RL: Model-Free RL

Forest Agostinelli
University of South Carolina

Topics Covered in This Class

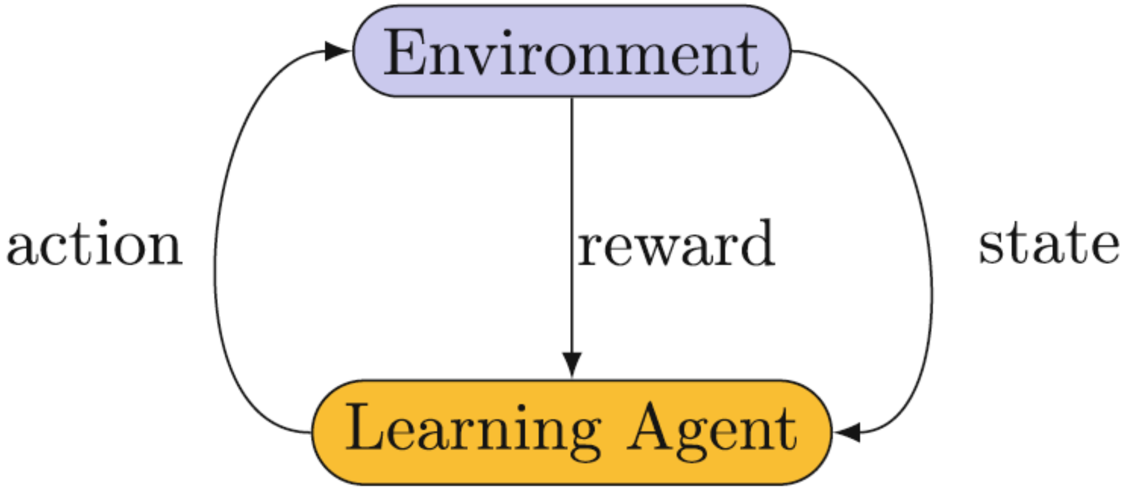
- **Part 1: Search**
  - Pathfinding
    - Uninformed search
    - Informed search
  - Adversarial search
  - Optimization
    - Local search
    - Constraint satisfaction
- **Part 2: Knowledge Representation and Reasoning**
  - Propositional logic
  - First-order logic
  - Prolog
- **Part 3: Knowledge Representation and Reasoning Under Uncertainty**
  - Probability
  - Bayesian networks
- **Part 4: Machine Learning**
  - Supervised learning
    - Inductive logic programming
    - Linear models
    - Deep neural networks
    - PyTorch
  - Reinforcement learning
    - Markov decision processes
    - Dynamic programming
    - Model-free RL
  - Unsupervised learning
    - Clustering
    - Autoencoders

Outline

- Review
- Model-free prediction
  - Monte Carlo prediction
  - Temporal difference prediction
- Model-free control
  - Monte Carlo control
  - Temporal difference control

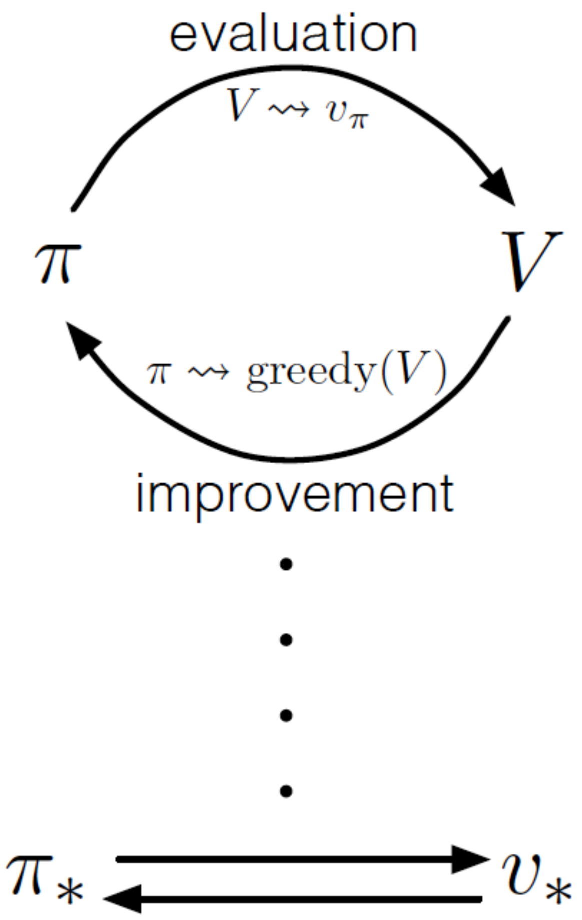
Reinforcement Learning

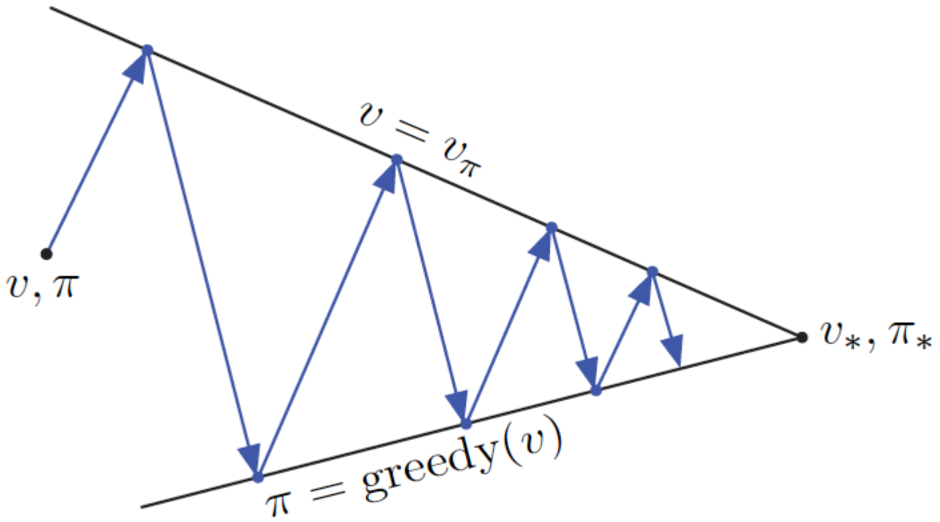
- **Reinforcement learning**: learning to map states to actions so that we maximize the expected future reward we receive from the environment.
- This mapping of states to actions is called a policy function.
  - Deterministic: $a = \pi(s)$
  - Stochastic: $\pi(a|s) = P(A = a|S = s)$
- At each time step t
  - In state S_t, agent takes action A_t
  - Based on state s_t and action a_t, the environment transitions to state S_{t+1} and outputs reward R_{t+1}



Generalized Policy Iteration

- **Policy Evaluation:** Estimate the expected future reward when following policy π
- **Policy Improvement:** Improve policy π so that it obtains a greater expected future reward
- We can obtain an optimal policy by iterating between policy evaluation and policy improvement





Dynamic Programming Summary

- **Policy Evaluation**: Uses Bellman equation as an update rule

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

- **Policy Improvement**: Behave greedily with respect to value function

$$\pi'(s) = \operatorname*{argmax}_a(r(s,a) + \gamma \sum_{s'} p(s'|s,a)\, V(s'))$$

- **Policy Iteration**: Iterate between policy evaluation and policy improvement until convergence
- **Value Iteration**: Uses Bellman optimality equation as an update rule

$$V(s) = \max_a(r(s,a) + \gamma \sum_{s'} p(s'|s,a)\, V(s'))$$

Crucial Assumption

- Assuming environment dynamics are known
  - $P(S_{t+1} = s', R_{t+1} = r | S_t = s, A_t = a)$
- Environment dynamics are unknown in many real-world scenarios
  - Self-driving cars
  - Space exploration
- Even if known, may be too costly to compute (i.e. physics)

