Value Based - Part 1

Deep Reinforcement Learning

Mohammad Hossein Rohban, Ph.D.

Courtesy: Some slides are adopted from CS 285 Berkeley, and CS 234 Stanford, and Pieter Abbeel's compact series on RL.

RIML Lab, CE Department, Sharif University of Technology

Spring 2026

Value Function

traject. $\longrightarrow$ $S_0, a_0, S_1, R(S_0, a_0, S_1), a_1, S_2, \cdots =$

- $V^*(s)$ = expected utility starting in s, and acting optimally in all subsequent actions.

Obj.: $\max_{a_0, a_1, \ldots} \mathbb{E}\left\{\sum \gamma^t R(s_t, a_t, s_{t+1})\right\}$ (Return)

$$V^*(s) = \max_{a_i' s} \mathbb{E}\left(\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) \,\middle|\, s_0 = s\right)$$

arg; discount factor ≤ 1; reward; Return

state trans. stoch.

Goal: How to approx. val. function for all s.

$\pi : S \longrightarrow A$

Value Iteration

- $V_0^*(s)$ = optimal value for state s when H=0
  - $V_0^*(s) = 0 \forall s$
- $V_1^*(s)$ = optimal value for state s when H=1
  - $V_1^*(s) = \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_0^*(s'))$
- $V_2^*(s)$ = optimal value for state s when H=2
  - $V_2^*(s) = \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_1^*(s'))$
- $V_k^*(s)$ = optimal value for state s when H = k
  - $V_k^*(s) = \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_{k-1}^*(s'))$

$\{ P(s_{t+1}|s_t, a_t)$ known
$\{ R(s_t, a_t, s_{t+1})$ known

Horizon

$V_k^*(s)$:= max possible return in at most ∞ $\leftrightarrow$ K steps starting @ state s.

$V^*(s) = V_\infty^*(s)$

* For the $V_k(s)$ to be optimal, the reward to go should also be optimal.

$$\begin{bmatrix} V_0^*(s_1) \\ \vdots \\ V_0^*(s_n) \end{bmatrix} \xrightarrow{\mathcal{T}} \begin{bmatrix} V_1^*(s_1) \\ \vdots \\ V_1^*(s_n) \end{bmatrix} \xrightarrow{\mathcal{T}} \cdots \begin{bmatrix} V_k^*(s_1) \\ \vdots \\ V_k^*(s_n) \end{bmatrix}$$

Value Iteration

Algorithm:

Start with $V_0^*(s) = 0$ for all s.

For k = 1, ... , H:

For all states s in S:

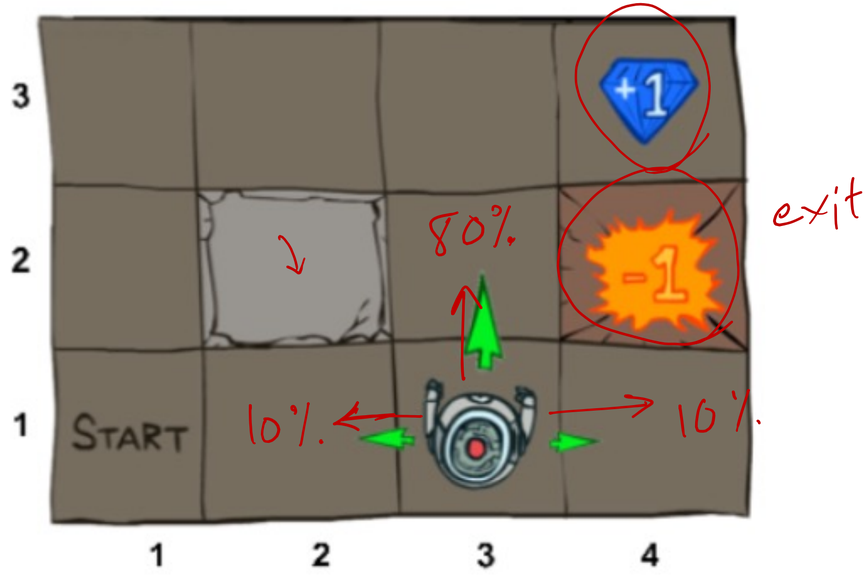
$$V_k^*(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_{k-1}^*(s'))$$

$$\pi_k^*(s) \leftarrow \arg\max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_{k-1}^*(s'))$$

