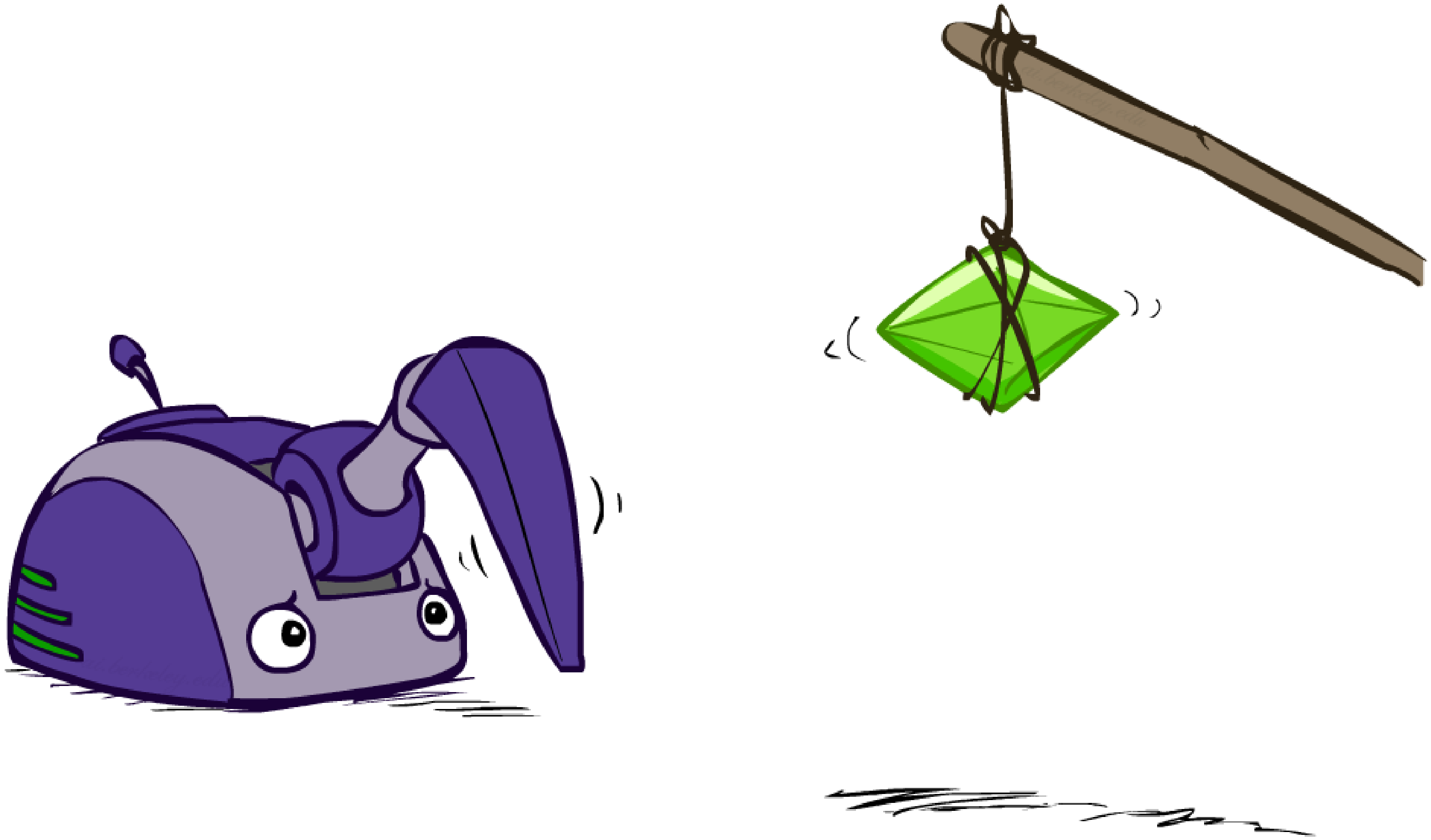
CS 594 Modern Reinforcement Learning

Lecture 2: Monte Carlo and Temporal Difference Methods

Reinforcement Learning



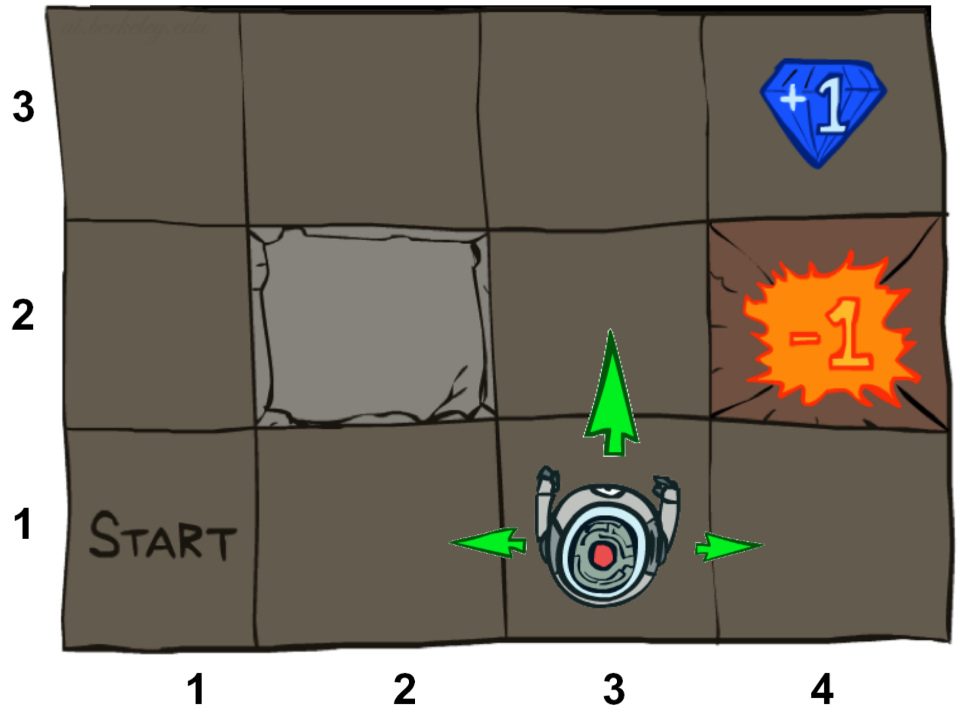