Crucial Assumption

$$V(s) = \max_a \left(r(s,a) + \gamma \sum_{s'} p(s'|s,a) V(s') \right)$$

Unknown

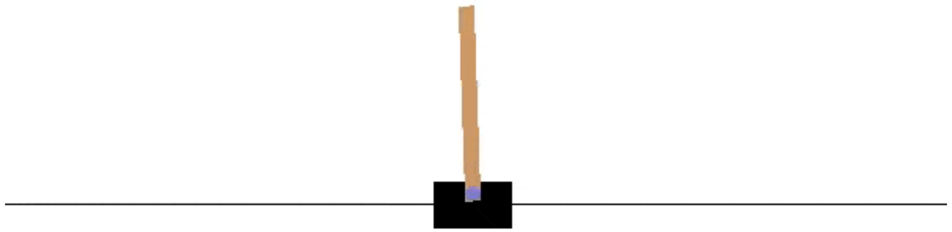
Unknown

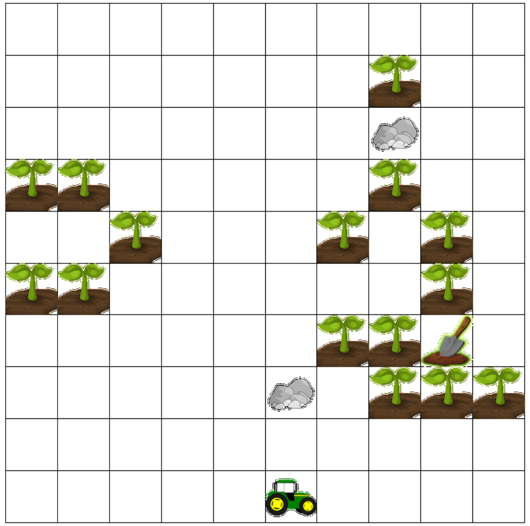
Model-Free Reinforcement Learning

- Instead of using a model, learn from experience
- We know $\mathcal{A}$
  - We know what actions we can take
  - We do *not* know $p(s', r|s, a)$.
- We may not know $\mathcal{S}$
  - That is, we may not be able to simply enumerate every possible state

Model-Free RL: Examples

- Self-driving cars
- Disaster cleanup
- Conversational agent





Model-Free RL: Prediction vs Control

- **Prediction (policy evaluation)**
  - Previously, when we wanted to know the v_π, we used the Bellman equation as an update equation
  - We proved that we have found v_π when we reach a fixed point
  - However, this requires a model
  - Nonetheless, we can evaluate v_π (predict the expected reward)
  - However, we cannot know for sure if we have conveged to v_π
- **Control**
  - We now want to learn how to act (control)
  - We cannot know for sure if we have converged to v_*
  - The concepts of policy iteration are used: iterate between policy evaluation and improvement
  - For now, we will need q_π

Model-Free RL: Exploration vs Exploitation

- **Exploration**: Learn more about the environment
- **Exploitation**: Use what you have learned to obtain more reward

Model-Free RL: On-Policy vs Off-Policy

- Your policy determines your experience
- We need to explore using randomness
  - May not be the best policy
- Experience may be delicate and hard to obtain (i.e. a hospital)
- **Behavior policy**: policy that we use to interact with the environment
- **Target policy**: policy that we wish to evaluate and/or improve

Monte Carlo Prediction (Policy Evaluation)

- We want to know the value of some policy π

$$v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$$

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+1+k}$$

- Can approximate via a running average
  - $N(S_t) = N(S_t) + 1$ Number of times state has been seen
  - $S(S_t) = S(S_t) + G_t$ - Sum of all returns from S_t

$$V(S_t) = S(S_t)/N(S_t)$$

  - Will converge to $v_\pi(S_t)$ as $N(S_t) \to \infty$
  - First visit or every visit

Monte Carlo Policy Prediction

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 $Returns(S_t)$

      $V(S_t) \leftarrow \text{average}(Returns(S_t))$

- Can do every state, though this adds bias due to correlation
- Have to wait for the end of the episode to update estimates

Monte Carlo Prediction: Incremental Update

- Or, can do an incremental update
  - $V(S_t) = V(S_t) + \alpha(G_t - V(S_t))$
  - α is the learning rate
- $V_n(s) = \frac{1}{n(s)} \sum_{k=1}^{n(s)} G^k$
- $= \frac{1}{n(s)} (G^{n(s)} + \sum_{k=1}^{n(s)-1} G^k)$
- $= \frac{1}{n(s)} \left(G^{n(s)} + (n(s) - 1)V_{n-1}(s)\right)$
- $= \frac{G^{n(s)}}{n(s)} + V_{n-1}(s) - \frac{V_{n-1}(s)}{n(s)}$
- $= V_{n-1}(s) + \frac{1}{n(s)} (G^{n(s)} - V_{n-1}(s))$
- $= V_{n-1}(s) + \alpha_n (G^{n(s)} - V_{n-1}(s))$
- Shown to converge to v_π if Robbins-Monro conditions are met
  - $\sum_{n=0}^{\infty} \alpha_n = \infty$
  - $\sum_{n=0}^{\infty} \alpha_n^2 < \infty$

Temporal Difference Prediction

- For Monte-Carlo methods, we have to wait until the end of the episode before we can learn
  - We cannot learn from positive or negative experiences before our episode has ended
  - Does not work for infinite horizon problems
- Temporal differences methods can learn after every step through bootstrapping
  - This exploits the Markov property

Temporal Difference Prediction

- Monte-Carlo
  - $v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$
  - $V(S_t) = V(S_t) + \alpha(G_t - V(S_t))$
- Temporal Differences
  - $v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s]$
  - $v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s]$
  - $V(S_t) = V(S_t\ ) + \alpha(\mathrm{R}_{t+1} + \gamma V(S_{t+1}) - V(S_t))$
  - Note that $V(S_{t+1})$ is **not** an unbiased estimate of $v_\pi(S_{t+1})$
- Shown to converge to v_π if Robbins-Monro conditions are met
  - $\sum_{n=0}^{\infty} \alpha_n = \infty$
  - $\sum_{n=0}^{\infty} \alpha_n^2 < \infty$

TD(0) Prediction

Tabular TD(0) for estimating v_π

Input: the policy π to be evaluated
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop for each episode:
    Initialize S
    Loop for each step of episode:
        $A \leftarrow$ action given by π for S
        Take action A, observe R, S'
        $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
        $S \leftarrow S'$
    until S is terminal