- This is called a value update or Bellman update/back-up

Value Iteration

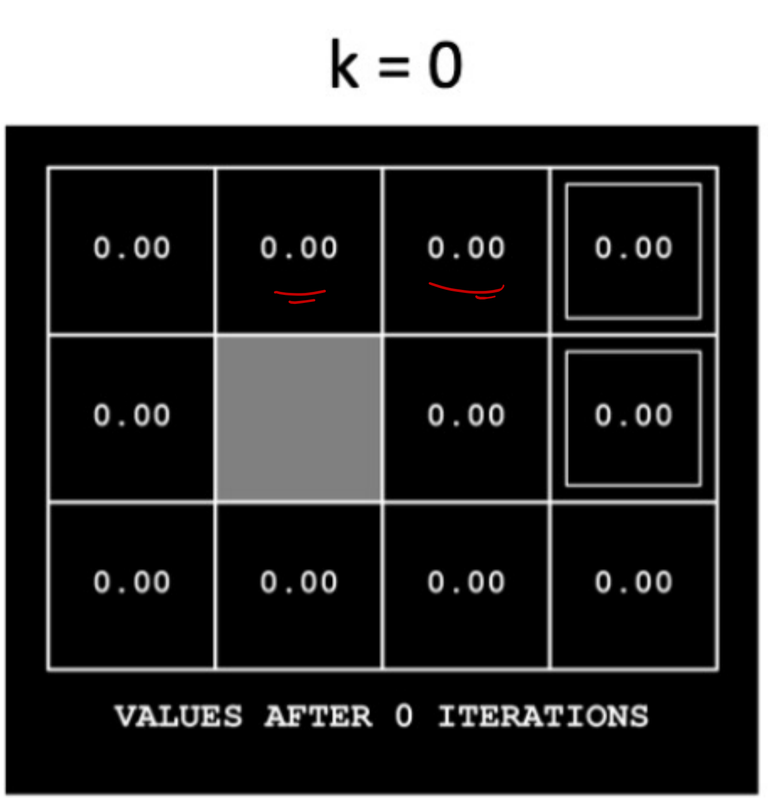
$$V_0(s) \leftarrow 0$$



Noise = 0.2

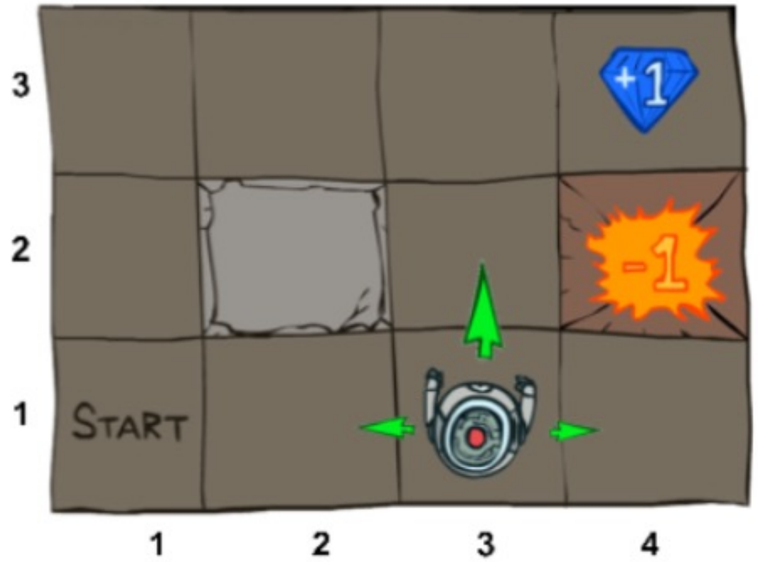
Discount = 0.9

$\gamma =$



Value Iteration

$$V_1(s) = \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_0(s'))$$



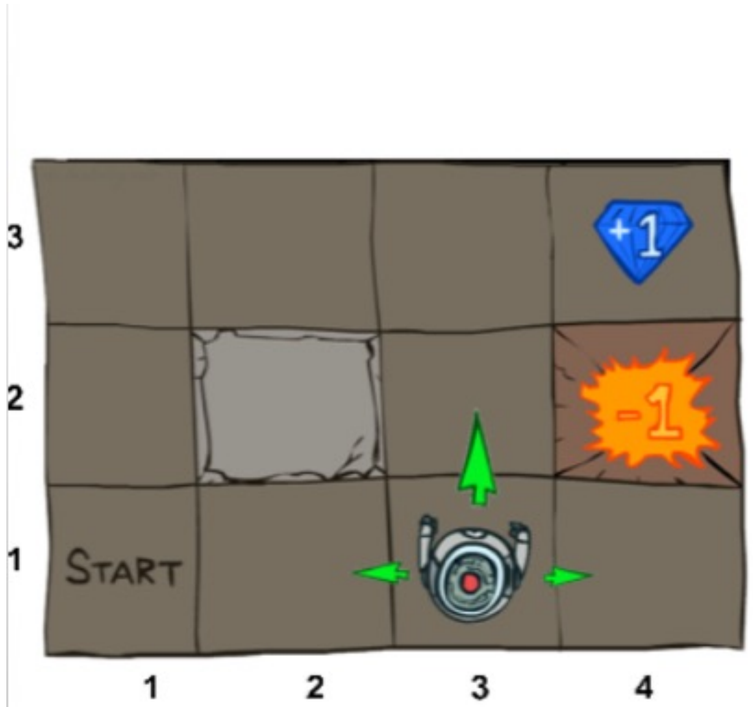
Noise = 0.2

Discount = 0.9

k = 0

Value Iteration

$$V_2(s) = \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_0(s'))$$



Noise = 0.2

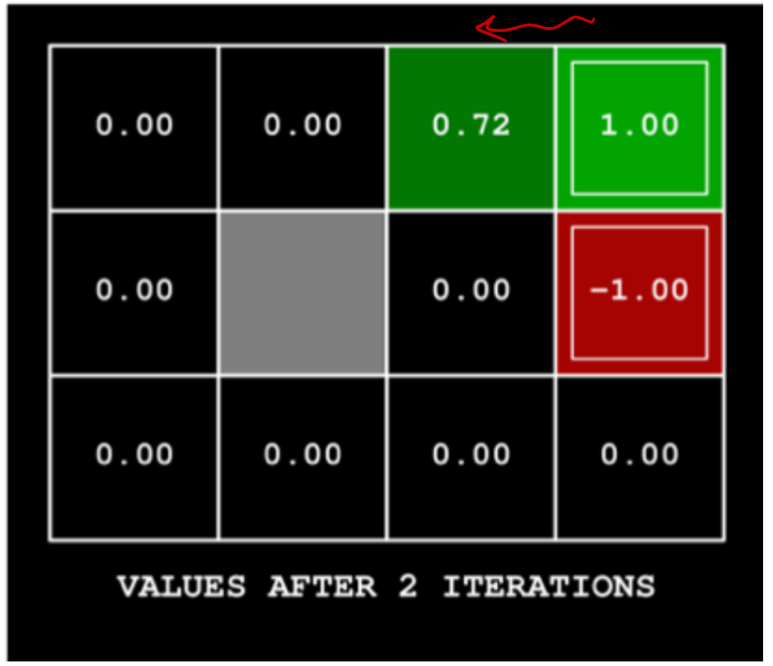
Discount = 0.9

Value Iteration

$$V_2(s) = \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_0(s'))$$

Propagation

k = 2

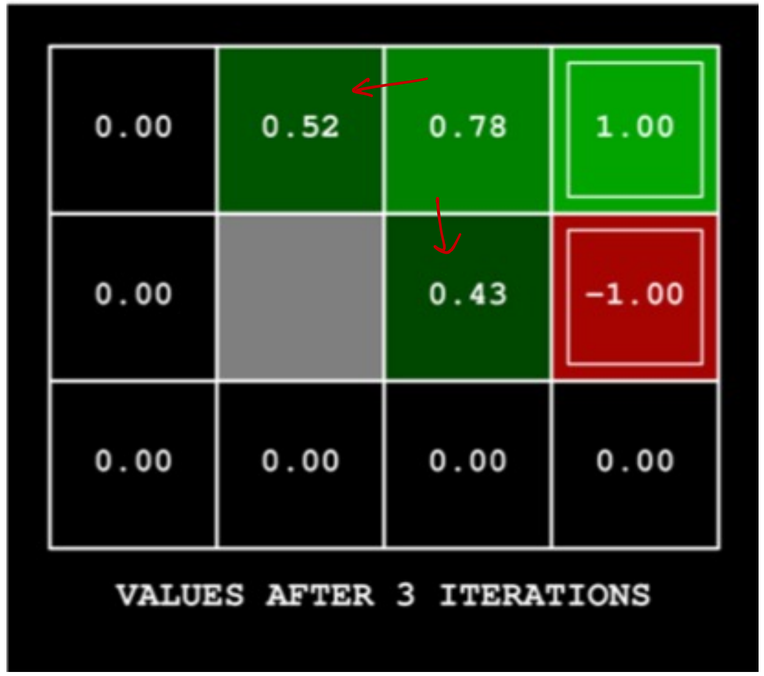


Noise = 0.2
Discount = 0.9

Value Iteration

$$V_{k+1}(s) = \max_{a} \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_0(s'))$$

