ml-mipt-advanced

Lecture 08. Model-free RL

Nikolay Karpachev
25.03.2024

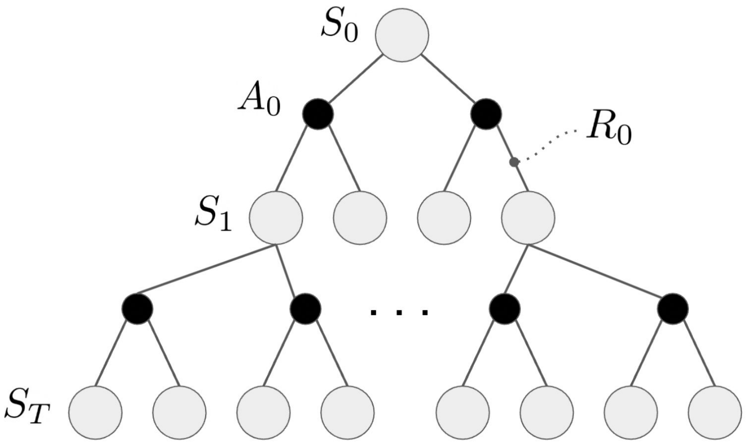
Outline

- Model-based and model-free RL
- Recap: solving mdp with dynamic programming
- Model-free prediction
    - Monte-Carlo vs. TD
- Model-free control
    - SARSA
    - Q-Learning
- Exploration / exploitation tradeoff

Model-free vs. Model-based RL

Model-based RL

1. Know **the complete dynamics** of MDP
2. Can plan ahead
3. Do not need actual experiences to estimate retrurn

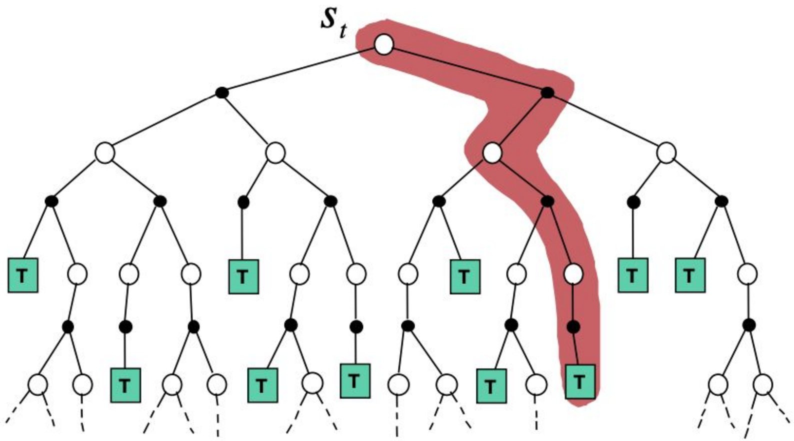


$$v_\pi(s) = \sum_a \pi(a \mid s) \sum_{r,\, s'} p(r, s' \mid s, a) \left[r + \gamma v_\pi(s')\right]$$

Model-free vs. Model-based RL

Model-free RL

1. MDP inner structure is unknown
2. Can only try stuff and estimate
3. Need samples of past experiences to learn



Recap: model-based learning

State and action-value functions

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{T-1} R_T$$

$$\begin{aligned}
v_\pi(s) &\triangleq \mathbb{E}_\pi \left[G_t \mid S_t = s \right] \\
&= \mathbb{E}_\pi \left[R_t + \gamma G_{t+1} \mid S_t = s \right] \\
&= \sum_a \pi(a \mid s) \sum_{r,\, s'} p(r, s' \mid s, a) \Big[r + \gamma \mathbb{E}_\pi \left[G_{t+1} | S_{t+1} = s' \right] \Big] \\
&= \sum_a \pi(a \mid s) \sum_{r,\, s'} p(r, s' \mid s, a) \left[r + \gamma v_\pi(s') \right]
\end{aligned}$$

