:最速上升方向不再根据欧式距离决定，而根据 J 的策略概率分布π决定.KL 散度在概率分布空间上描述两个分布的距离.

确定型 AC:以上算法为有限动作集 MDPs 即 a~ $\pi_\theta(s)$.本算法的策略则是确定性的 a=$\pi_\theta(s)$.动作的好坏又 Q(s, $\pi_\theta(s)$)反应.

1: 定义 Q 函数逼近器 $Q_w(s,a)$, 策略逼近器 $\pi_\theta(s,a)$, 初始化 $\mathbf{w}$, θ, $s_t = s_0$, $t = 0$
2: **repeat**
3: 采样动作 $a_t = \pi_\theta(s_t) + \mathcal{N}_t$ 并执行, 观测 r_{t+1}, s_{t+1}
4: 计算 TD 误差: $\delta = r_{t+1} + \gamma Q_{\mathbf{w}}(s_{t+1}, \pi_\theta(s_{t+1})) - Q_{\mathbf{w}}(s_t, a_t)$
5: 更新 Critic: $\mathbf{w} \leftarrow \mathbf{w} + \beta\delta\nabla_{\mathbf{w}} Q_{\mathbf{w}}(s_t, a_t)$
6: 更新 Actor: $\theta \leftarrow \theta + \alpha\nabla_a Q_{\mathbf{w}}(s_t, a)\big|_{a=\pi_\theta(s_t)} \nabla_\theta \pi_\theta(s_t)$
7: $t \leftarrow t + 1$
8: **until**