二、MDPs

<Markov process / Markov chain>.

时间，状态都离散的马尔可夫过程 → 马尔可夫链.

马尔可夫过程（链）由元组 (S, P) 组成
- S 为有限状态的集合.
- P 为状态转移矩阵 $P_{ss'} = P[S_{t+1} = s' \mid S_t = s]$

（大写的 S_t 为随机状态，小写 s 已确定状态），　转移矩阵 P（$|S| \times |S|$ 阶方阵）

<Episodes（幕）.>

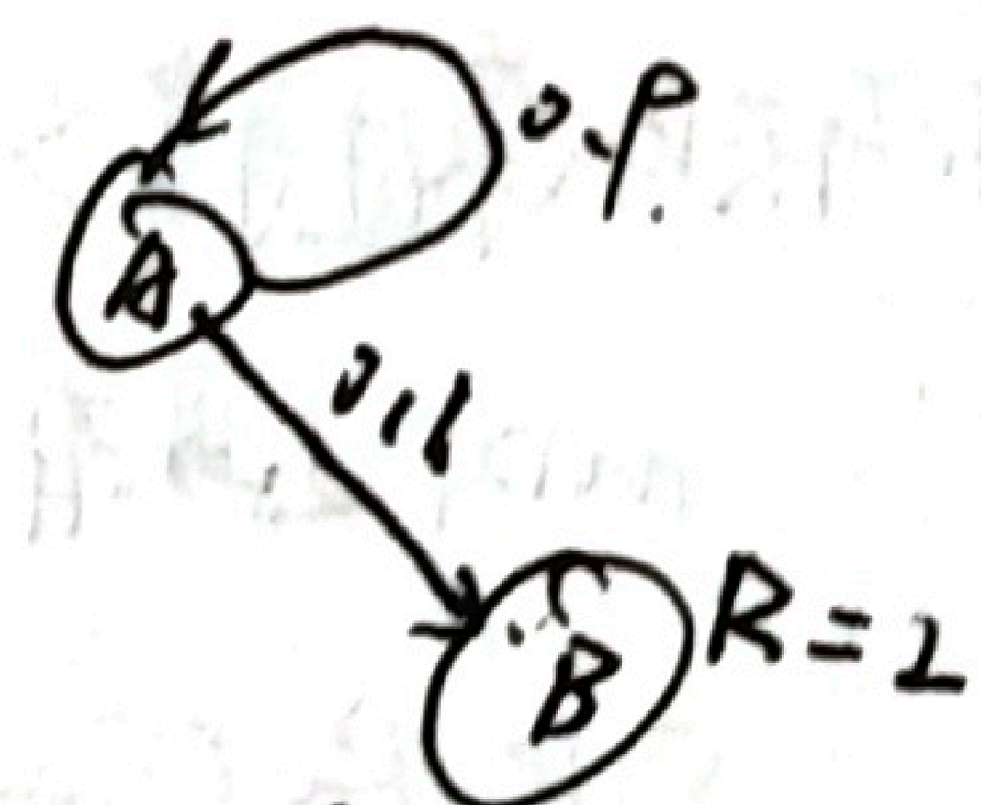
从初始状态 $S_1 = C_1$ 开始，从马尔可夫链中采取子序列，并终止于终止状态，那组个子序列就叫做 Episodes.

<马尔可夫奖励过程>

Markov Reward Process (MRP) 由元组 (S, P, R, γ) 构成.

S、P 在 Markov chain 中已有，另外的
- R_s 为奖励函数，$R_s = E[R_{t+1} \mid S_t = s]$（去除了奖励的随机性）
- γ 为折扣因子，$\gamma \in [0, 1]$

<奖励函数> $R_s = E[R_{t+1} \mid S_t = s]$　$(t+1)$ 一般认为到达状态 s 之后，系统才给出 Reward，所以写为 $(t+1)$ 的下标，如果在某一 s，奖励没有随机性，如



那么 $R_s = R_{(t+1)} = 2$，图里这样记为 $R=2$.

<回报，return>

从 t 时刻的 S_t 开始，直至终止状态时，所有奖励的衰减之和 G_t 称为 Return.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}.$$

PPT 中称之为 accumulative discounted reward $\sum_{t=0} \gamma^t r_t$ ～ t=1 时有值，当前 s 的 reward 也要计算.

Return 针对某 1 条幕说明/计算.

<价值函数，Value Function>（只有 1 条 episodes 的 return 高，并不能说明这个状态好）.

价值函数 $v(s)$ 给出状态 s 的长期价值（Long-term value）.