Monte-Carlo and TD(0) Policy Evaluation

Monte -Carlo

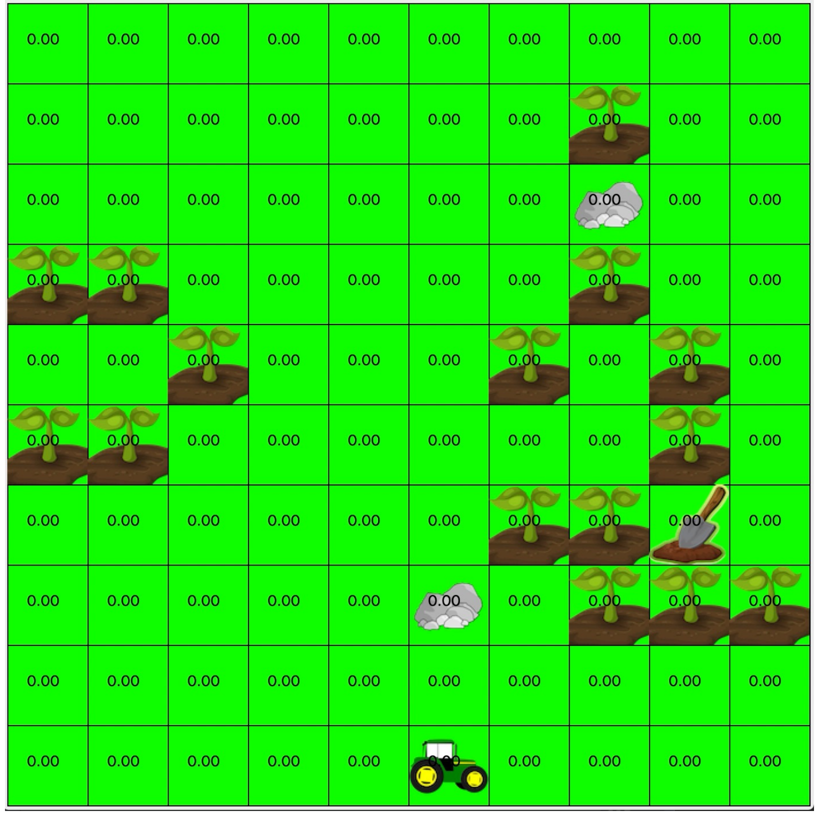
TD(0)

MC vs TD(0): Bias/Variance

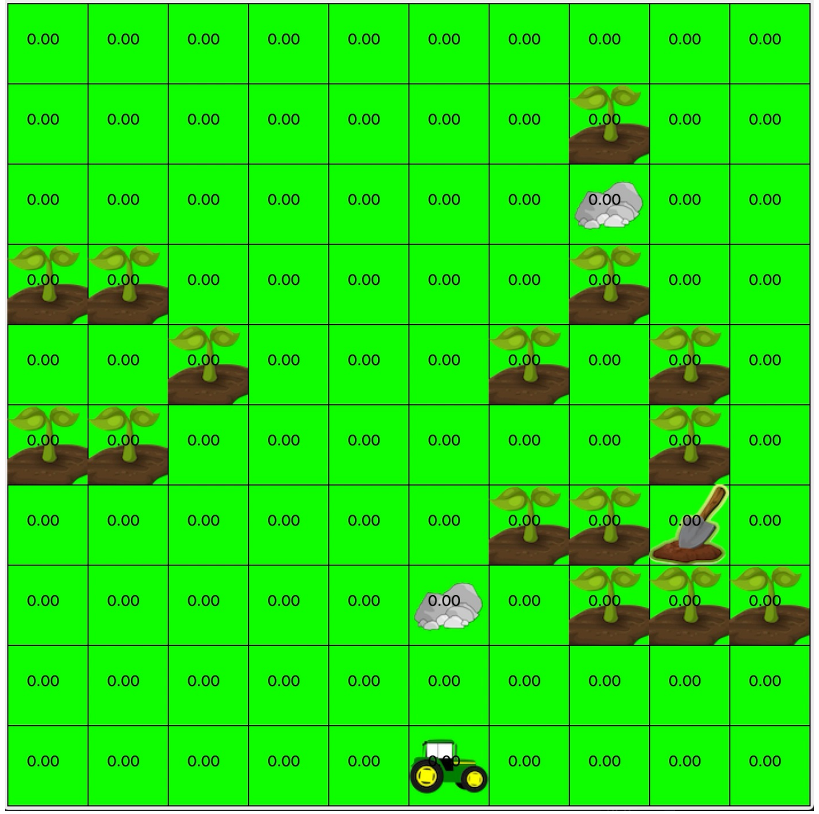
- Monte Carlo
  - Samples $\mathbb{E}_\pi[G_t | S_t = s]$
  - Randomness introduced at every sample from random policies, transitions, and rewards.
- TD(0)
  - Estimates $\mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s]$ with $V(S_{t+1})$
  - $V(S_{t+1})$ is, most likely, initially incorrect
  - Only samples one step
- **Monte Carlo is unbiased but more variance**
- **TD(0) is biased but less variance**

MC vs TD(0): Step-by-Step

- Evaluate uniform random policy on AI Farm
- Learning rate = 0.01



Monte-Carlo



TD(0)

MC vs TD(0): Over 40,000 iterations

- Evaluate uniform random policy on AI Farm
- Learning rate = 0.01
  - Not optimal, needs to be tuned
- Videos show estimates after every 1000 episodes.

-2274.06	-2246.84	-2197.41	-2135.24	-2069.99	-2009.64	-1959.49	-1918.20	-1857.38	-1820.67
-2297.28	-2265.07	-2206.14	-2134.31	-2061.11	-1995.42	-1946.64	-1884.72	-1829.27	-1779.97
-2348.70	-2306.01	-2223.76	-2130.77	-2040.71	-1960.30	-1893.92	-1831.79	-1742.02	-1685.95
-2389.81	-2333.52	-2248.12	-2120.29	-2006.68	-1907.14	-1823.96	-1704.48	-1608.05	-1531.88
-2385.21	-2337.15	-2212.90	-2091.58	-1954.59	-1833.61	-1688.29	-1541.14	-1351.83	-1297.63
-2326.68	-2265.96	-2170.76	-2025.55	-1882.47	-1731.44	-1550.44	-1268.95	-907.49	-956.19
-2226.86	-2176.25	-2076.61	-1953.40	-1814.29	-1655.26	-1411.06	-974.74	0.00	-610.45
-2124.66	-2082.58	-2002.04	-1893.14	-1762.04	-1602.23	-1410.82	-1116.95	-768.32	-822.15
-2060.53	-2023.37	-1951.82	-1851.10	-1725.48	-1576.81	-1402.05	-1211.91	-1032.18	-985.69
-2029.55	-1994.57	-1926.78	-1829.95	-1707.97	-1564.48	-1404.66	-1243.45	-1109.79	-1049.74