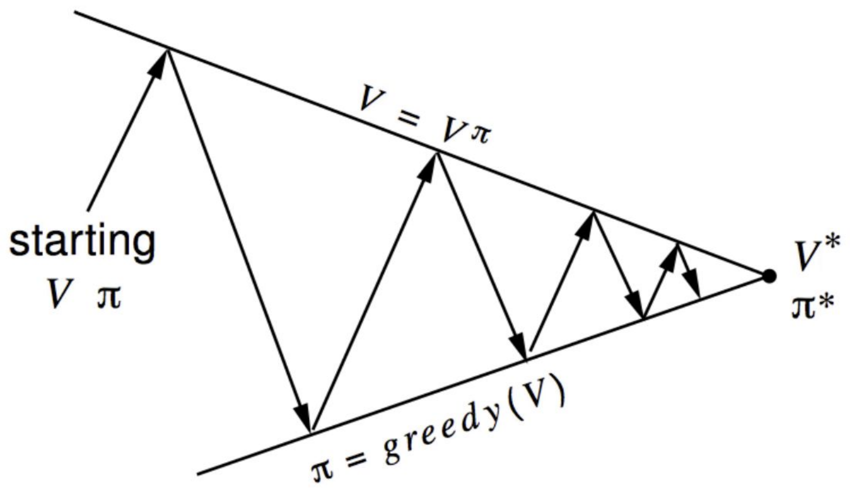
State and action-value functions

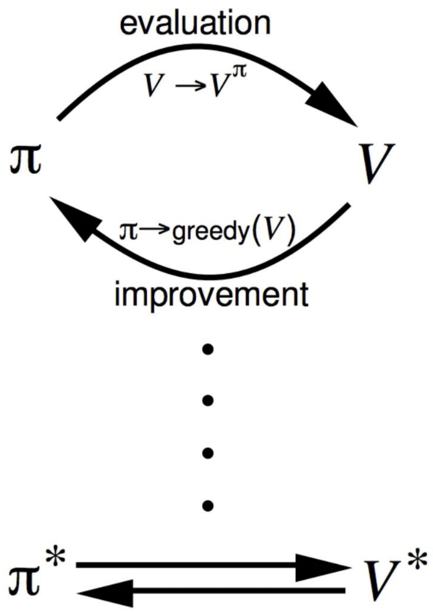
$$G_t = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{T-1} R_T$$

$$\begin{aligned}
q_\pi(s,a) &= \mathbb{E}_\pi\left[\, G_t \mid S_t = s,\, A_t = a \,\right] \\
&= \mathbb{E}_\pi\left[\, R_t + \gamma G_{t+1} \mid S_t = s,\, A_t = a \,\right] \\
&= \sum_{r,\,s'} p(r, s' \mid s, a)\Big[r + \gamma \mathbb{E}_\pi\left[\, G_{t+1} | S_{t+1} = s' \,\right]\Big] \\
&= \sum_{r,\,s'} p(r, s' \mid s, a)\left[r + \gamma v_\pi(s')\right]
\end{aligned}$$

Policy iteration

1. Policy evaluation - given policy p, estimate V_p
2. Policy improvement - improve p greedily





Policy iteration: Bellman equations

<u>Bellman expectation equations</u>

$$
\begin{aligned}
v_\pi(s) &= \sum_a \pi(a \mid s) \sum_{r,\, s'} p(r, s' \mid s, a) \left[r + \gamma v_\pi(s') \right] \\
&= \mathbb{E}_\pi \left[R_t + \gamma v_\pi(S_{t+1}) \mid S_t = s \right]
\end{aligned}
$$

$$
\begin{aligned}
q_\pi(s, a) &= \sum_{r,\, s'} p(r, s' \mid s, a) \left[r + \gamma v_\pi(s') \right] \\
&= \sum_{r,\, s'} p(r, s' \mid s, a) \Big[r + \gamma \sum_{a'} \pi(a' \mid s') q_\pi(s', a') \Big]
\end{aligned}
$$