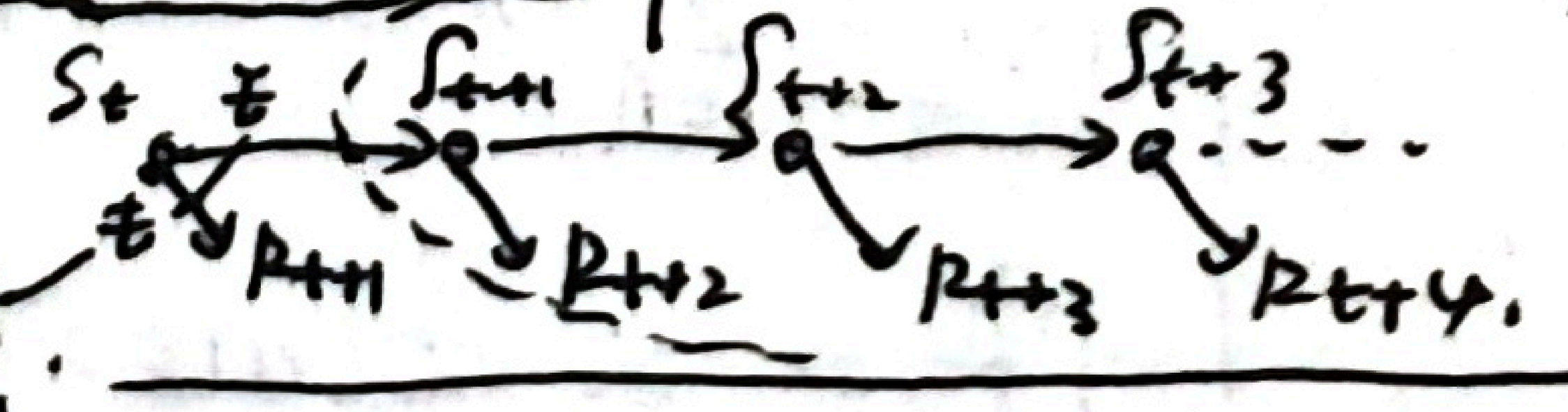
input：状态　output：价值.

在 MRP 中，一个 s 的期望回报为这个 s 的价值函数.

$$v(s) = E[G_t \mid S_t = s]$$

如何计算？
- 去除了 Reward 随机性
- 去除了状态转移的随机性.

<贝尔曼方程，Bellman Equation>.



消除随机性

$$v(s) = E[G_t \mid S_t = s].$$
$$= E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s]$$
$$= E[R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \dots) \mid S_t = s]$$
$$= E[R_{t+1} + \gamma G_{t+1} \mid S_t = s]$$

相当于从第一步开始到全部期望.

↓

前置去除随机 ⟹ $E[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s]$.

$$\Rightarrow v(s) = E[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s]$$

递归.

$v(s) = E[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s]$.

瞬时奖励　　后继状态的折扣值.

The other form of Bellman Equation:

$$v(s) = R_s + \gamma \sum_{s' \in S_{t+1}} P_{ss'} v(s')$$

$\sum P_{ss'}$ → 转移函数

$v(s')$ → 下一个状态的价值函数

γ → 折扣因子

R_s → 当前状态的奖励.

<贝尔曼方程的矩阵形式> ↓ 贝尔曼方程为线性方程.

$$\vec{v} = R + \gamma P \vec{v}$$

⇓

$$\begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix} = \begin{bmatrix} R_1 \\ \vdots \\ R_n \end{bmatrix} + \gamma \begin{bmatrix} P_{11} & \cdots & P_{1n} \\ \vdots & & \vdots \\ P_{n1} & \cdots & P_{nn} \end{bmatrix} \begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix}$$

⇓

$$\vec{v} = (I - \gamma P)^{-1} R$$

计算复杂度为 $O(|S|^3)$

直接求解应用于小型马尔可夫奖励过程.

<马尔可夫决策过程> (MDP).

MDP是具有决策性质的MRP, 由元组 (S, A, P, R, γ) 组成.

S, P, R, γ 在MRP中已有提及, A 为有限动作的集合.

但是 P 和 R_s 有所改变

$$P^a_{ss'} = P[S_{t+1} = s' \mid S_t = s, A_t = a]$$

$$R^a_s = E[R_{t+1} \mid S_t = s, A_t = a].$$

不只有系统随机性决定, 还与action有关系.

<策略, Policy>.

Policy 完全定义了智能体的行为.