[Slides adapted from those created by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley. All materials available at http://ai.berkeley.edu.]

Markov Decision Processes

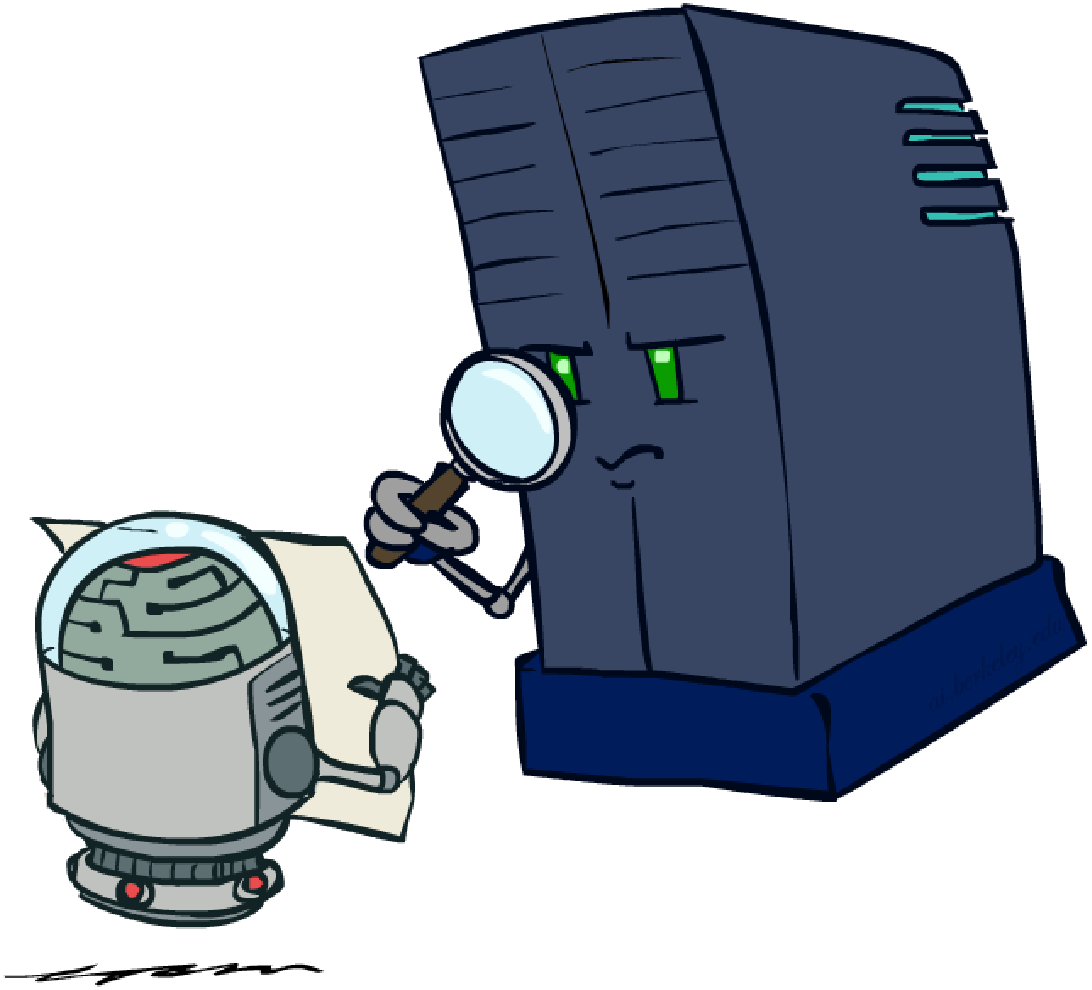
- Trajectory S_0, A_0, R_1, S_1, A_1, R_2, …