*How to estimate V and Q functions for **a given policy** pi*

Policy iteration: Bellman equations

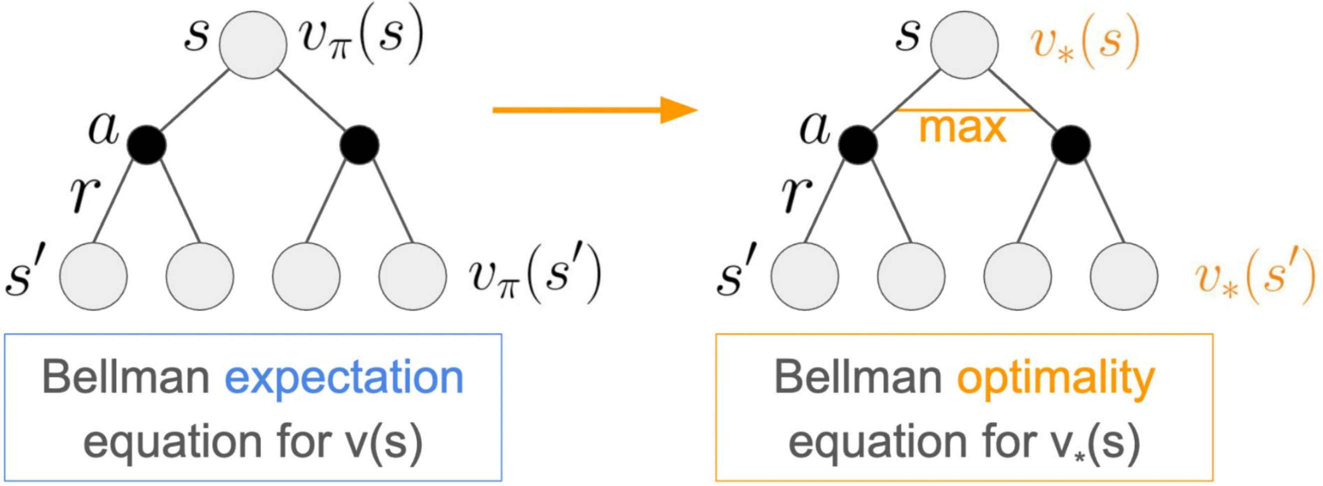
<u>Bellman optimality equations</u>

Optimal strategy

$$\pi \geq \pi' \quad \Leftrightarrow \quad v_\pi(s) \geq v_{\pi'}(s) \quad \forall\ s$$

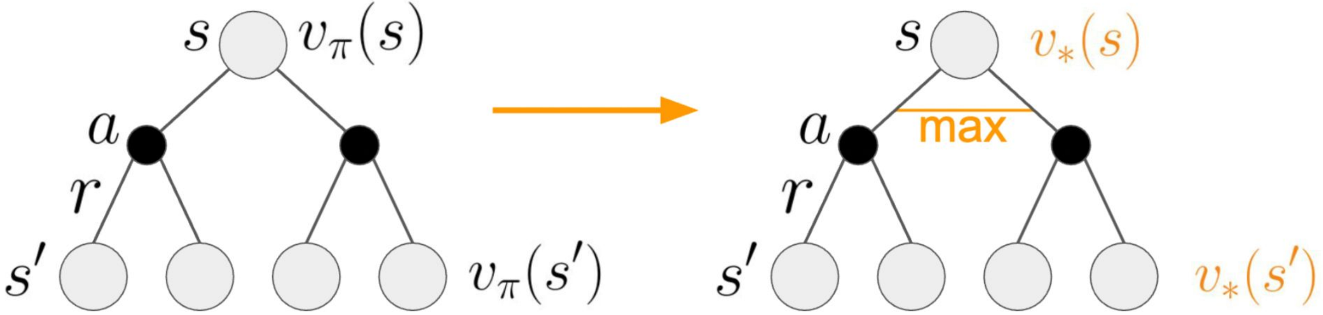
Best policy π_* is better or equal to any other policy