马尔可夫决策过程中, Policy 仅取决于当前状态.

input: s　　output: a 动作

$$\pi(a|s) = P[A_t = a \mid S_t = s].$$

在组合下, 又可以写成列向量 → $\begin{bmatrix} \pi(a_1|s) \\ \vdots \\ \pi(a_n|s) \end{bmatrix}$ ⇒ 策略!

给定某个状态的不同policy组合

有了 $\pi(a|s)$, 即 Policy, P 和 R 发生变化, markov chain, MRP, MDP 都会发生变化:

$$P^{\pi}_{ss'} = \sum_{a \in A} \pi(a|s) P^a_{ss'}$$

$$R^{\pi}_s = \sum_{a \in A} \pi(a|s) R^a_s$$

<状态价值函数 (State-Value)>. ← 对特定的π，可评估策略.

$$v_\pi(s) = E_\pi[G_t \mid S_t = s].$$

遵从策略π.

$v_\pi(s)$ 为从s出发遵循π得到的期望回报.

我得到"|"的后面，即"| $S_t = s$, π".

如何求解?

由于π的加入，$v_\pi(s)$ 可以区分"业余手"和"职业手"的区别，但是如果评估同一个人，就没有办法，因此引入：

<动作价值函数, action-Value>. ← 对同一个π → 可评估决策.

$$Q^\pi(s,a) = E_\pi[G_t \mid S_t = s, A_t = a].$$

遵从策略π

$Q_\pi(s,a)$ 为从s出发遵循π，对当前状态s执行a得到的期望回报.

<贝尔曼期望方程, Bellman Expectation Equation> Bellman Equation + action

$$v_\pi(s) = E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s]$$ ← a不确定，后续A也随机，

↳对动作消除随机，对 $S_t \to S_{t+1}$ 状态转移消除随机.

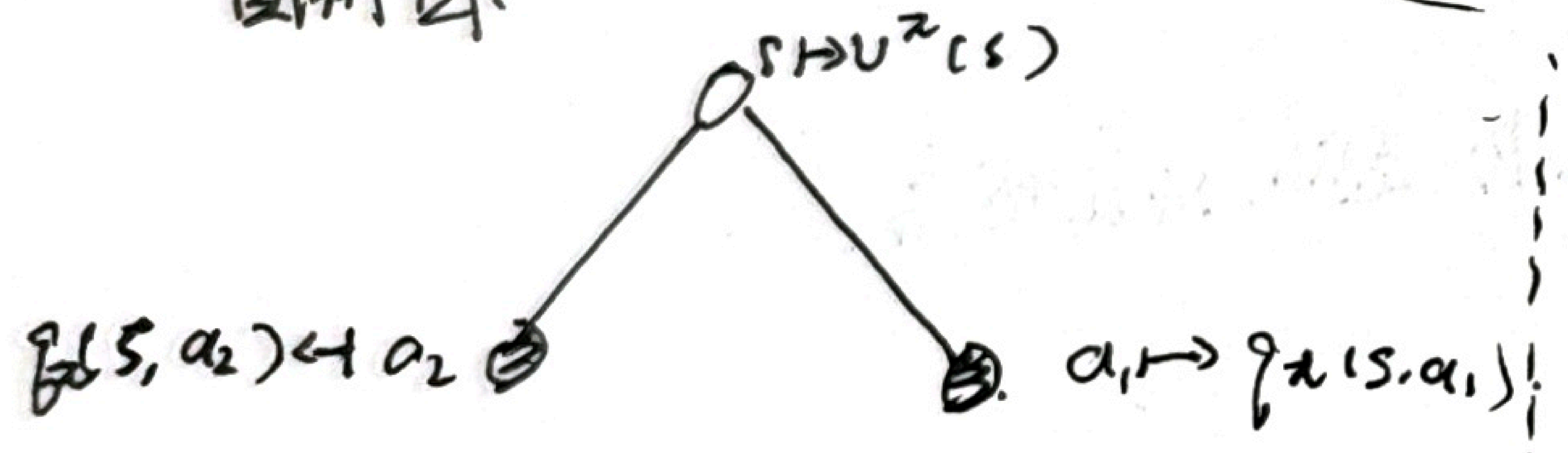
$$Q^\pi(s,a) = E_\pi[R_{t+1} + \gamma Q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a]$$ ← a确定，后续A随机.