- A (finite) MDP is defined by:
  - A finite set of states $s \in S$
  - A finite set of actions $a \in A$
  - A finite set of rewards $r \in R$
  - Dynamics $p(s',r|s,a) = \Pr(S_t=s',R_t=r|S_{t-1}=s,A_{t-1}=a)$
  - Discount Rate γ
  - Possibly start state (distribution), terminal state

- New twist: don't have access to dynamics!

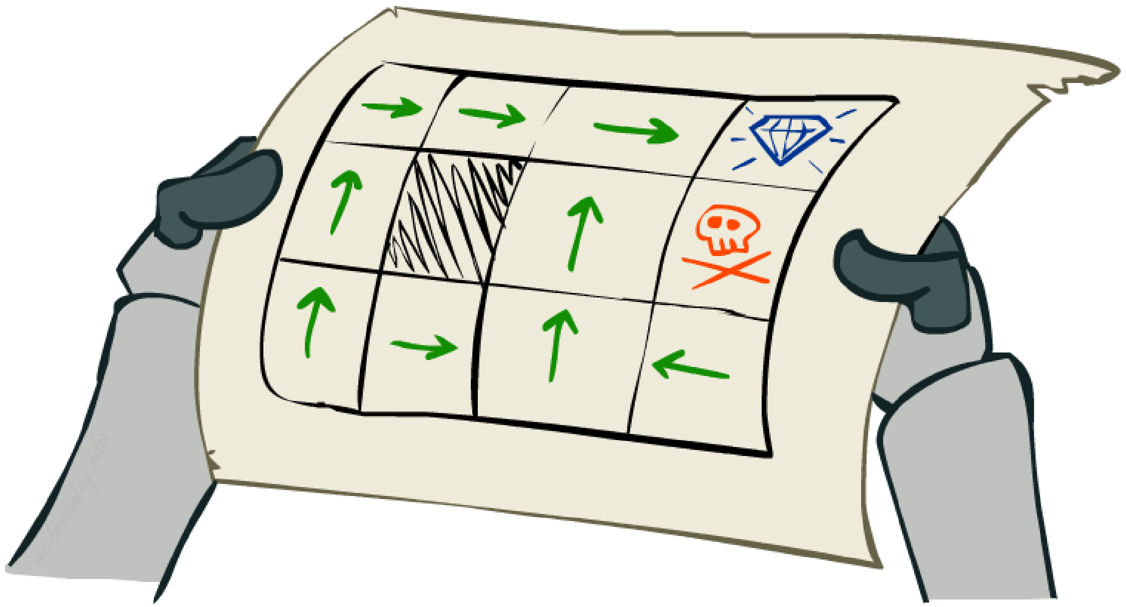


Policy Evaluation



Passive Reinforcement Learning

- Simplified task: policy evaluation
  - Input: a fixed policy $\pi(s)$
  - You don't know the dynamics
  - Goal: learn the state values



- In this case:
  - Learner is "along for the ride"
  - No choice about what actions to take
  - Just execute the policy and learn from experience
  - This is NOT offline planning! You actually take actions in the world.

Monte Carlo Estimates

- Estimate $v_\pi(s)$
- Can we do this without the dynamics or even the policy???
- Given realized trajectory s_0, a_0, r_1, s_1, a_1, r_2, s_2, …

- $v_\pi(s) = E_\pi[G_t \mid S_t = s]$
- $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+2} + \ldots = \sum_{k=0}^{T} \gamma^k R_{t+k+1}$
- $\sum_{k=0}^{T} \gamma^k r_{t+k+1}$
- Correct on average, but might have high variance