k = 3



Noise = 0.2
Discount = 0.9

Value Iteration

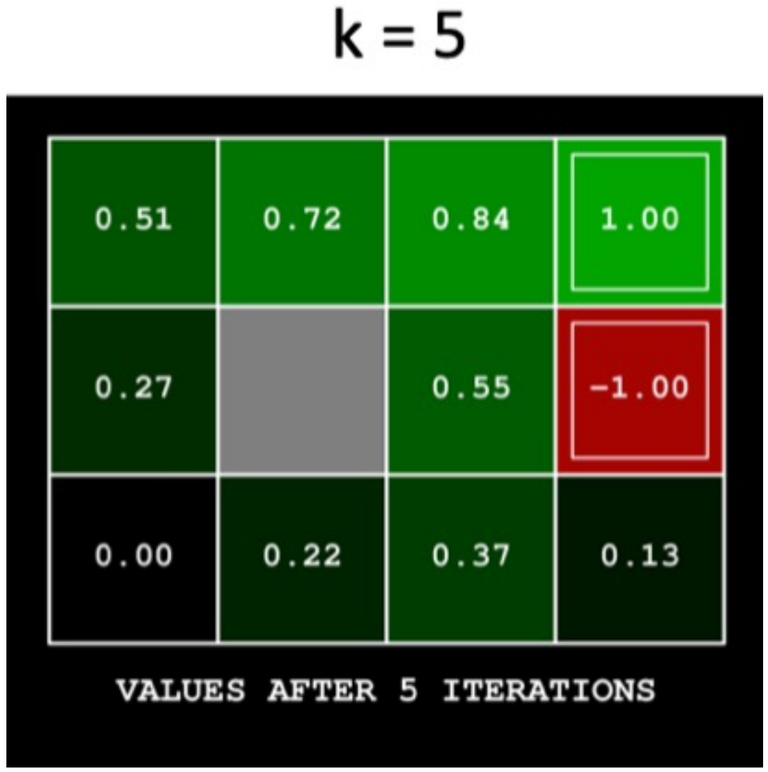
$$V_{k+1}(s) = \max_{a} \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_0(s'))$$

k = 4

Noise = 0.2
Discount = 0.9

Value Iteration

$$V_{k+1}(s) = \max_{a} \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_0(s'))$$



Noise = 0.2
Discount = 0.9

Value Iteration

$$V_{k+1}(s) = \max_{a} \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_0(s'))$$

k = 6

Noise = 0.2
Discount = 0.9

Value Iteration

$$V_{k+1}(s) = \max_{a} \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_0(s'))$$

Noise = 0.2
Discount = 0.9

Value Iteration

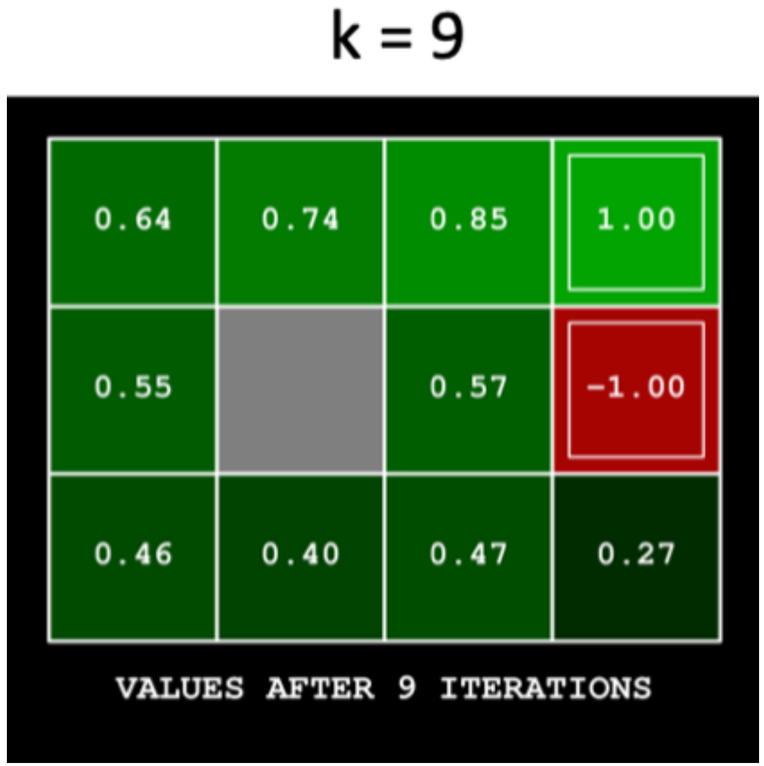
$$V_{k+1}(s) = \max_{a} \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_0(s'))$$

k = 8

Noise = 0.2
Discount = 0.9

Value Iteration

$$V_{k+1}(s) = \max_{a} \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_0(s'))$$



Noise = 0.2
Discount = 0.9

Value Iteration

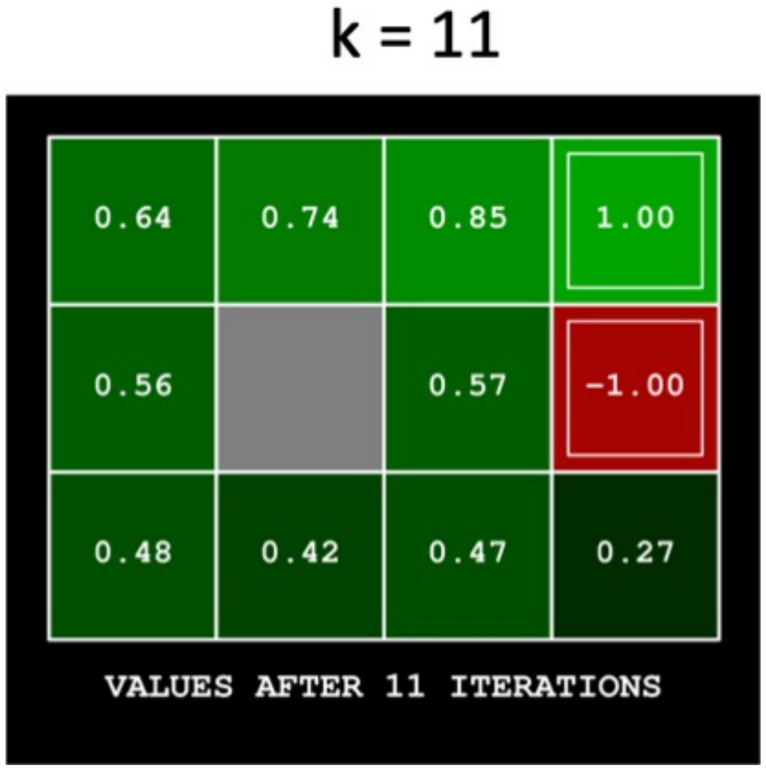
$$V_{k+1}(s) = \max_{a} \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_0(s'))$$

k = 10

Noise = 0.2
Discount = 0.9

Value Iteration

$$V_{k+1}(s) = \max_{a} \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_0(s'))$$