Different

考虑回 A_t 的随机性

$v_\pi(s)$ 比 $Q_\pi(s,a)$ 少了 A_t 的随机性. ← 以为了a动作的随机性

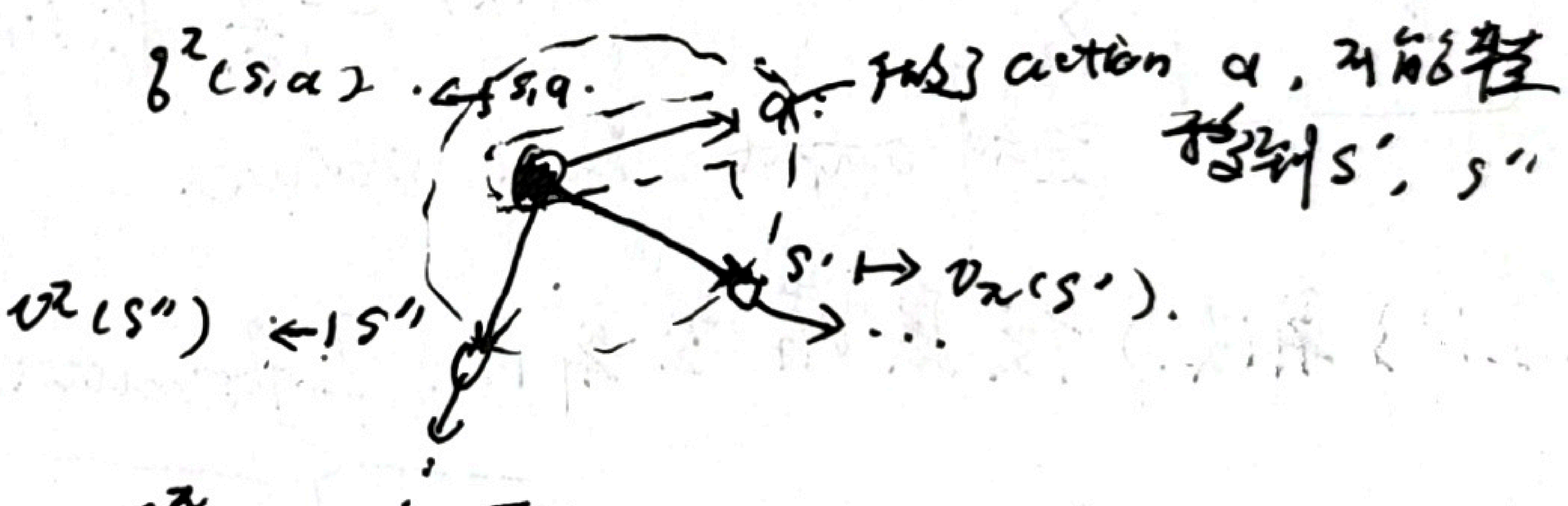
回溯图:



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

消除特定动作a的随机性

●可操节点，如

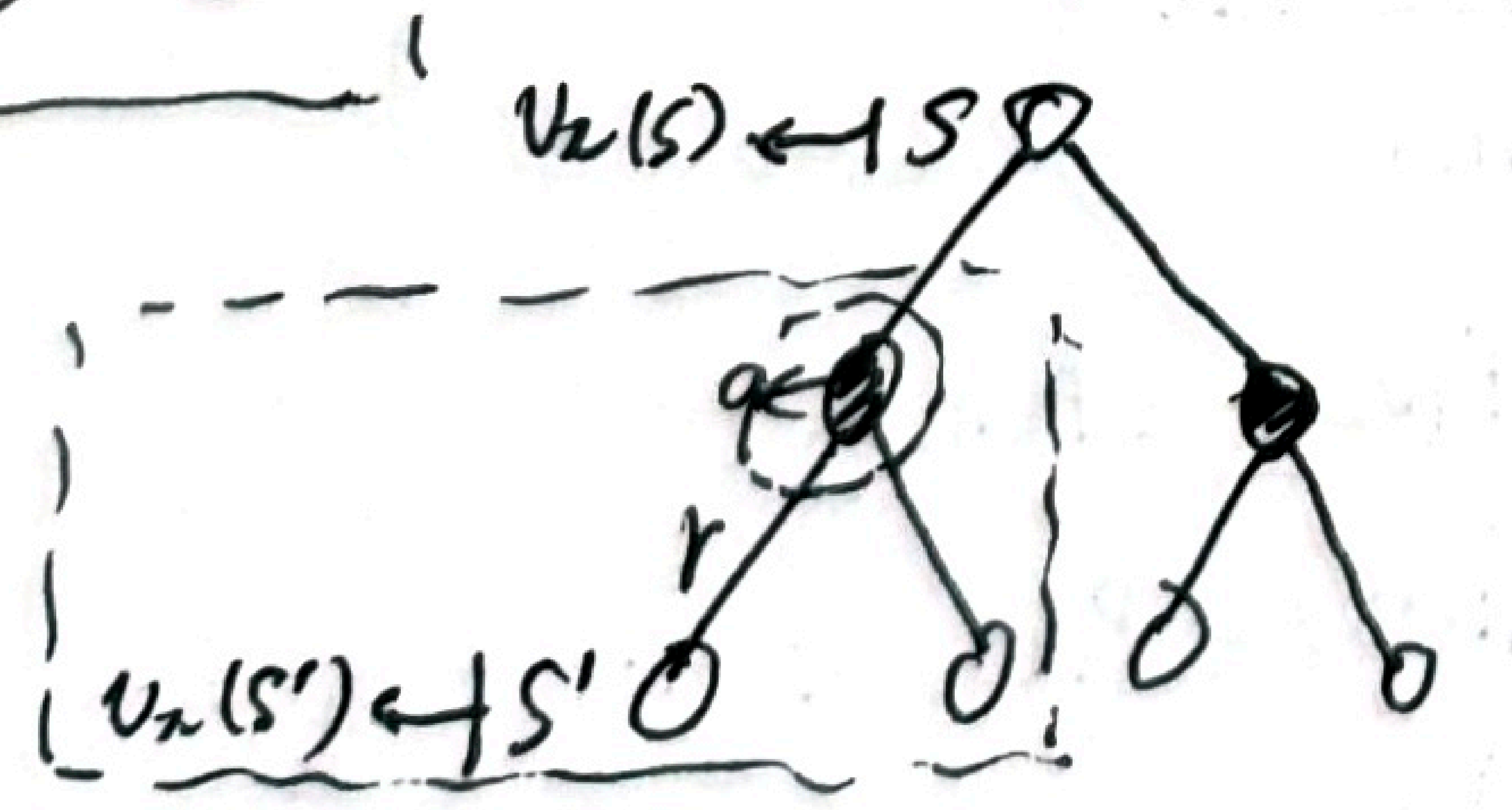


~~$q^\pi(s,a) = E[R_{t+1}$~~

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

无随机性.

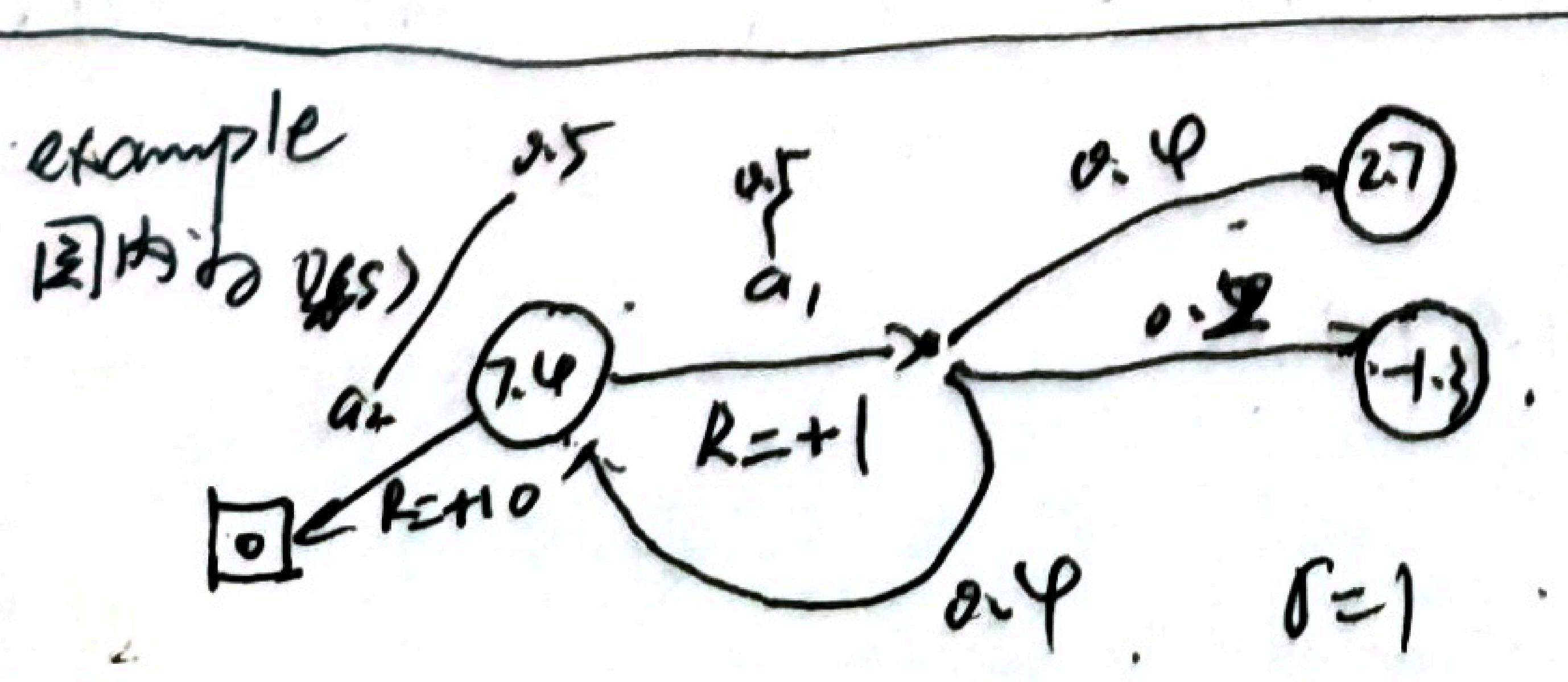
加上特定动作a的随机性，指定性.