Improved Estimates By Averaging

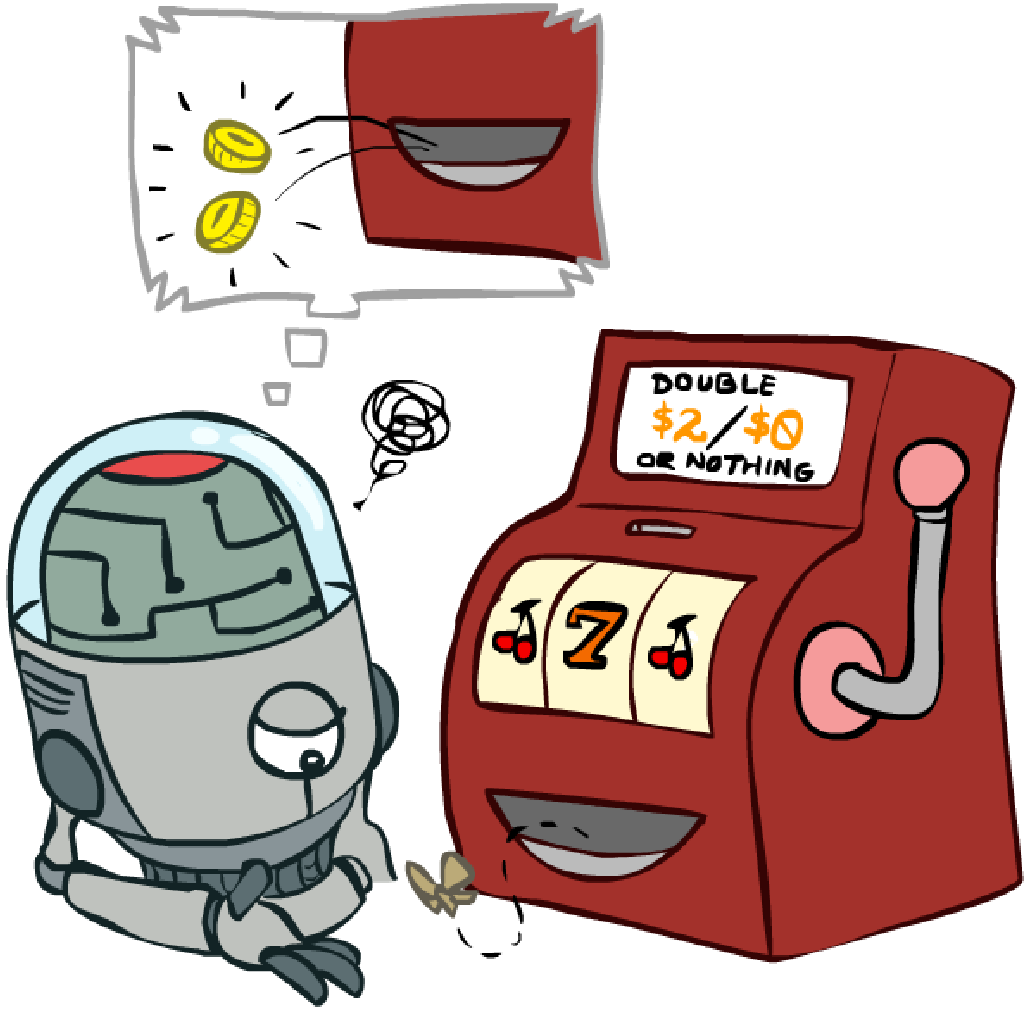
- Get samples $g_1(s),\ldots,g_n(s)$

- Estimate $v_\pi(s) \approx \frac{1}{n} \sum_i g_i(s)$

Incremental Average Computation

- $v_1(s) = g_1(s)$
- $v_2(s) = \frac{1}{2}(g_1(s) + g_2(s))$
- $v_3(s) = \frac{1}{3}(g_1(s) + g_2(s) + g_3(s))$
- $v_2(s) = \frac{1}{2}(v_1(s) + g_2(s))$
- $v_3(s) = \frac{1}{3}(2v_2(s) + g_3(s))$
- $v_k(s) = v_{k-1}(s) + \frac{1}{k}(g_k(s) - v_{k-1}(s))$

Temporal Difference Learning



TD(0)

- Monte Carlo Average Estimate:

$$v_k(s) = v_{k-1}(s) + \alpha_k\big(g_k(s) - v_{k-1}(s)\big)$$

- Policy Evaluation:

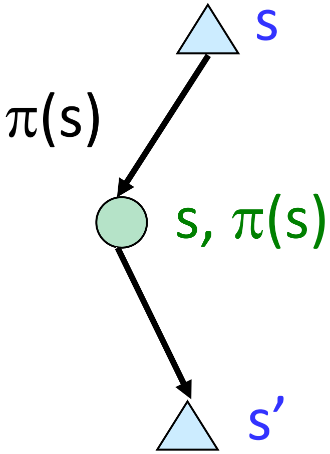
$$v_k(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)(r + \gamma v_{k-1}(s'))$$

- TD(0):

$$v_k(s) = v_{k-1}(s) + \alpha_k\big(r + \gamma v_{k-1}(s') - v_{k-1}(s)\big)$$

Temporal Difference Learning

- Big idea: learn from every experience!
  - Update V(s) each time we experience a transition (s, a, s', r)
  - Likely outcomes s' will contribute updates more often