True v_π

0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

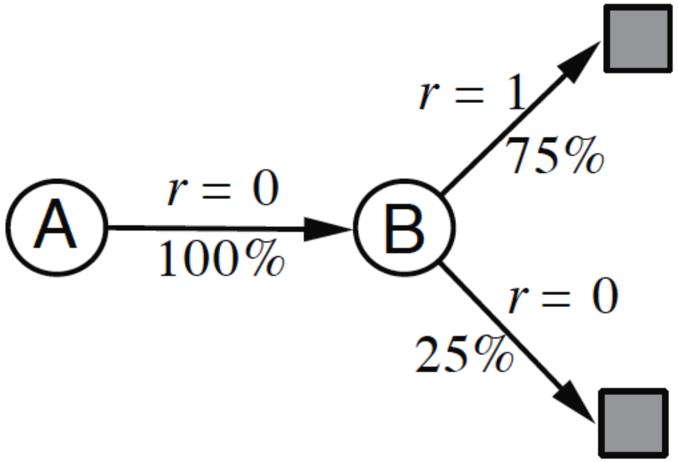
Monte-Carlo

0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

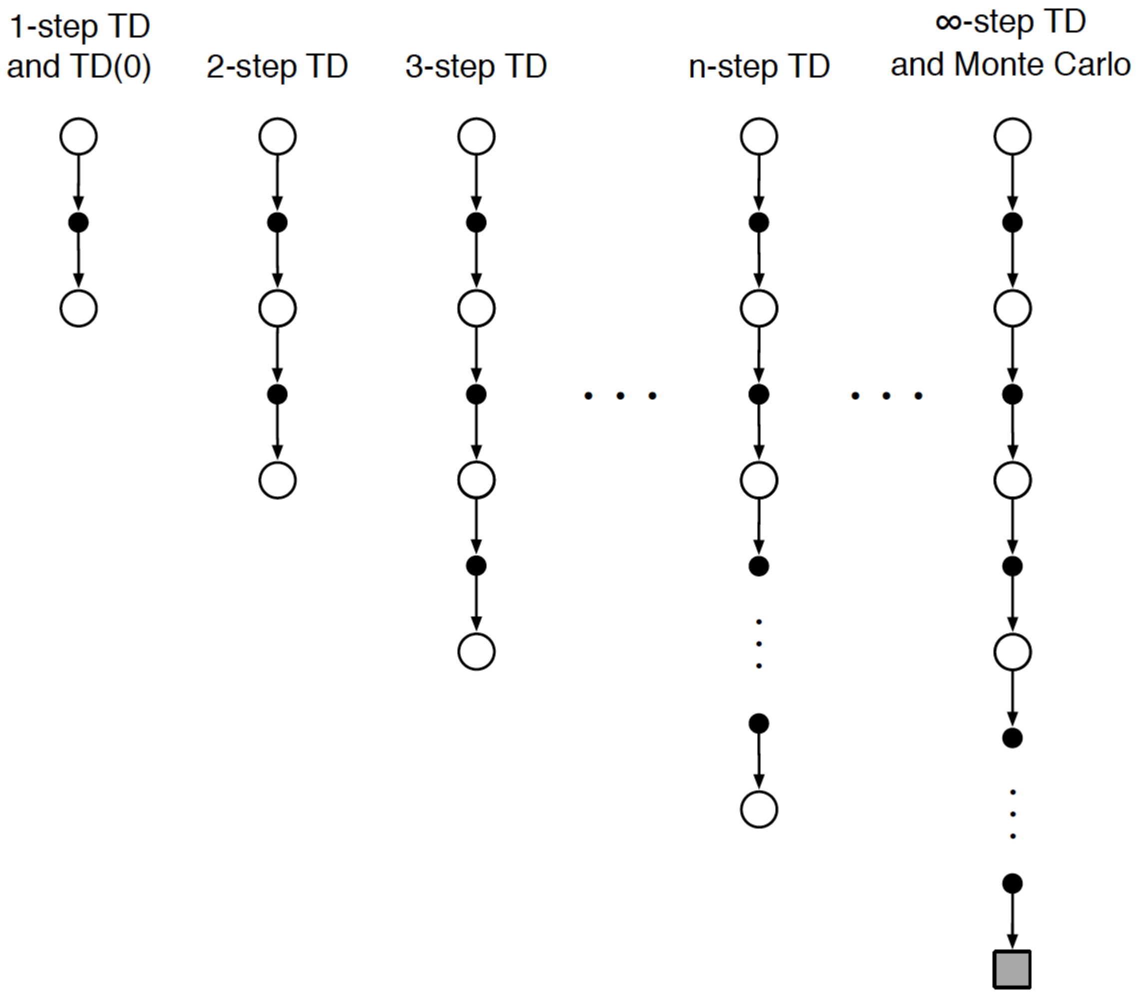
TD(0)

Monte Carlo vs TD

- Consider this MDP
- Say you experience the following eight episodes:
  - A(0), B(0)
  - B(1) <- 6 episodes
  - B(0) <- 1 episode
- If you repeatedly loop over these episodes while doing Monte Carlo or TD, what are the predicted values for A and B
- V(B)
  - MC: ¾
  - TD: ¾
- V(A)
  - MC: 0
  - TD: ¾
- Temporal difference finds the correct value for V(A) because it bootstraps from its estimate of V(B)



n-step Temporal Difference Prediction

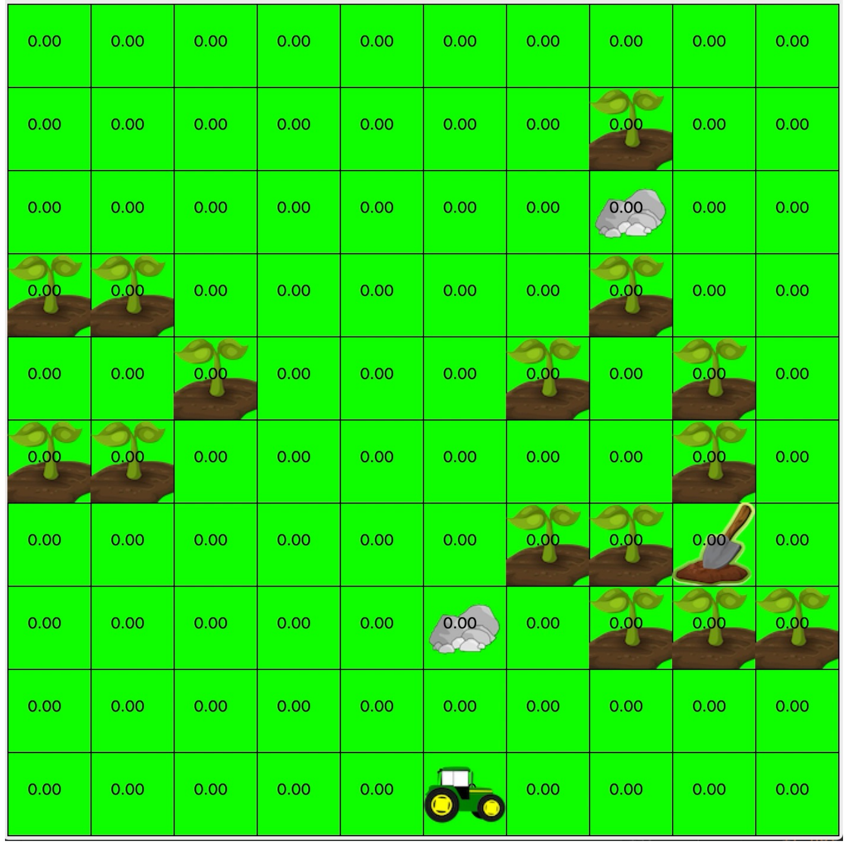


n-step Temporal Difference Prediction

- Monte Carlo Prediction
  - $v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$
  - $V(S_t) = V(S_t) + \alpha(G_t - V(S_t))$
- Temporal Difference Prediction
  - $v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s]$
  - $V(S_t) = V(S_t) + \alpha(\mathrm{R}_{t+1} + \gamma V(S_{t+1}) - V(S_t))$
- n-step Temporal Difference Prediction
  - $v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 v_\pi(S_{t+2}) | S_t = s]$
  - $v_\pi(s) = \mathbb{E}_\pi[\mathrm{R}_{\mathrm{t+1}} + \gamma \mathrm{R}_{\mathrm{t+2}} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n}) | S_t = s]$
  - $G_{t:t+n} = \mathrm{R}_{\mathrm{t+1}} + \gamma \mathrm{R}_{\mathrm{t+2}} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})$
  - $V(S_t) = V(S_t) + \alpha(G_{t:t+n} - V(S_t))$