Noise = 0.2
Discount = 0.9

Value Iteration

$$V_{k+1}(s) = \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_0(s'))$$

Almost Converged

k = 12

Noise = 0.2
Discount = 0.9

Value Iteration

$$V_{k+1}(s) = \max_{a} \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_0(s'))$$

k = 100

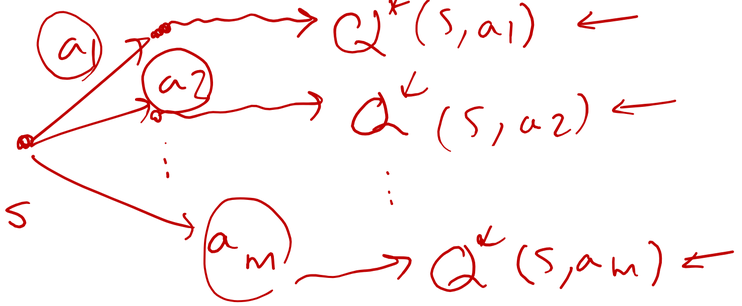
Noise = 0.2
Discount = 0.9

Q-Values

$\pi^*(s) = \arg\max_a Q^*(s,a)$

may be non-optimal

- $Q^*(s,a)$ = expected utility starting in s, taking action a, and (thereafter) acting optimally



- Bellman Equation:

$$Q^*(s,a) = \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma \max_{a'} Q^*(s',a'))$$

- Q-value Iteration:

$$Q^*_{k+1}(s,a) \leftarrow \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma \max_{a'} Q^*_k(s',a'))$$

$Q_{k+1}\left(\begin{bmatrix}5\\1\end{bmatrix}\right)$

$Q_{k+1}\left(\begin{bmatrix}6\\2\end{bmatrix}\right)$

$k \to \infty$

$Q_k\left(\begin{bmatrix}5\\1\end{bmatrix}\right)$ ✓

$Q_{k+1}\left(\begin{bmatrix}5\\1\end{bmatrix}\right)$ ✗

Q-Value Iteration

noise = 20% → 40%

$$Q^*_{k+1}(s,a) \leftarrow \sum_{s'} P(s'|s,a)\left(R(s,a,s') + \gamma \max_{a'} Q^*_k(s',a')\right)$$

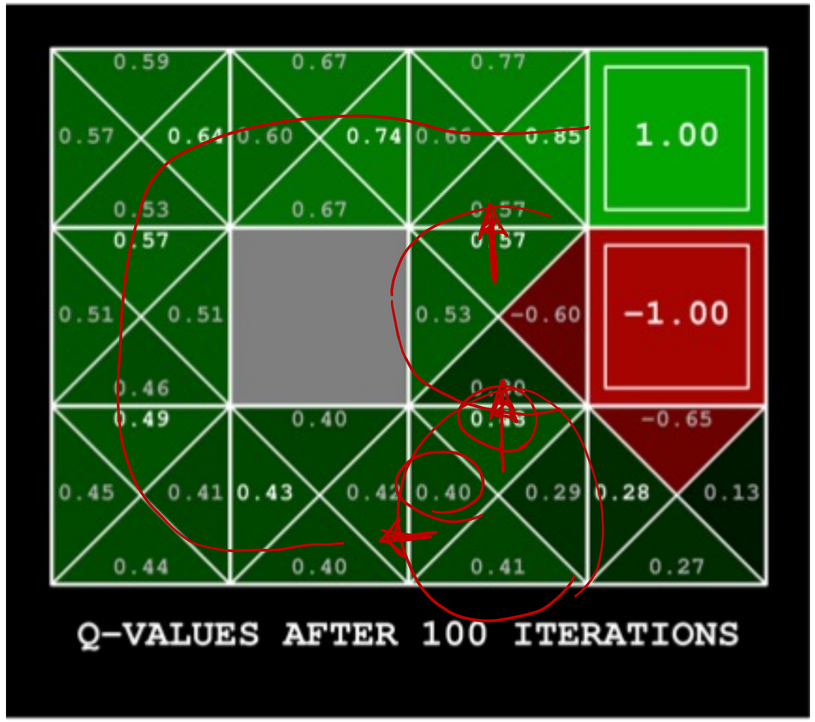
|state| < ∞

$R \leq C$

$\gamma < 1$

$x = f(x)$

k = 100



Alg. converges to

V^* or Q^*

for all states

Noise = 0.2
Discount = 0.9

Policy Evaluation

- Recall value iteration:

  - $V_k^*(s) \leftarrow max_a \sum_{s'} P(s'|s,a)\big(R(s,a,s') + \gamma V_{k-1}^*(s')\big)$

- Policy evaluation for a given $\pi(s)$:

  - $V_k^\pi(s) \leftarrow \sum_{s'} P(s'|s,\pi(s))(R(s,\pi(s),s') + \gamma V_{k-1}^\pi(s'))$

- At convergence:

  - $\forall s V^\pi(s) \leftarrow \sum_{s'} P(s'|s,\pi(s))(R(s,\pi(s),s') + \gamma V^\pi(s))$

$P(s_{t+1} \mid s_t, a_t)$
$R(s_t, a_t, s_{t+1})$

→ Q^*, π^*, V^*

Val. Iteration: $Q^*_0 \to Q^*_1 \to \cdots \to Q^*_k \to \pi^*$

Policy Iteration: $\pi_0 \to \pi_1 \to \cdots \to \pi^*$

vs.

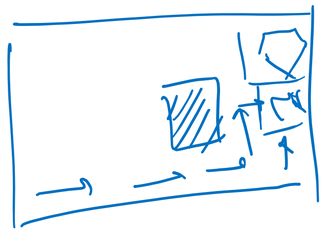
Eval.

What does eval. mean for a given policy π_0?

V^{π_0} $Q^{\pi_0}(s,a)$ = expected ret. when starting @ s, take a, then follow π_0 to choose actions!

eval. Policy Improv.

$Q^{\pi_1} \leftarrow \pi_1$ ← Critiscism of π_0 ← $Q^{\pi_0}(s,a)$



$\pi_1 \to \pi_2 \to \cdots$ impro.

(1)

$$Q^{\pi}_{K+1}(s,a) \leftarrow \sum_{s'} P(s'|s,a)\left[R(s,a,s') + \gamma Q^{\pi}_{k}(s', \pi(s'))\right]$$

$$\gamma \sum_{a} Q^{\pi}_{k}(s',a)\,\pi(a|s')$$

(2) Policy Improvement

$$\pi_1(s) = \arg\max_a Q^{\pi_0}(s,a)$$

Policy Iteration

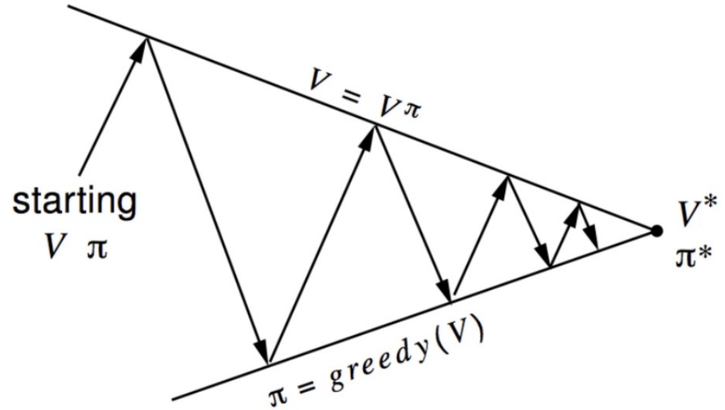
- One iteration of policy iteration
  - Policy evaluation for current policy π_k. Iterate until convergence

$$V_{i+1}^{\pi_k}(s) \leftarrow \sum_{s'} P(s'|s, \pi_k(s))[R(s, \pi_k(s), s') + \gamma V_i^{\pi_k}(s')]$$