- Temporal difference learning of values
  - Policy still fixed, still doing evaluation!
  - Move values toward value of whatever successor occurs: running average

Sample of V(s): $$sample = R(s, \pi(s), s') + \gamma V^{\pi}(s')$$

Update to V(s): $$V^{\pi}(s) \leftarrow (1 - \alpha)V^{\pi}(s) + (\alpha)sample$$

Same update: $$V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha(sample - V^{\pi}(s))$$

Exponential Moving Average

- Exponential moving average
  - The running interpolation update: $\bar{x}_n = (1-\alpha) \cdot \bar{x}_{n-1} + \alpha \cdot x_n$
  - Makes recent samples more important:

$$\bar{x}_n = \frac{x_n + (1-\alpha) \cdot x_{n-1} + (1-\alpha)^2 \cdot x_{n-2} + \dots}{1 + (1-\alpha) + (1-\alpha)^2 + \dots}$$

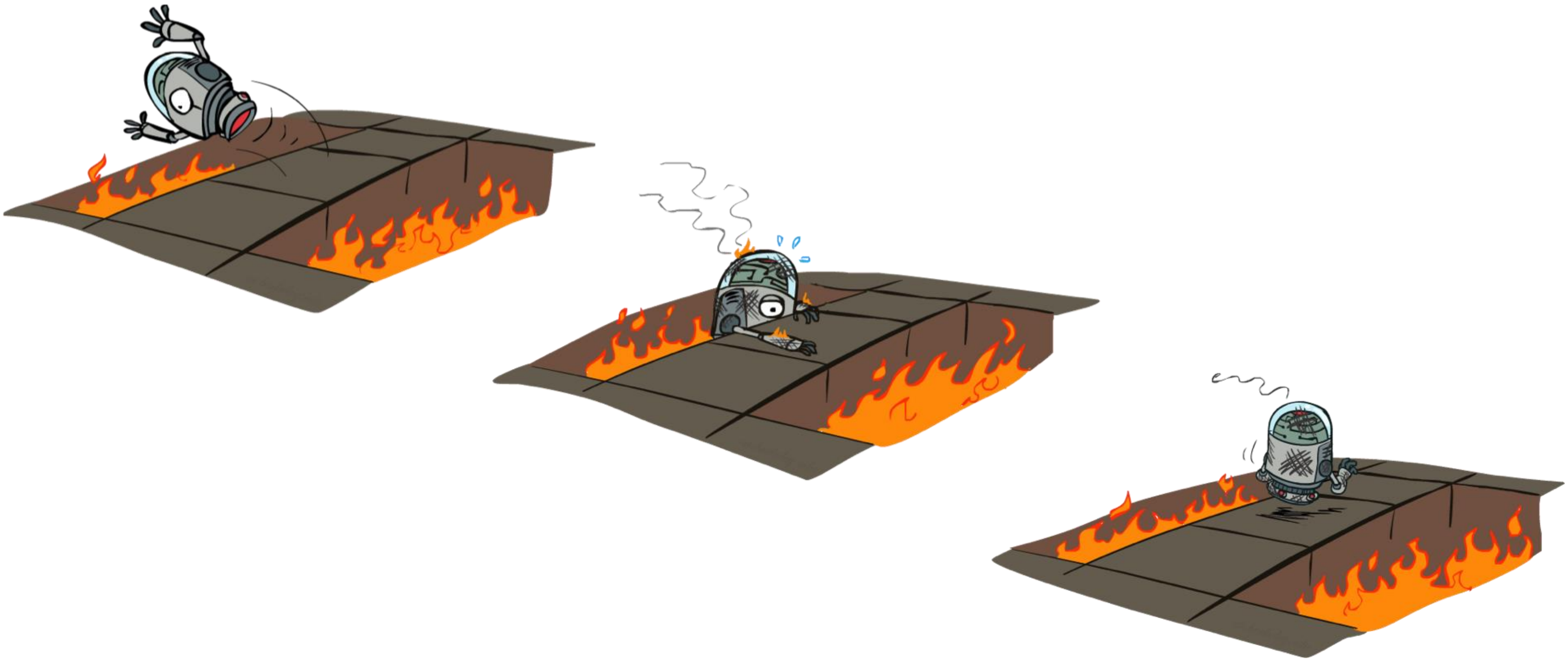
  - Forgets about the past (distant past values were wrong anyway)