Policy iteration: Bellman equations



$$v_\pi(s) = \sum_a \pi(a \mid s) \sum_{r,\, s'} p(r, s' \mid s, a) \left[r + \gamma v_\pi(s') \right]$$
$$= \mathbb{E}_\pi \left[R_t + \gamma v_\pi(S_{t+1}) \mid S_t = s \right]$$

Policy iteration: Bellman equations

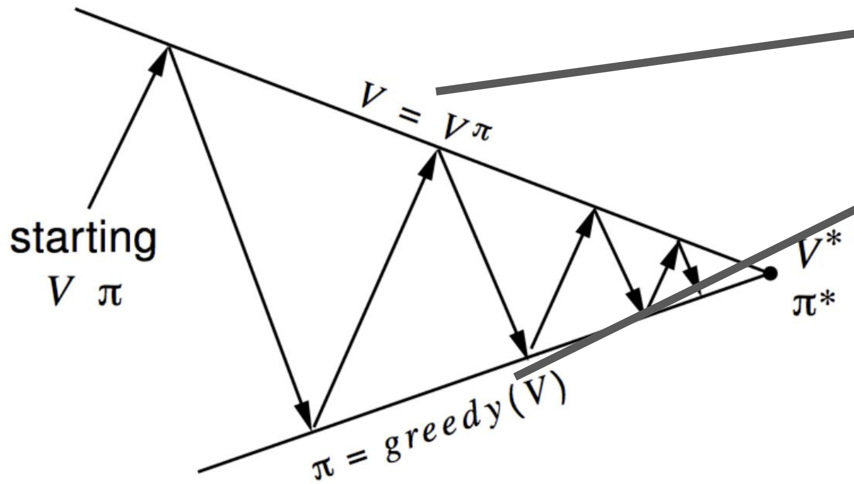


Bellman expectation equation for v(s)

Bellman optimality equation for $v_*(s)$

$$
\begin{aligned}
v_*(s) &= \max_a \sum_{r,\, s'} p(r, s' \mid s, a)\left[r + \gamma v_*(s')\right] \\
&= \max_a \; \mathbb{E}\left[R_t + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a\right]
\end{aligned}
$$

Policy iteration: convergence



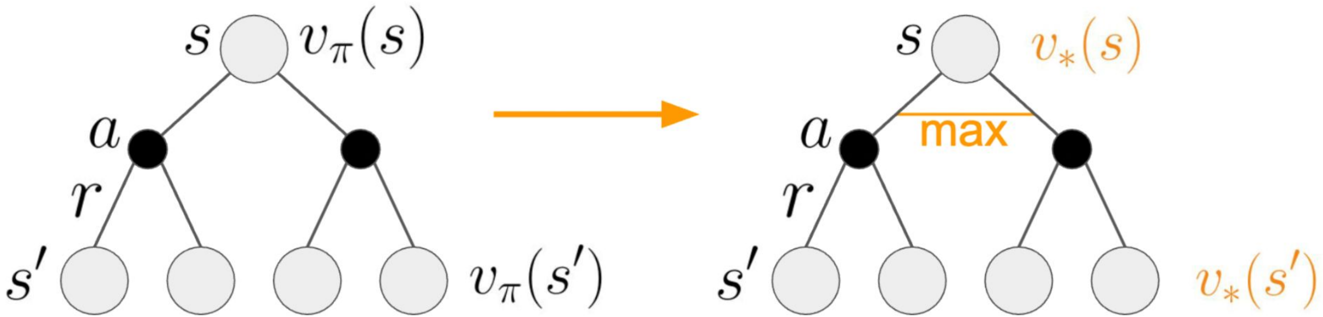
Contraction operators (both)

Hence, a fixed point exists

Model-free prediction

Model-free prediction

Q.: How to estimate V, Q functions for a given policy, **without MDP dynamics**?