  - Policy improvement: find the best action according to one-step look-ahead

$$\pi_{k+1}(s) \leftarrow \arg\max_a \sum_{s'} P(s'|s, a)[R(s, a, s') + \gamma V^{\pi_k}(s')]$$

- Repeat until policy converges
  - At convergence: optimal policy; and converges faster than value iteration under some conditions



One-step look ahead improves the policy

- Consider an alternative policy $\pi^{(1)}_{(k+1)}(t,s)$ that takes the prescribed actions in $\pi_{k+1}(s)$ only at time t = 0; and stays the same as $\pi_k(s)$ in later times.
- The value function V(s) for this new time-dependent policy is larger than or equal to V(s) for the original policy $\pi_k(s)$ for all s. Why?
- Now let $\pi^{(2)}_{(k+1)}(t,s)$, which takes the prescribed action in $\pi_{k+1}(s)$ only at times t = 0 and t = 1, and stays the same as $\pi_k(s)$ in later times.
- Similarly, V(s) gets improved for $\pi^{(2)}_{(k+1)}(t,s)$ compared to $\pi^{(1)}_{(k+1)}(t,s)$ for all s.
- Repeating this argument $\pi^{(\infty)}_{(k+1)}(t,s)$ becomes the same as $\pi_{k+1}(s)$.

An Example

noise = 0
γ = disc. = 1
Policy Eval.

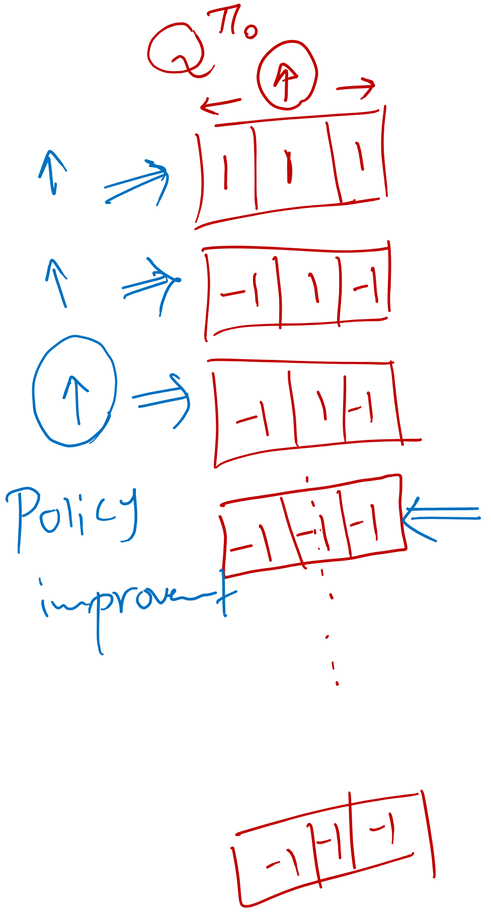
- Let this be the initial policy π_0, show how policy improvement, makes this policy better.

$P(s_{t+1}/s_t, a_t)$
R
π^*

Policy Itr.
Val. Itr.

π_0

+1	→	+1
-1	↑	-1
-1	→	-1
-1	↑ ↑	-1
-1	← ↑	-1
-1	↑ ↑	-1
-1	→ ↑	-1



Planning vs. Learning

↳ World Model ✗

- Assumed to have access to the dynamics $P(s'|s,a)$.
- We don't have access to this in the real world.
- We need to estimate (or learn) the value functions.

without access to this
sampling ← learning

$$Q^*(s,a) = \sum_{s'} P(s'|s,a)\left(R(s,a,s') + \gamma \max_{a'} Q^*(s',a')\right) \quad ?$$

$n \to \infty$

empirical mean

$$\frac{1}{n}\sum_{i=1}^{n} X_i \approx \mathbb{E}(X) = \int x \cdot f_X(x)\, dx$$

$$X_1 \dots X_n \overset{iid}{\sim} f_X \qquad = \sum_x x \cdot P(X = x)$$

Monte-Carlo Prediction

Control

Monte Carlo Methods - Introduction

- Experience samples to learn without a model

Experience = Data

MC Pred: Estimate the val. function based on the experienced episodes.

- MC methods require only experience— sample sequences of states, actions, and rewards from actual or simulated interaction with an environment.

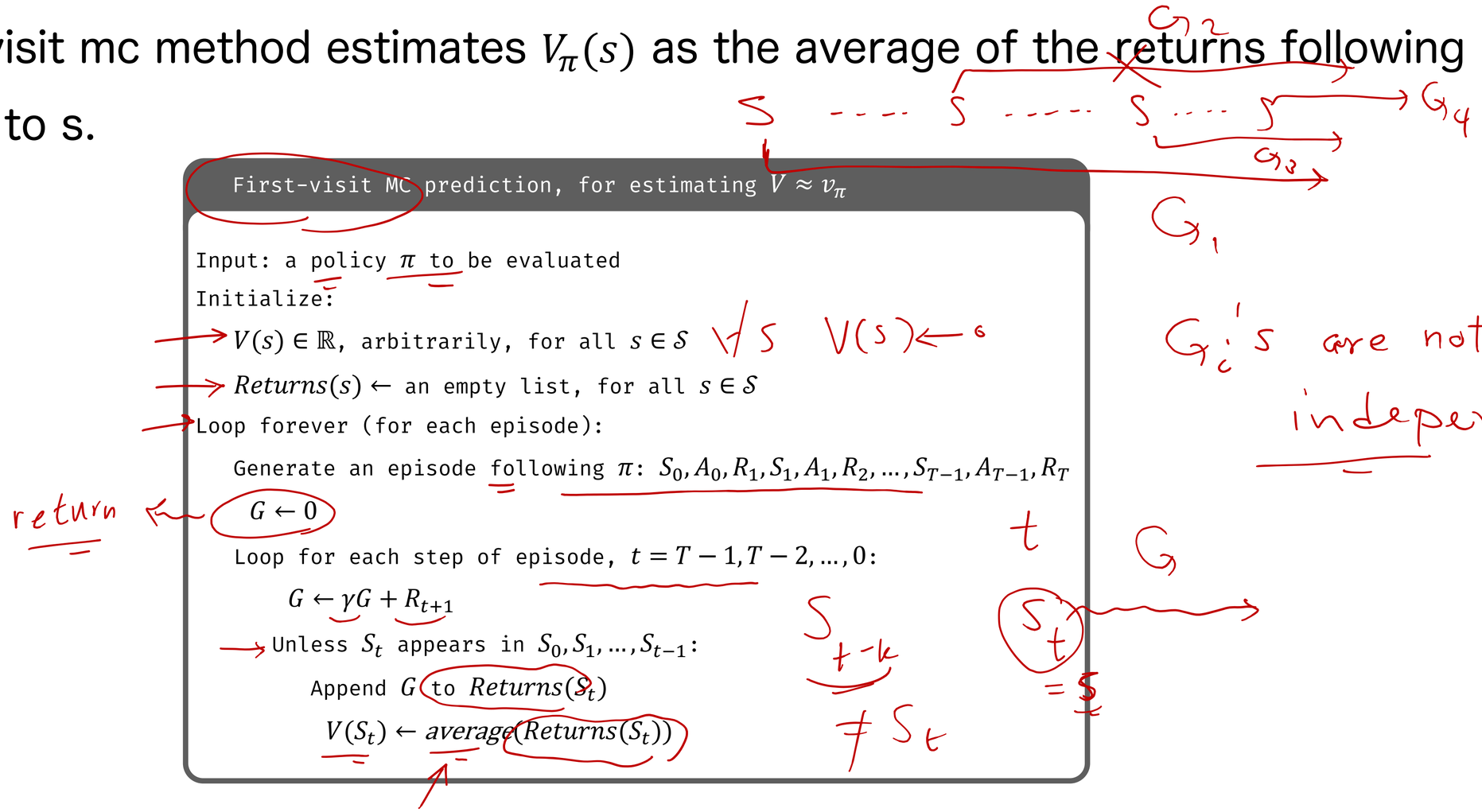
- We can learn with samples: episodes!