- Decreasing learning rate (alpha) can give converging averages

Importance Weighting

- What if we have data generated by b but want to evaluate π?
- $v_\pi(s) = E_\pi[G_t \mid S_t = s]$
- $= \sum_{a_t, r_{t+1}, s_{t+1}, \ldots} \Pr_\pi[\, a_t, r_{t+1}, s_{t+1}, \ldots | S_t = s_t] \sum_{k=t}^{T} \gamma^k r_{k+1}$
- $= \sum_{a_t, r_{t+1}, s_{t+1}, \ldots} \prod_{k=t}^{T-1} \pi(a_k | s_k) p(s_{k+1}, r_{k+1} | s_k, a_k) \sum_{k=t}^{T} \gamma^k r_{k+1}$
- $= \sum_{a_t, r_{t+1}, s_{t+1}, \ldots} \prod_{k=t}^{T-1} \frac{\pi(a_k | s_k)}{b(a_k | s_k)}$
  $\prod_{k=t}^{T-1} b(a_k | s_k) p(s_{k+1}, r_{k+1} | s_k, a_k) \sum_{k=t}^{T} \gamma^k r_{k+1}$
- $= E_b\left[\left(\prod_{k=t}^{T-1} \frac{\pi(A_k | S_k)}{b(A_k | S_k)} \right) G_t \,\middle|\, S_t = s \right]$

On-policy TD Learning

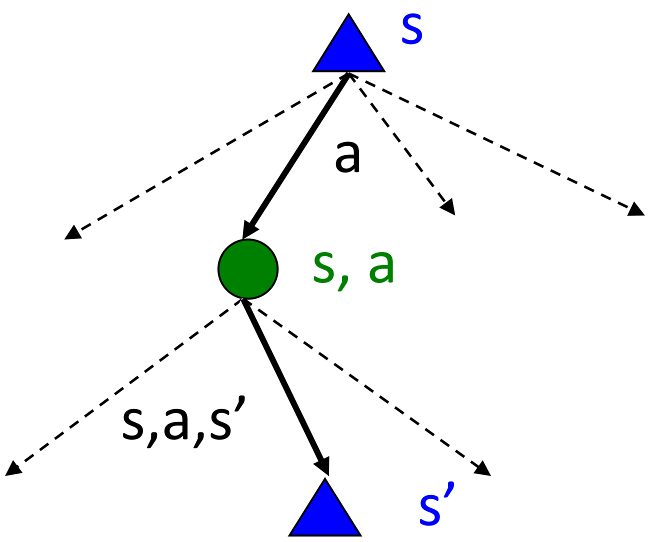


Problems with TD Value Learning

- TD value learning is a model-free way to do policy evaluation, mimicking Bellman updates with running sample averages
- However, if we want to turn values into a (new) policy, we're sunk:

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

$$Q(s,a) = \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V(s') \right]$$



- Idea: learn Q-values, not values
- Makes action selection model-free too!

Q TD(0)

- TD(0):

$$v_k(s) = v_{k-1}(s) + \alpha_k\big(r + \gamma v_{k-1}(s') - v_{k-1}(s)\big)$$

- Q-version of TD(0):

$$q_k(s,a) = q_{k-1}(s,a) + \alpha_k\big(r + \gamma q_{k-1}(s',a') - q_{k-1}(s,a)\big)$$

Sarsa

Initialize state s, action a

Do:

- Take action a
- Observe r, s’
- Choose a’ based on $\pi(q)$
- $q(s,a) = q(s,a) + \alpha\big(r + \gamma q(s',a') - q(s,a)\big)$
- s = s’, a=a’

Until episode ends (or forever)