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

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

example
图内的 v(s)



$$v_\pi(s) = 0.5\times(10+0) + 0.5\times(1 + 0.4\times 2.7 + 0.2\times(-1.3) + 0.4\times 7.4)$$

↑
7.4所指的，即7.4

<最优价值函数, Optimal Value Function>.

最终目标.

最优状态价值函数 $v_\pi^*(s) = \max_\pi v_\pi(s)$

最优动作价值函数 $q_\pi^*(s) = \max_\pi q_\pi(s, a)$.

都是对不同策略取最优.

最优价值函数知晓, MDP 已被求解.

Bellman equation:

~~$v_\pi^*(s) =$~~

$$Q_\pi^*(s,a) = E[R_{t+1} + \gamma \max_{a'} Q^*(s',a') \mid s,a]$$

<最优策略, Optimal Policy>.

<偏序> 对于任意的状态 s 都有 $v_\pi(s) \geq v_{\pi'}(s)$, $\pi \geq \pi'$.

在MDP中, 至少存在一个策略会不劣于其他策略, 即 $\pi_* \geq \pi, \forall \pi$, π_* → optimal policy

$$\pi_*(a|s) = \begin{cases} 1 & 若\ a = \arg\max_{a \in A} q_*(s,a) \\ 0 & \end{cases}$$

⇒ 只需最优的动作. 就要求解

arg ⇒ argument, ~~[illegible]~~ 自变量

<贝尔曼最优方程> (Bellman Optimality Equation). 非线性, 不能通过矩阵运算.

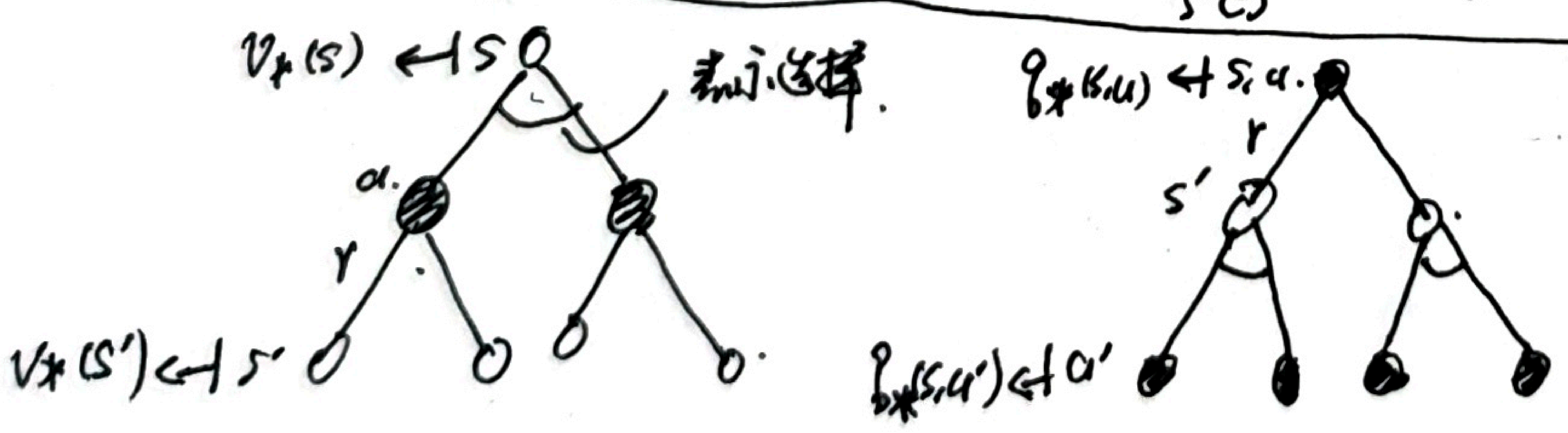
$$v_\pi(s) = \sum_{a \in A} \pi(a|s) q_\pi(s,a) = \max_a q_*(s,a) \mapsto v_\pi^*(s).$$