no access to World Model $P(s'|s,a)$

We don't have access to $P(s'|s,a)$ → Model Free Learning!

Monte-Carlo prediction

- Suppose we wish to estimate $V_\pi(s)$, the value of a state s under policy π.
- The first-visit mc method estimates $V_\pi(s)$ as the average of the returns following first visits to s.



First-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

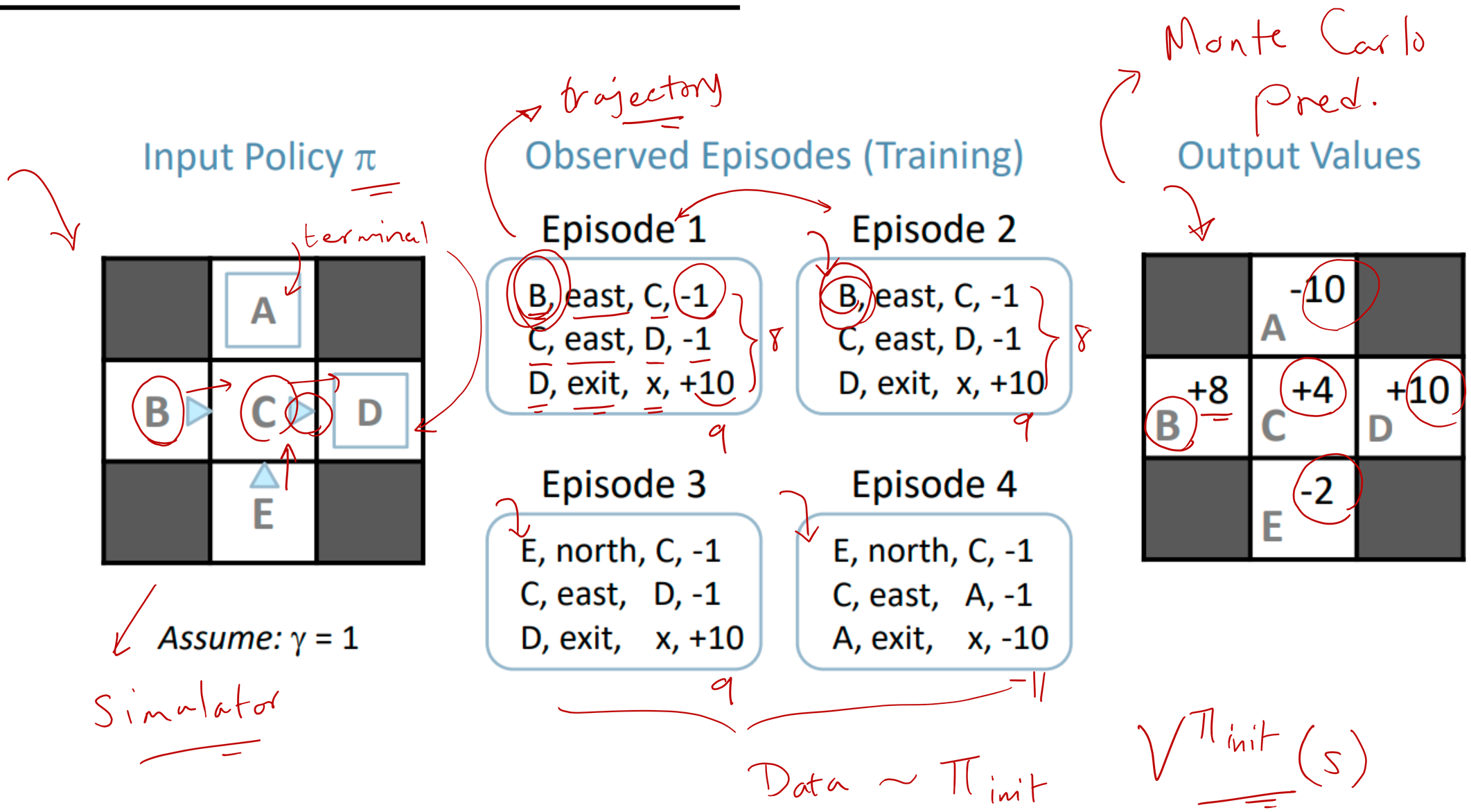
$G \leftarrow \gamma G + R_{t+1}$

Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$:

Append G to $Returns(S_t)$

$V(S_t) \leftarrow average(Returns(S_t))$

Episodes: another example



Every Visit Monte-Carlo Policy

Every Visit Monte-Carlo Policy

Initialize $N(s) = 0, G(s) = 0 \forall s \in S$

Loop

Sample episode $i = s_{i,1}, a_{i,1}, r_{i,1}, s_{i,2}, a_{i,2}, r_{i,2}, \ldots, s_{i,T_i}$

Define $G_{i,t} = r_{i,t} + \gamma r_{i,t+1} + \gamma^2 r_{i,t+2} + \cdots \gamma^{T_i - 1} r_{i,T_i}$ as return from time step t onwards in i'th episode.