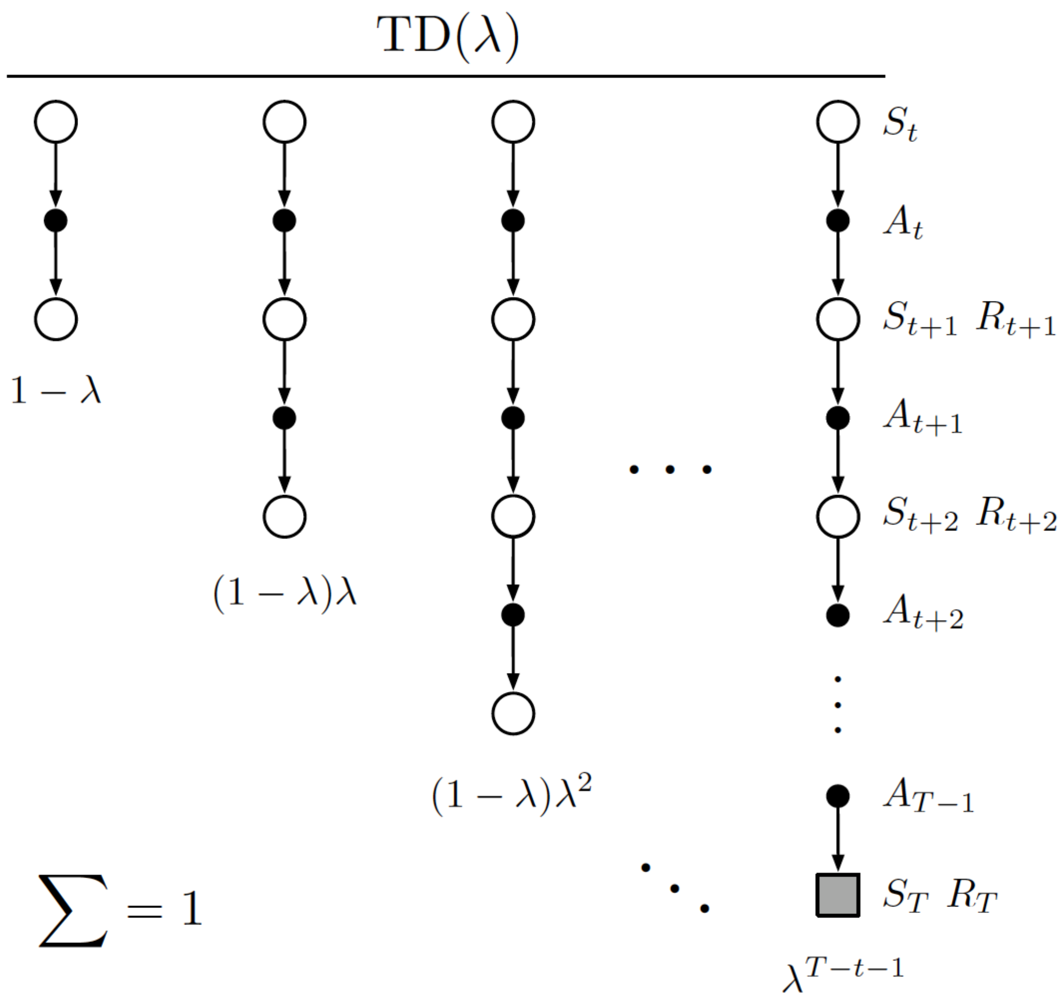
n-step Temporal Difference Prediction

- n=5
- Must be tuned to the problem at hand

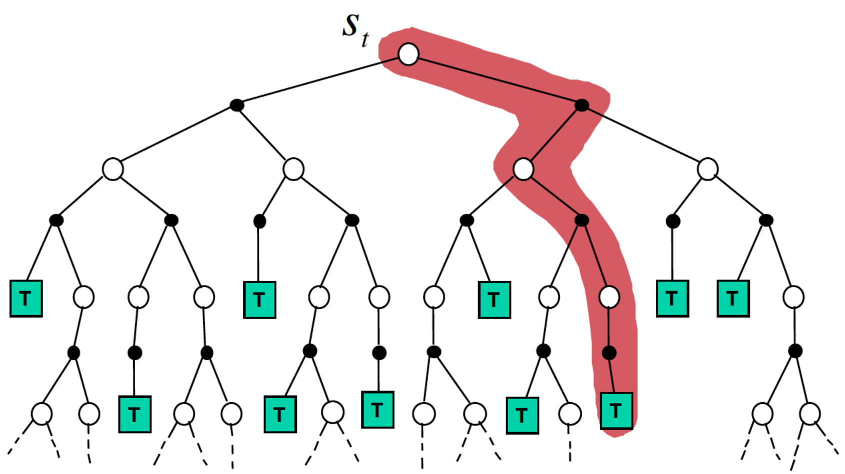


TD(λ)

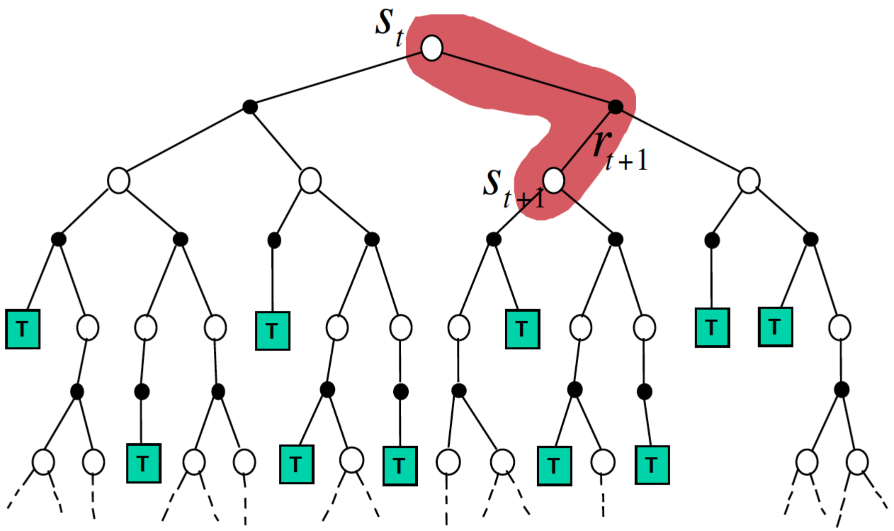
- Average over n-step returns
- Have to wait n-steps before updating state for n-step TD
- Use eligibility traces to update states without having to wait n-steps
  - Achieves approximately the same update
- TD(0): $\lambda = 0$
- Monte Carlo $\lambda = 1$