Expected Sarsa

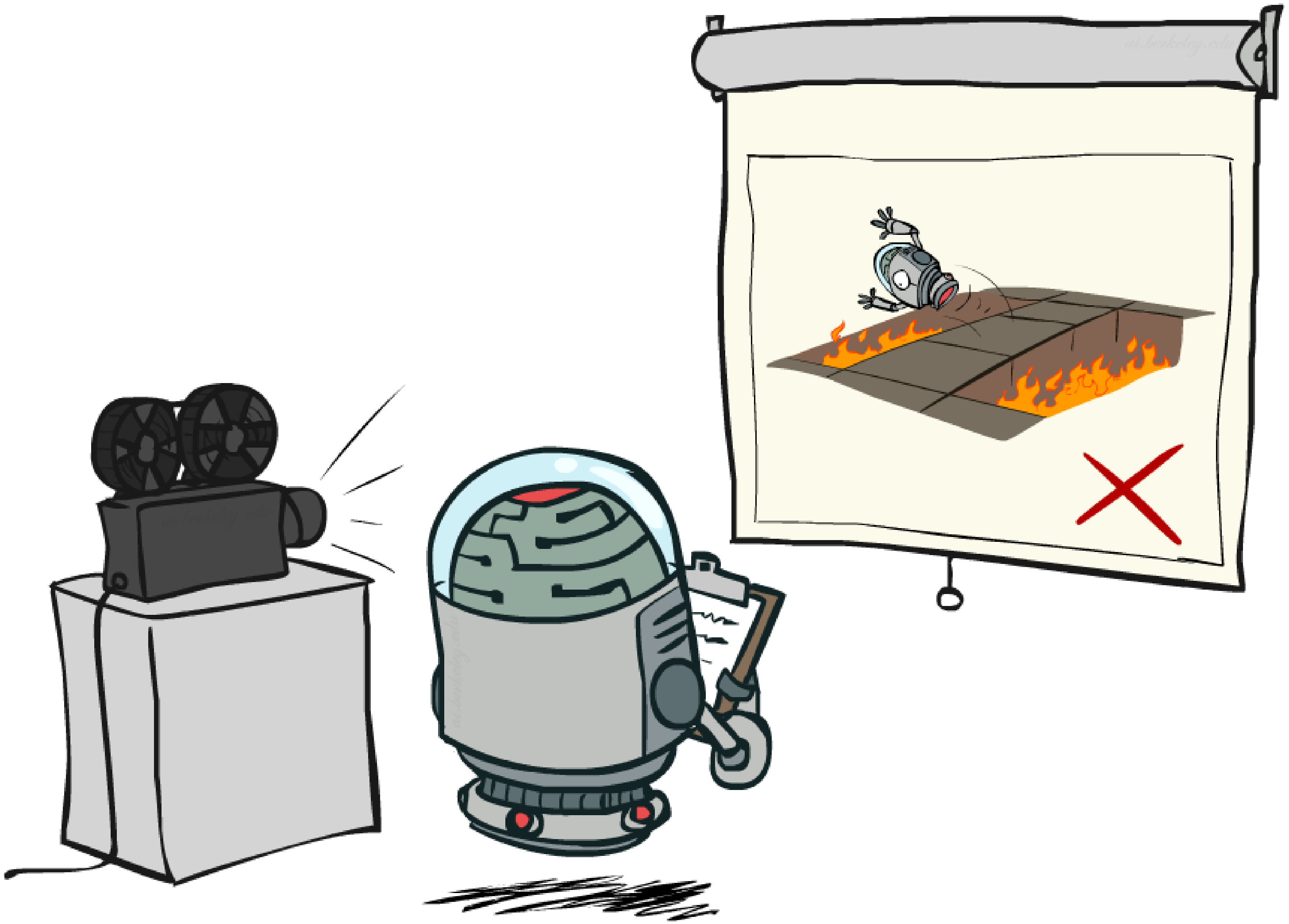
Initialize state s, action a

Do:

- Take action a
- Observe r, s’
- Choose a’ based on $\pi(q)$
- $q(s,a) = q(s,a) + \alpha\left(r + \gamma E_{a''\sim\pi(q)}[q(s',a'')] - q(s,a)\right)$
- s = s’, a=a’

Until episode ends (or forever)

Off-policy TD Learning



Expected Sarsa

Initialize state s, action a

Do:

- Take action a
- Observe r, s’
- Choose a’ based on $\pi(q)$
- $q(s,a) = q(s,a) + \alpha\left(r + \gamma E_{a'' \sim \pi(q)}[q(s',a'')] - q(s,a)\right)$
- s = s’, a=a’

Until episode ends (or forever)

Q-Learning

Initialize state s, action a

Do:

- Take action a
- Observe r, s’
- Choose a’ based on $\pi(q)$
- $q(s, a) = q(s, a) + \alpha\big(r + \gamma \max_{a''}[q(s', a'')] - q(s, a)\big)$
- s = s’, a=a’

Until episode ends (or forever)

Q-Learning

- TD Form:

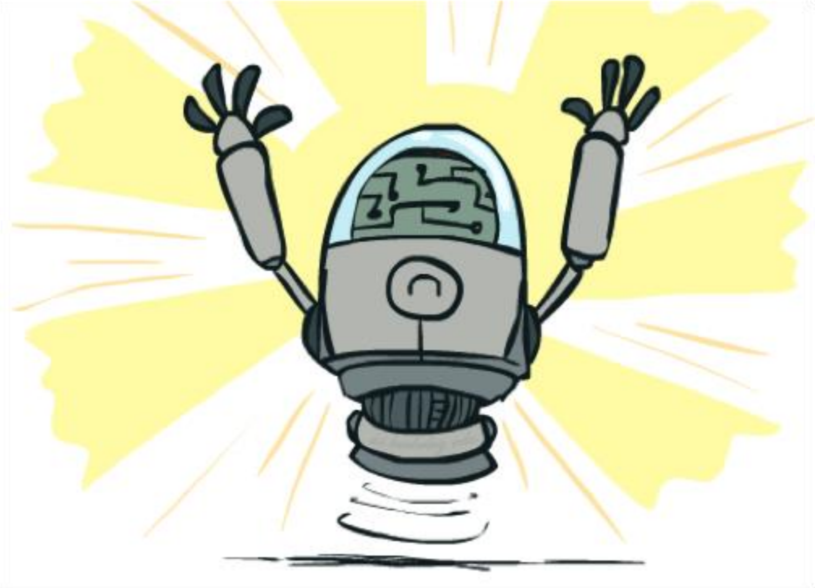
$$q(s,a) = q(s,a) + \alpha\left(r + \gamma \max_{\mathrm{a}''}[q(s',a'')] - q(s,a)\right)$$