For each time step t until T_i (the end of the episode i):

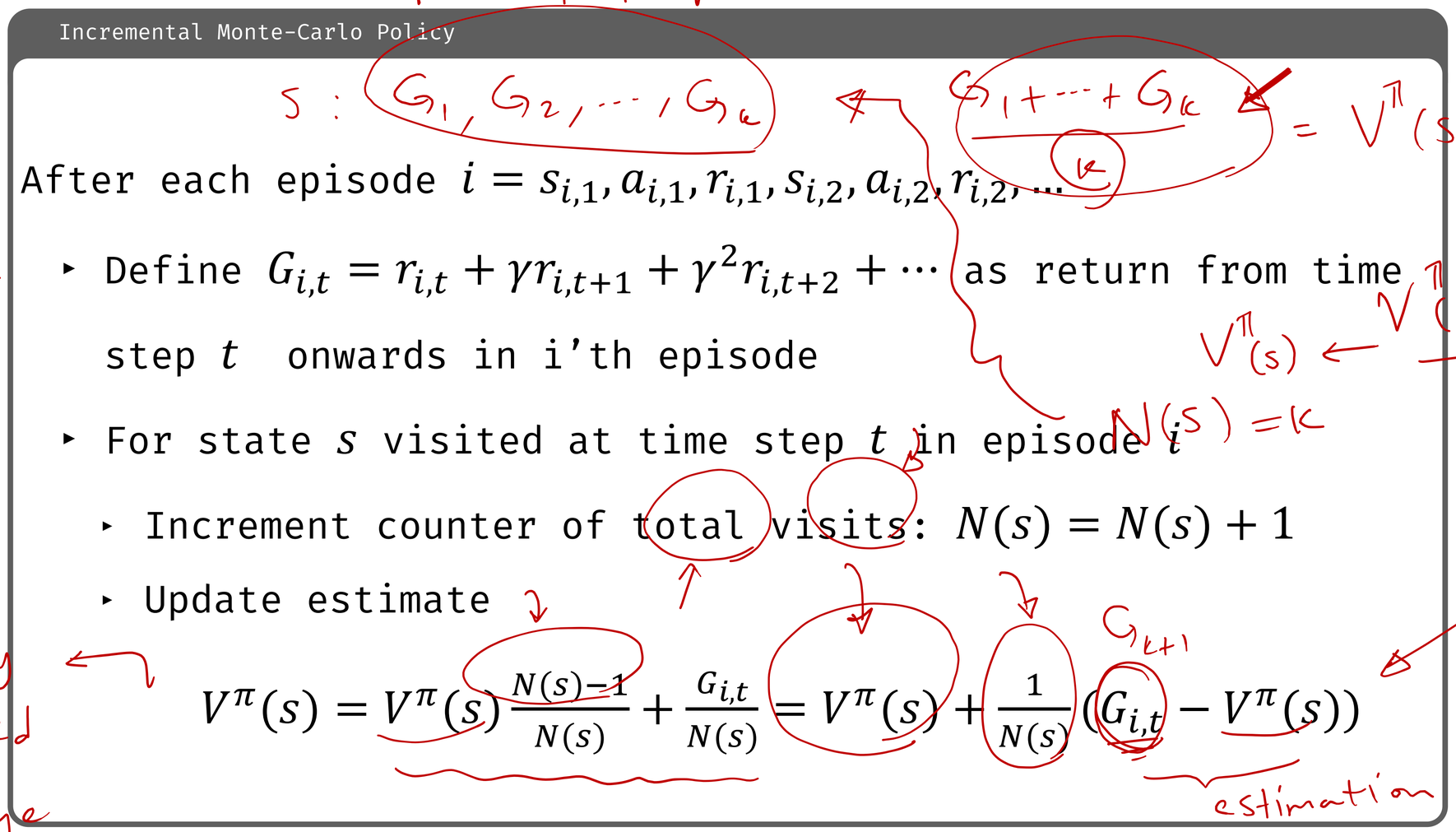
state s is the state visited at time step t in episodes i

Increment counter of total visits: $N(s) = N(s) + 1$

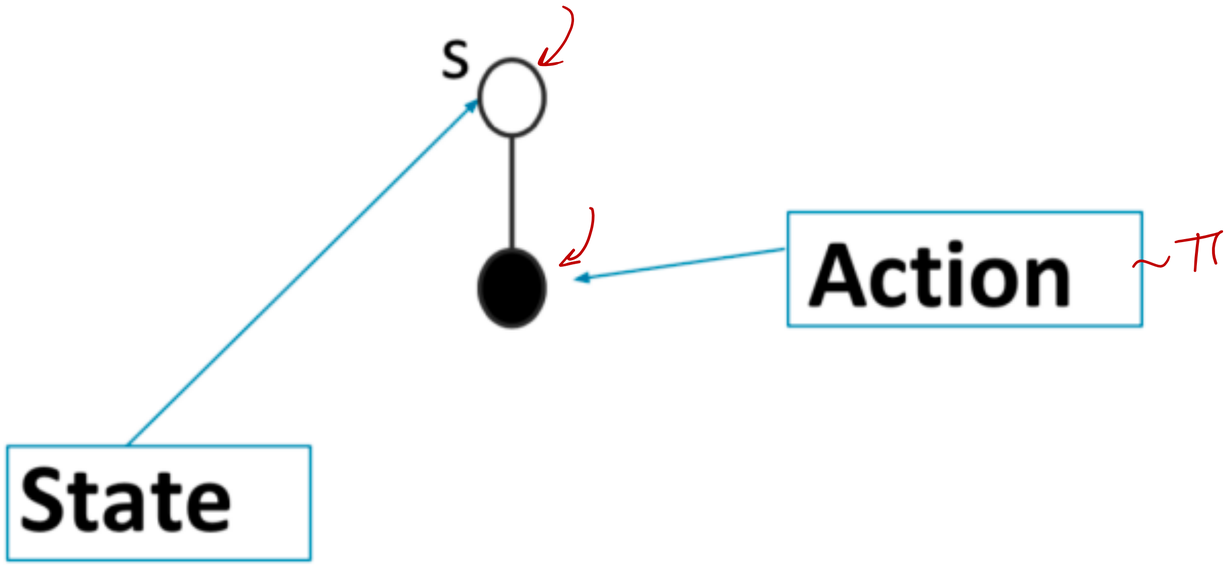
Increment total return: $G(s) = G(s) + G_{i,t}$