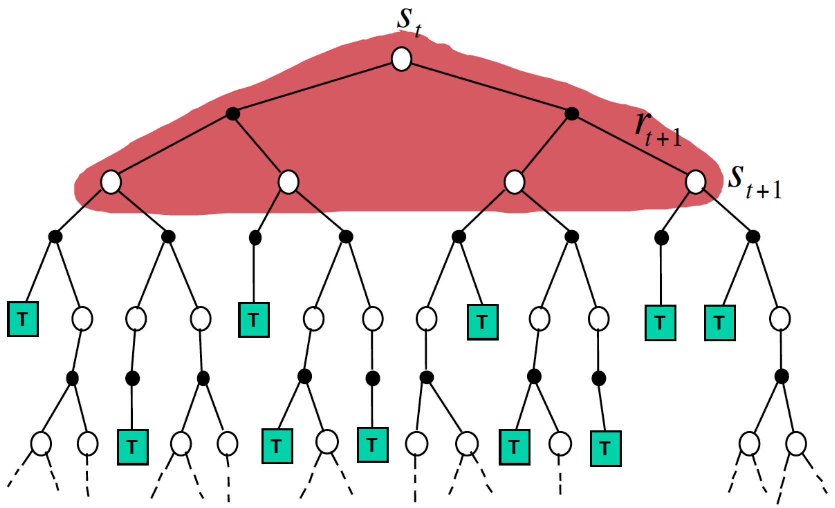
Visualization of Backups



Monte-Carlo



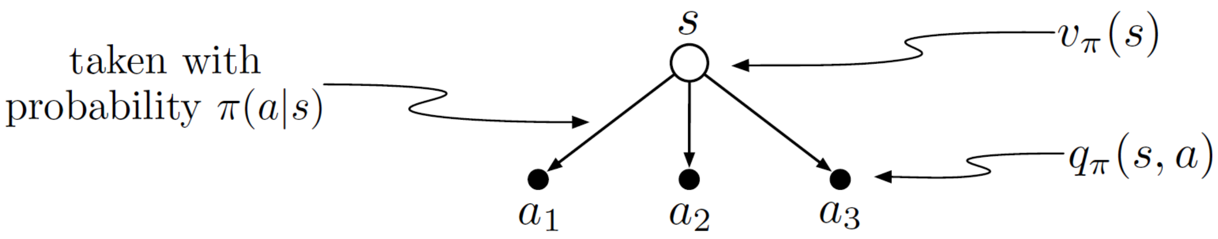
TD(0)

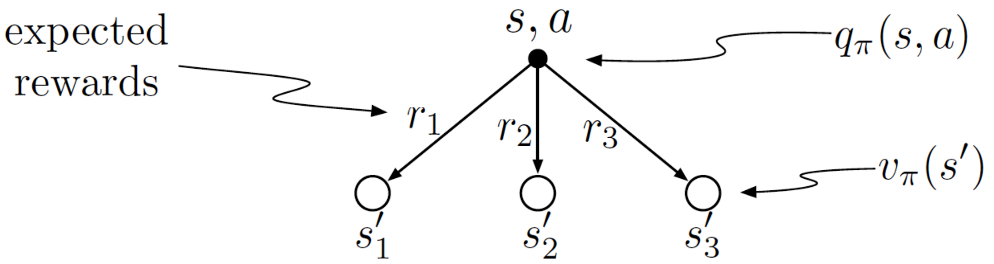


Dynamic Programming

Model-Free Control

- In this dynamic programming, we induced a policy by doing a one step lookahead using the value function
  - $\pi(s) = \underset{a}{\operatorname{argmax}}(r(s,a) + \gamma \sum_{s'} p(s'|s,a) V(s'))$
- However, we cannot do this in the model-free case because we do not have access to a model
- Therefore, we use an action-value function to induce a policy
  - $q_\pi(s,a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a] = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+1+k} \, | S_t = s, A_t = a]$
  - $q_\pi(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) \, v_\pi(s')$
  - $\pi(s) = \underset{a}{\operatorname{argmax}}(Q(s,a))$



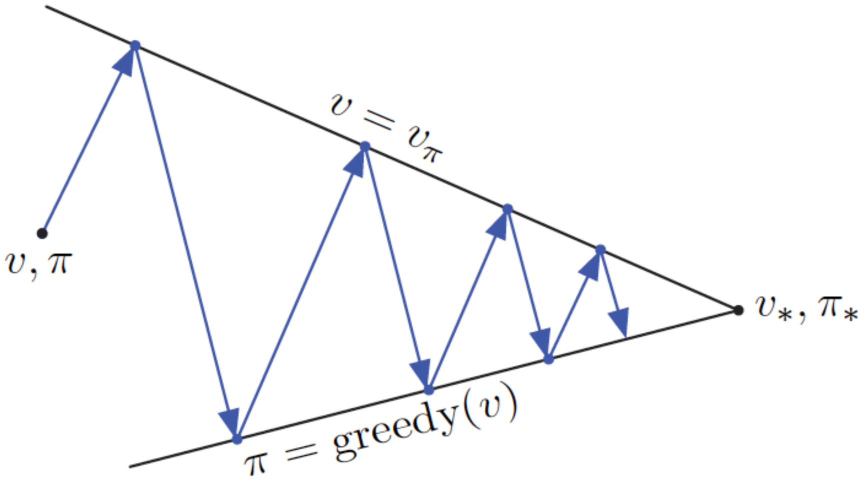


Model-Free Control: Exploration

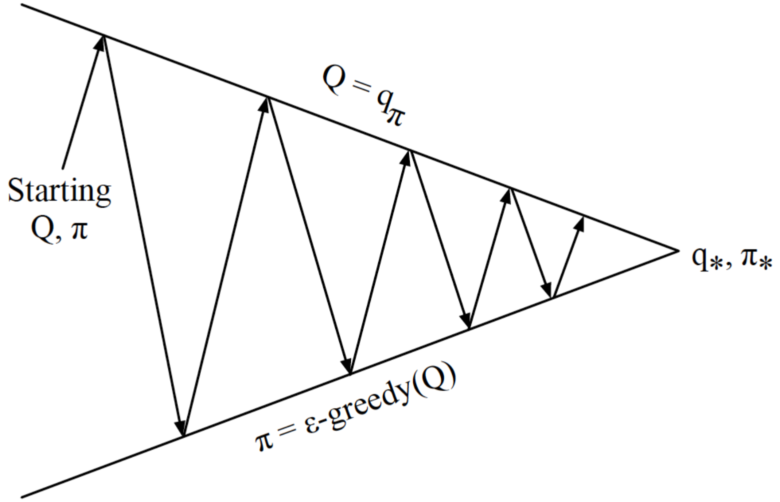
- How do we ensure that we explore our state space?
  - In dynamic programming, we assumed that we could just loop over every possible state
  - Cannot do this in the model-free case
- ϵ-greedy policy
  - Take a random action with probability ϵ
  - Take the greedy action, $\underset{a}{\operatorname{argmax}}\big(Q(s,a)\big)$, with probability 1- ϵ
- While there are many more sophisticated exploration methods, ϵ-greedy exploration can work well on some problems

Model-Free Control

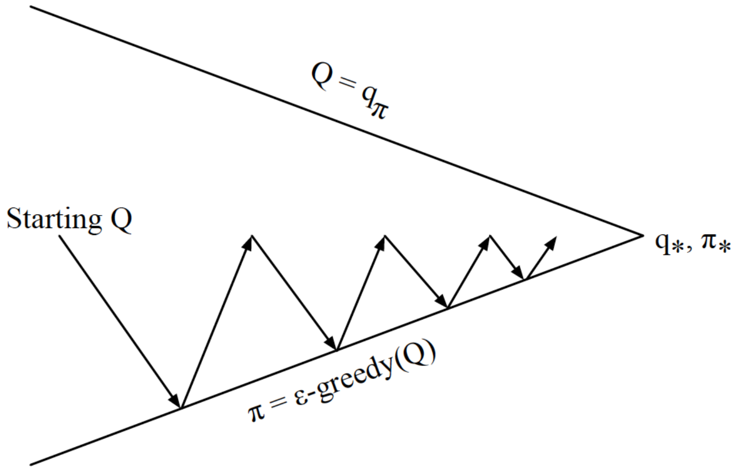
- **Policy Evaluation**: Learn an action-value function.
- **Policy Improvement**: Act epsilon greedily with respect to it.