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

other form:

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

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



三、From Q-learning to Actor Critic.

$Q(S,A)$ 的求解：→ Sarsa 单步更新 $Q(S,A)$，策略评估，再 贪婪 优化。

↑ ε-greedy

<Sarsa> on-policy: S, A, R, S', ---, A → Sarsa.

$$q_\pi(s,a) = E_\pi[G_t \mid S_t=s, A_t=a]$$

$$= E_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t=s, A_t=a]$$

$Q(S,A) \xleftarrow{\text{更新}} Q(S,A) + \alpha(R + \gamma Q(S',A') - Q(S,A))$.

G_t.

~~用一个 agent 表示~~

在第一种中的 Sarsa 依赖于 Bellman 期望方程

1: 随机初始化 $Q(S,a)$, $\forall s \in S$, $a \in A(s)$. 令 $Q(\text{terminal-state}, \cdot) = 0$.
2: REPEAT (对于每个 episode):
3: 初始化 S.
4: 使用从 Q 得到的策略(eg. ε-greedy)在状态 S 时选择动作 A.
5: REPEAT(对于 episode 中的每一步)
6: 执行动作 A, 观察得到 R, S'
7: 根据从 Q 中得到策略 (eg. ε-greedy) 在状态 S' 时选择动作 A'.
8: $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma Q(S',A') - Q(S,A)]$
9: $S \leftarrow S'$; $A \leftarrow A'$.
10: 直到 S 为终止状态.

<On-policy, off-policy, 在轨, 离轨>.

<off-policy>.

agent 自身

其它主元素, 举例子.

目标策略: 用来学习的策略. 行为策略: 生成行为样本的策略.

计算目标策略 $\pi(a|s)$ 以计算 $V_\pi(s)$ 或 $q_\pi(s,a)$.

同时遵循行为策略 $\mu(a|s)$ $\{S_1, A_1, R_2, \ldots, R_T\} \sim \mu$

<Q-Learning> → off policy.

依赖于 Bellman 最优 equation

目标

G_t.

Q 更新的接口.

价到人生东西.

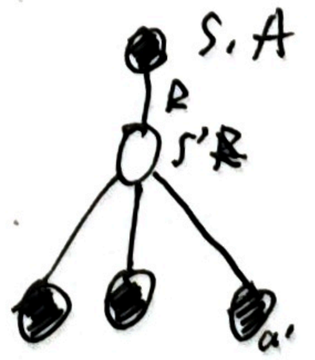
$$Q(S_t,A_t) \leftarrow Q(S_t,A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A') - Q(S_t,A_t))$$

$R_{t+1} + \gamma Q(S_{t+1}, A')$ 为人为的

agent 得到的. 其余下时步中的

行为策略和目标策略都随 Q 的更新而优化.

目标策略是贪婪的, 行为策略是 ε-greedy 的.



目标更新简化.

$$R_{t+1} + \gamma Q(S_{t+1}, A') \quad \text{贪婪}$$

$$= R_{t+1} + \gamma Q(S_{t+1}, \arg\max_{a'} Q(S_{t+1}, a'))$$

$$= R_{t+1} + \max_{a'} \gamma Q(S_{t+1}, a')$$

Q-Learning:

$$Q(S,A) \leftarrow Q(S,A) + \alpha(R + \gamma \max_{a'} Q(S',a') - Q(S,A))$$

Sarsamax (Q-Learning).

随机初始化 $Q(s,a), \forall s \in S, a \in A(s)$, 将 $Q(terminal\text{-}state, \cdot) = 0$.
REPEAT (对于每一个 episode):
  初始化 S.
  REPEAT (对于 episode 中每一步):   ← 不需要自己选择动作.
    根据从 Q 得到的策略 (eg. ε-greedy) 在 S 时选择 A.
    执行 A, 观察到 R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_a (Q', A') - Q(S,A)]$ (A': 未确定, 因为操作者是别人.)
    S ← S'
  直到 S 为终止状态.