Update estimate $V^{\pi}(s) = G(s)/N(s)$

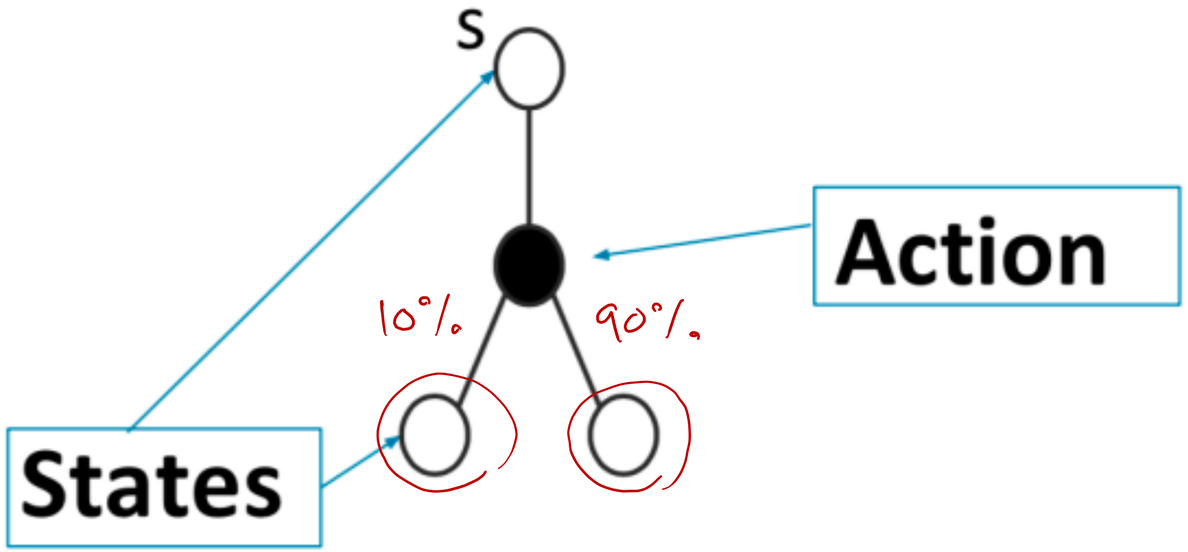
Incremental Monte-Carlo Policy



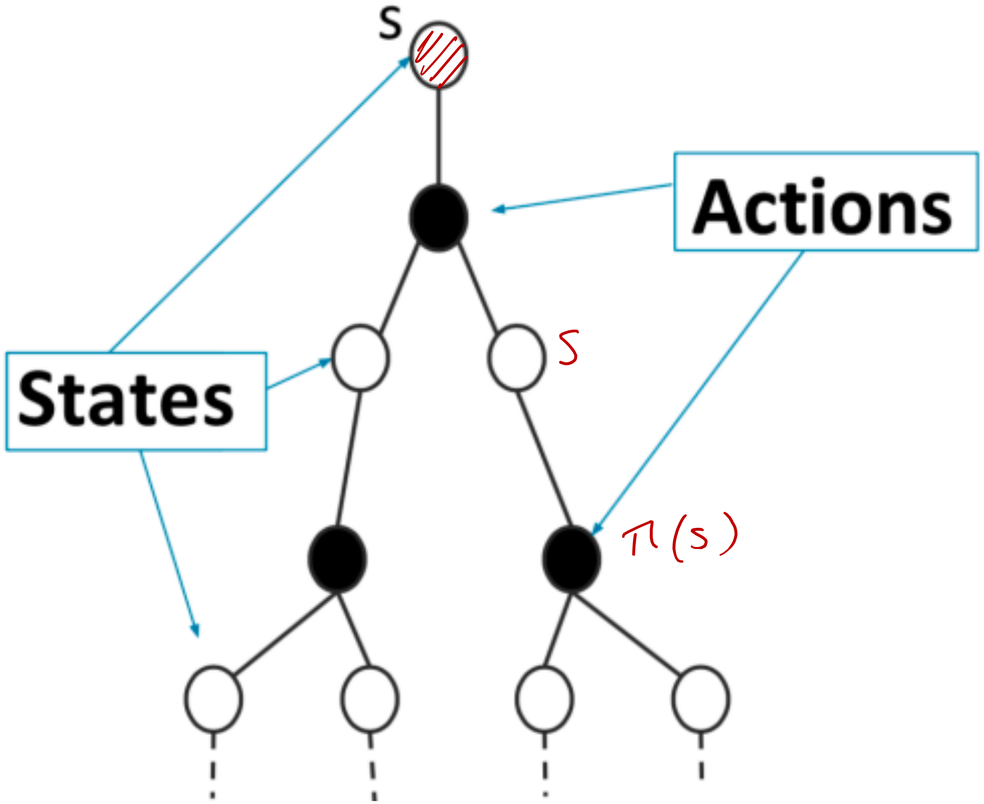
Policy Evaluation Diagram



Policy Evaluation Diagram

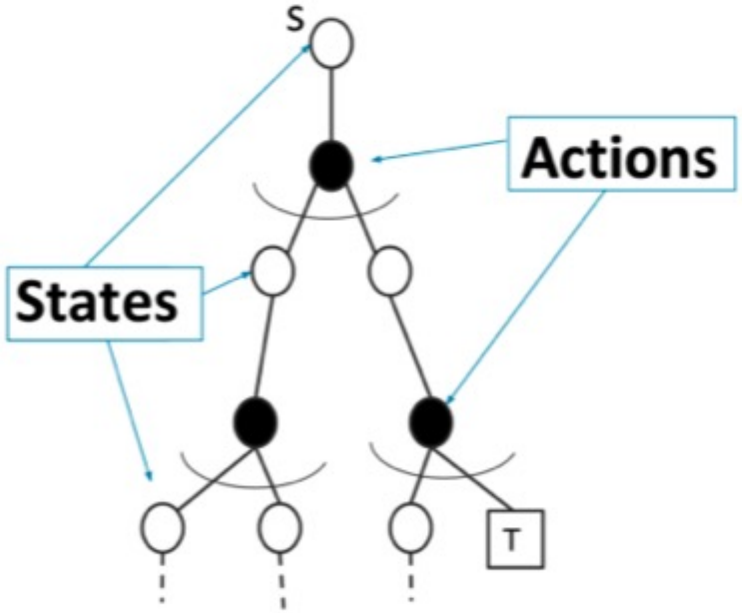


Policy Evaluation Diagram



Policy Evaluation Diagram

$$V^{\pi}(s) = V^{\pi}(s) + \alpha(G_{i,t} - V^{\pi}(s))$$



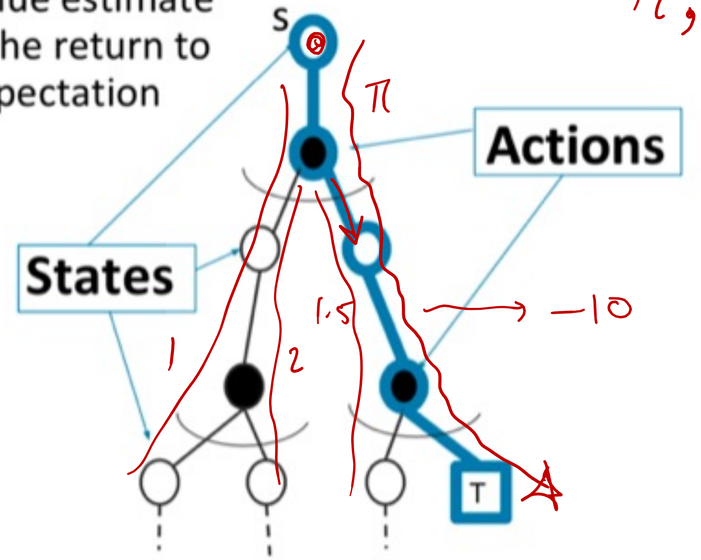
= Expectation

T = **Terminal state**

Policy Evaluation Diagram

$$V^{\pi}(s) = V^{\pi}(s) + \alpha(G_{i,t} - V^{\pi}(s))$$

MC updates the value estimate using a **sample** of the return to approximate an expectation



= Expectation

T = **Terminal state**

Val. Itr.

$$V^{\pi}_{k+1}(s) = \sum_{s'} P(s'|s,\pi(s)) \left[R(s,\pi(s),s') + \gamma V^{\pi}_{k}(s') \right]$$

$G_{i,t}$

$V^{\pi}(s) = -5$

$V^{\pi}(s) = 8$