Dynamic programming



Model-free with infinite time to estimate q_π



Model-free with finite time to estimate q_π

Model-Free Control

Policy improvement theorem: $v_{\pi'}(s) \geq q_\pi(s, \pi'(s)) \geq v_\pi(s)$ for all $s \in \mathcal{S}$

Theorem

For any ϵ-greedy policy π, the ϵ-greedy policy π' with respect to q_π is an improvement, $v_{\pi'}(s) \geq v_\pi(s)$

$$
\begin{aligned}
q_\pi(s, \pi'(s)) &= \sum_{a \in \mathcal{A}} \pi'(a|s) q_\pi(s, a) \\
&= \epsilon/m \sum_{a \in \mathcal{A}} q_\pi(s, a) + (1 - \epsilon) \max_{a \in \mathcal{A}} q_\pi(s, a) \\
&\geq \epsilon/m \sum_{a \in \mathcal{A}} q_\pi(s, a) + (1 - \epsilon) \sum_{a \in \mathcal{A}} \frac{\pi(a|s) - \epsilon/m}{1 - \epsilon} q_\pi(s, a) \\
&= \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s, a) = v_\pi(s)
\end{aligned}
$$

Therefore from policy improvement theorem, $v_{\pi'}(s) \geq v_\pi(s)$

Monte Carlo Control

On-policy first-visit MC control (for ε-soft policies), estimates $\pi \approx \pi_*$

Algorithm parameter: small $\varepsilon > 0$

Initialize:

$\pi \leftarrow$ an arbitrary ε-soft policy

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

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

Repeat forever (for each episode):

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

$G \leftarrow 0$

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

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

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

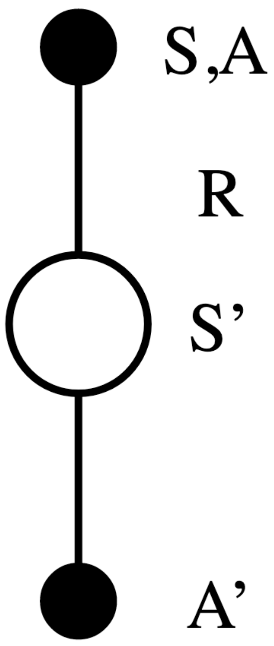
$A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)

For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$

Sarsa

- Model-free on-policy prediction (policy evaluation)
  - $V(S_t) = V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$
- Sarsa: model-free on-policy temporal-difference control
  - Sarsa: State, action, reward, state (next), action (next)
  - $Q(S_t, A_t) = Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$
  - Behavior policy: epsilon greedy
  - Target policy: epsilon greedy
  - Shown to converge to q_* if greedy in the limit with infinite exploration and if Robbins-Monro conditions hold for α



Sarsa

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$

Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
    Initialize S
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Loop 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

Q-learning

- Q-learning: model-free off-policy temporal-difference control
  - $Q(S_t, A_t) = Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$
  - Behavior policy: epsilon greedy
  - Target policy: greedy
  - Converges to q_* if Robbins-Monro conditions hold for α

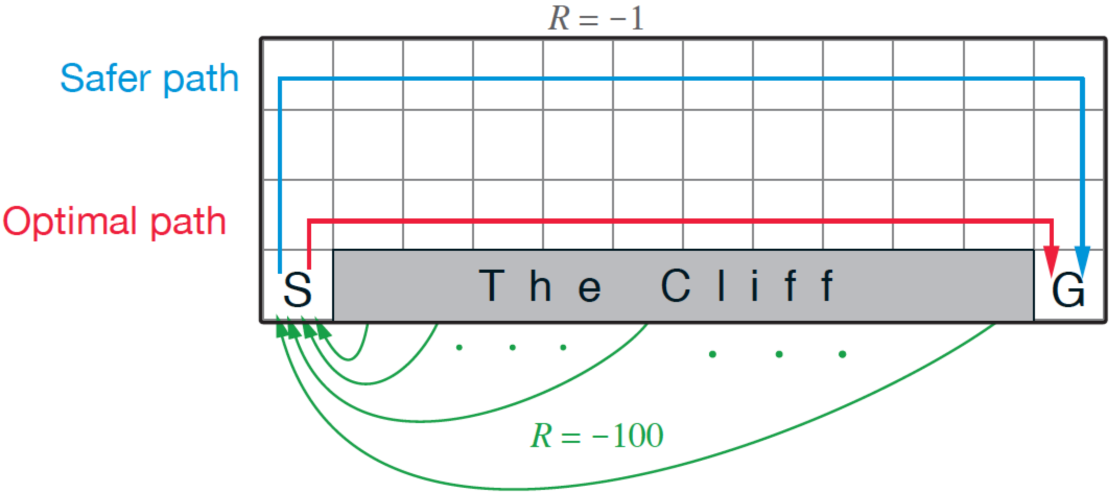
Q-learning

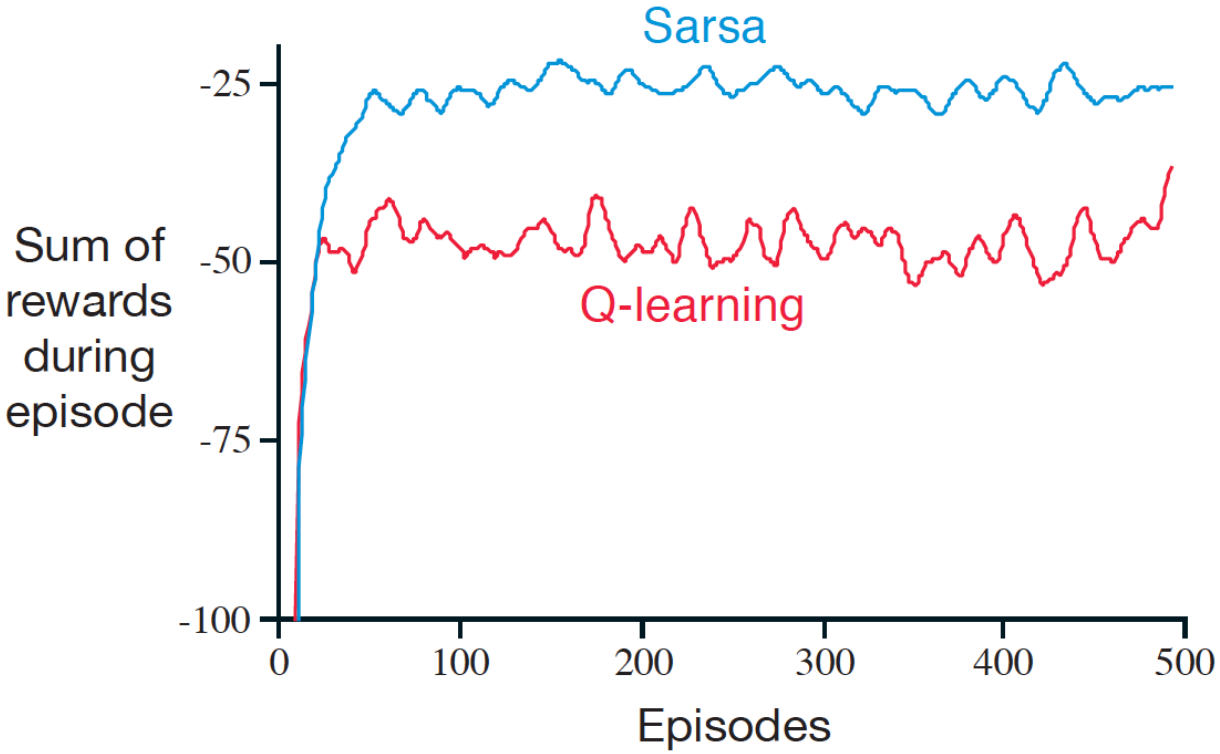
Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\textit{terminal}, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\left[R + \gamma \max_a Q(S', a) - Q(S, A)\right]$
    $S \leftarrow S'$
  until S is terminal