我更新了之后跟操作者共同优化Q.

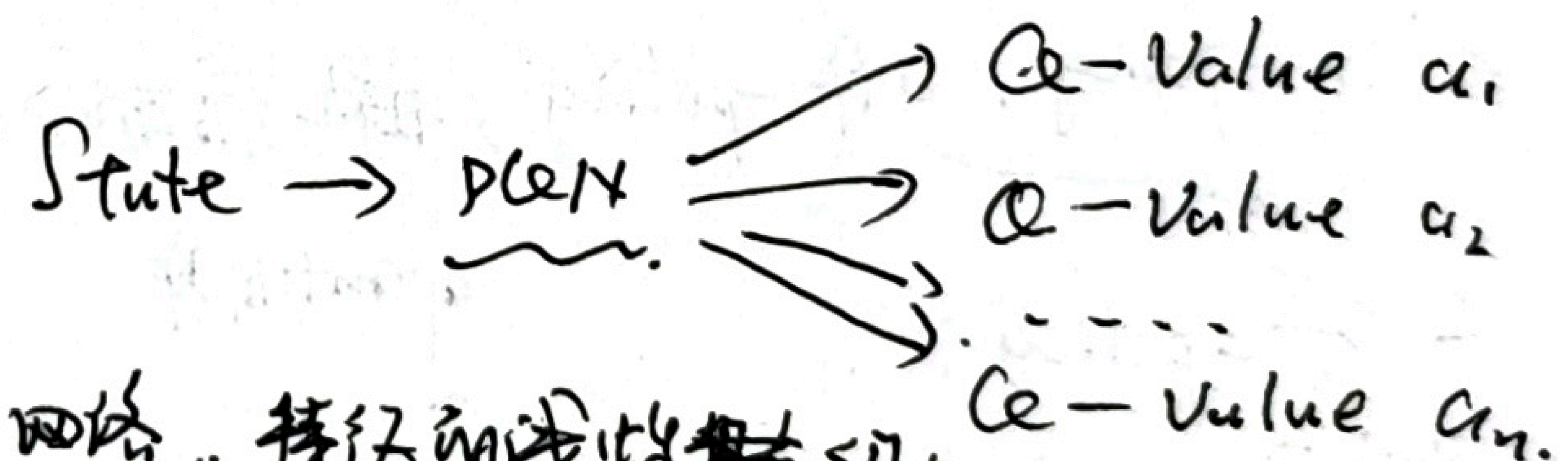
Sarsa 走远路, 采取总收益最高的路径.
Q-Learning 走近路, 总收益不如 Sarsa, 正所谓富贵险中求!

<DQN>

如果S个数少, 上述方法可用来求解其它优化类方法, 但是有无限状态呢?
以前的过程会建立一个 Q-table, 学习的过程就是填表的过程, 但是太多的数据, 无法储存. 而且并不是所有状态/动作可见.
解决方法: (用函数逼近价值函数, input → s,a, output Q).

$\hat{v}(S,w) \approx v_\pi(S)$ or $\hat{q}(S,a|w) \approx q_\pi(s,a)$.

input
↓
system
↓
output.



考虑可微分的函数逼近器 —— 神经网络, 特征的线性组合.
此外, 需要适用于非平稳非独立分布的数据训练方法. (QL中的数据特征)

<求解最优点>.

线性最小二乘: 矩阵求解
非线性最小二乘: <Gradient Descent>.

While True.
  $\nabla_w L(w) = \nabla_w \frac{1}{N}\sum L_i(f(x_i, w), y_i)$.   → 样本过大, 使用SGD优化.
  $w \leftarrow w - \alpha \nabla_w L(w)$ → $\Delta w = \alpha(y_i - f(x_i,w))\frac{\partial f}{\partial w} = -\frac{1}{2}\alpha\nabla_w L(w)$ ⇒ $w \leftarrow w + \Delta w$.

L(w) ↓ 求导.

GD, SGD, MBGD, GD with Momentum.   (折中.)

$\Delta w = -\frac{1}{2}\alpha \nabla_w J(w) = \alpha E_\pi[(v_\pi(S) - \hat{v}(S,w)) \nabla_w \hat{v}(S,w)]$

以SGD采样 $\Delta w = \alpha[(v_\pi(S) - \hat{v}(S,w)) \nabla_w \hat{v}(S,w)]$.

状态 S 可以由特征向量表示.