- 411 Form:

$$q(s,a) = (1-\alpha)q(s,a) + \alpha\left(r + \gamma \max_{\mathrm{a}''}[q(s',a'')]\right)$$

Q-Learning Properties

- Amazing result: Q-learning converges to optimal policy -- even if you're acting suboptimally!
- This is called off-policy learning
- Caveats:
  - You have to explore enough
  - You have to eventually make the learning rate small enough
  - … but not decrease it too quickly
  - Basically, in the limit, it doesn't matter how you select actions (!)



Convergence of TD Methods

Theorem 1 *A random iterative process $\Delta_{n+1}(x) = (1-\alpha_n(x))\Delta_n(x) + \beta_n(x)F_n(x)$ converges to zero w.p.1 under the following assumptions:*

1) *The state space is finite.*
2) $\sum_n \alpha_n(x) = \infty$, $\sum_n \alpha_n^2(x) < \infty$, $\sum_n \beta_n(x) = \infty$, $\sum_n \beta_n^2(x) < \infty$, *and* $\mathrm{E}\{\beta_n(x)|P_n\} \leq \mathrm{E}\{\alpha_n(x)|P_n\}$ *uniformly w.p.1.*
3) $\| \mathrm{E}\{F_n(x)|P_n\} \|_W \leq \gamma \| \Delta_n \|_W$, *where* $\gamma \in (0,1)$.
4) $\mathrm{Var}\{F_n(x)|P_n\} \leq C(1 + \| \Delta_n \|_W)^2$, *where C is some constant.*

Here $P_n = \{\Delta_n, \Delta_{n-1}, \ldots, F_{n-1}, \ldots, \alpha_{n-1}, \ldots, \beta_{n-1}, \ldots\}$ *stands for the past at step n. $F_n(x)$, $\alpha_n(x)$ and $\beta_n(x)$ are allowed to depend on the past insofar as the above conditions remain valid. The notation $\|\cdot\|_W$ refers to some weighted maximum norm.*

Lemma 1. *Consider a stochastic process $(\alpha_t, \Delta_t, F_t)$, $t \geq 0$, where $\alpha_t, \Delta_t, F_t : X \to \Re$ satisfy the equations*

$$\Delta_{t+1}(x) = (1 - \alpha_t(x))\Delta_t(x) + \alpha_t(x)F_t(x), \quad x \in X, \quad t = 0, 1, 2, \ldots.$$

Let P_t be a sequence of increasing σ-fields such that α_0 and Δ_0 are P_0-measurable and α_t, Δ_t and F_{t-1} are P_t-measurable, $t = 1, 2, \ldots$. Assume that the following hold:

1. *the set X is finite.*
2. $0 \leq \alpha_t(x) \leq 1$, $\sum_t \alpha_t(x) = \infty$, $\sum_t \alpha_t^2(x) < \infty$ *w.p.*1.
3. $\|E\{F_t(\cdot)|P_t\}\|_W \leq \kappa\|\Delta_t\|_W + c_t$, *where $\kappa \in [0, 1)$ and c_t converges to zero w.p.*1.[2]
4. $\mathrm{Var}\{F_t(x)|P_t\} \leq K(1 + \|\Delta_t\|_W)^2$, *where K is some constant.*

*Then, Δ_t converges to zero with probability one (w.p.*1).

Robbins-Monro Conditions

- For all pairs (s,a):
- $\sum_t \alpha_t(s,a) = \infty$
- $\sum_t \left(\alpha_t(s,a)\right)^2 < \infty$

- They go to zero but not too fast
- Implicit: Each (s,a) is visited an infinite number of times
  - Need to explore
  - E.g. ϵ-greedy

Summary: Monte Carlo and TD Methods

- Monte Carlo
  - A sampled trajectory is an unbiased estimate of the return
  - Reduce noise by averaging multiple samples
  - Use importance weighting to evaluate a different policy
- TD Methods
  - You don’t have to use the entire trajectory to do Monte Carlo updates
  - You can even adjust the policy while learning
  - Robbins-Monro conditions for convergence