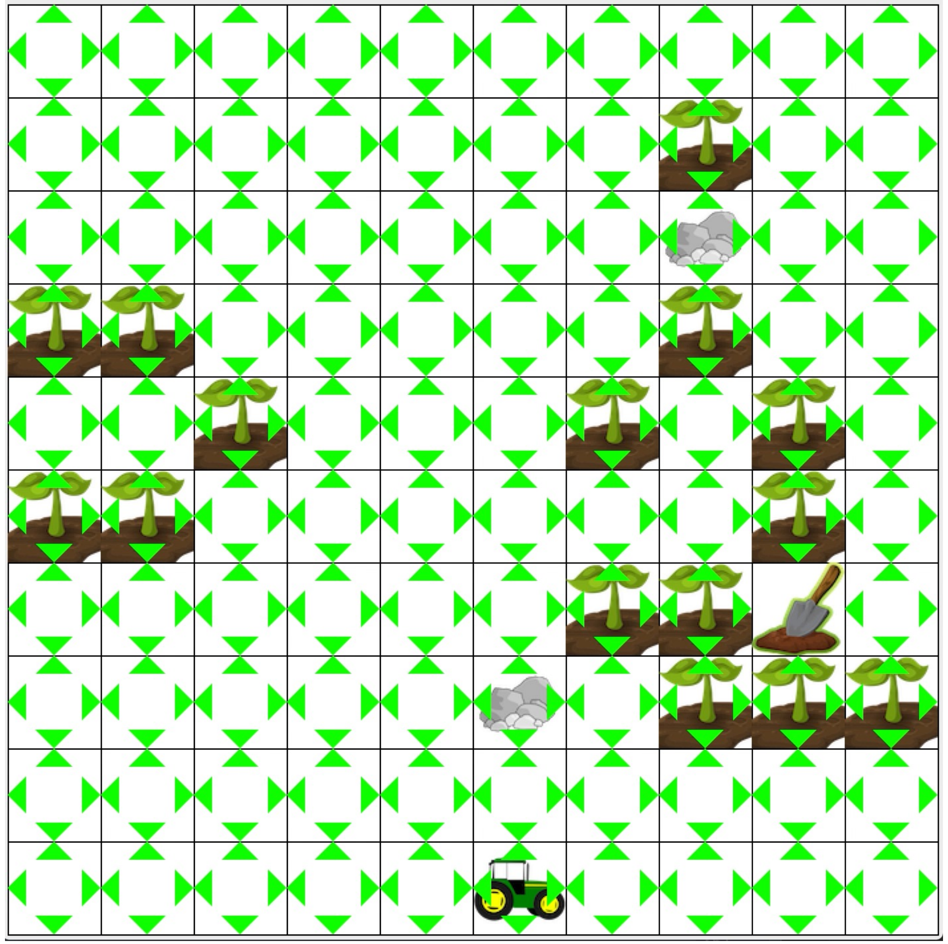
Sarsa vs Q-learning



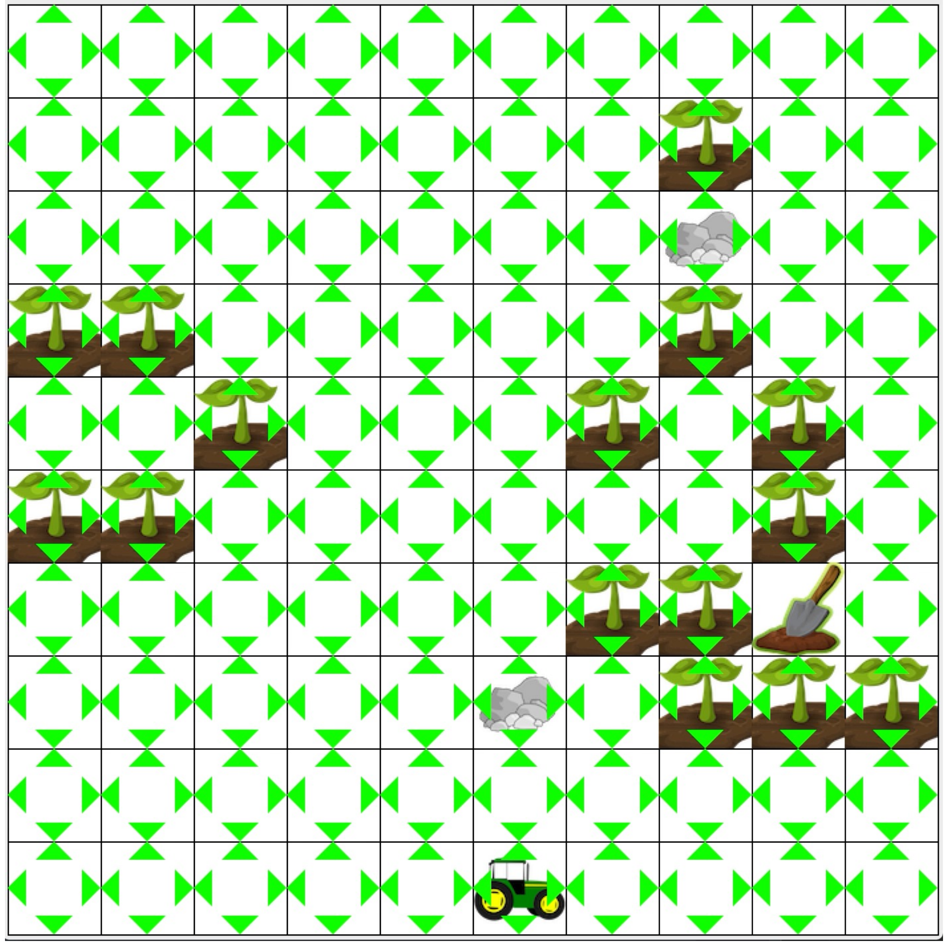


- Q-learning updates are more aggressive
- Since Q-learning is off policy, it can re-use its past experiences and even use the experience of other agents

Q-learning

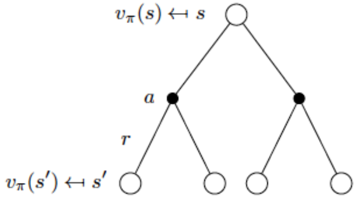
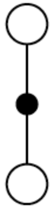
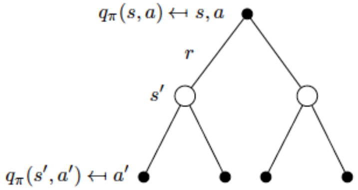
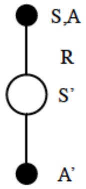
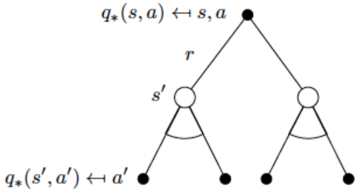
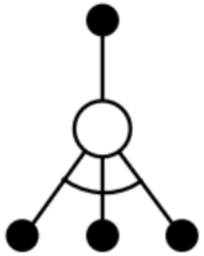


Q-learning step-by-step



Q-learning. Showing greedy policy after every 100 episodes.

Dynamic Programming and Temporal Differences

	Full Backup (DP)	*Sample Backup (TD)*
Bellman Expectation Equation for $v_\pi(s)$	 Iterative Policy Evaluation	 TD Learning
Bellman Expectation Equation for $q_\pi(s, a)$	 Q-Policy Iteration	 Sarsa
Bellman Optimality Equation for $q_*(s, a)$	 Q-Value Iteration	 Q-Learning

n-step Sarsa

- We can update our estimate of $Q(S_t, A_t)$ after n-steps
- Can speed up learning
  - Also have to tune n