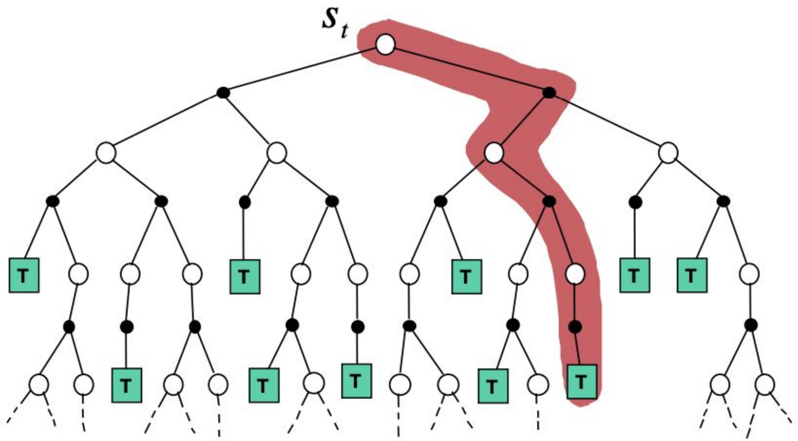
Bellman expectation equation for v(s)

Bellman optimality equation for $v_*(s)$

Any problems?

Model-free: Learning from trajectories

- Sample a lot of sessions from our current pi
- Look at the cumulative returns for each state
- Average every visit



Monte-Carlo Policy Evaluation

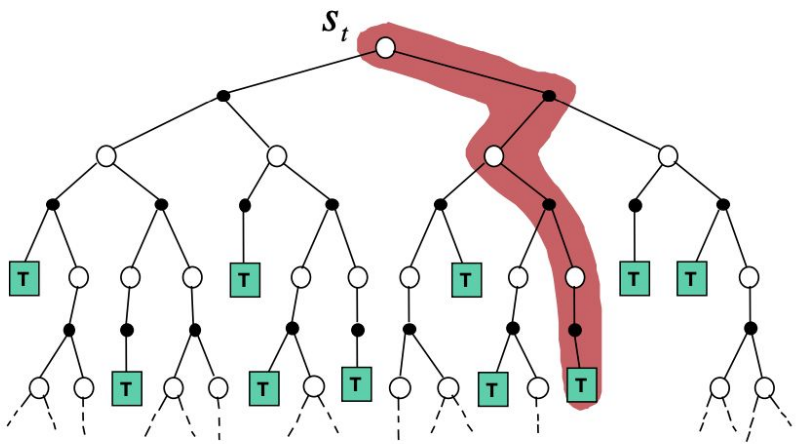
- Sample a lot of sessions from our current pi
- Every time state s is visited

$$N(s) \leftarrow N(s) + 1$$

$$S(s) \leftarrow S(s) + G_t$$

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

$$V(s) \rightarrow v_\pi(s) \text{ as } N(s) \rightarrow \infty$$



Incremental MC Policy Evaluation

Running mean updates

$$
\begin{aligned}
\mu_k &= \frac{1}{k}\sum_{j=1}^{k} x_j \\
&= \frac{1}{k}\left(x_k + \sum_{j=1}^{k-1} x_j\right) \\
&= \frac{1}{k}\left(x_k + (k-1)\mu_{k-1}\right) \\
&= \mu_{k-1} + \frac{1}{k}\left(x_k - \mu_{k-1}\right)
\end{aligned}
$$

Incremental MC Policy Evaluation

Running mean updates

$$\begin{aligned}
\mu_k &= \frac{1}{k}\sum_{j=1}^{k} x_j \\
&= \frac{1}{k}\left(x_k + \sum_{j=1}^{k-1} x_j\right) \\
&= \frac{1}{k}\left(x_k + (k-1)\mu_{k-1}\right) \\
&= \mu_{k-1} + \frac{1}{k}\left(x_k - \mu_{k-1}\right)
\end{aligned}$$

$$N(S_t) \leftarrow N(S_t) + 1$$

$$V(S_t) \leftarrow V(S_t) + \frac{1}{N(S_t)}\left(G_t - V(S_t)\right)$$

Incremental MC Policy Evaluation

Running mean updates

$$\begin{aligned}\mu_k &= \frac{1}{k}\sum_{j=1}^{k} x_j \\ &= \frac{1}{k}\left(x_k + \sum_{j=1}^{k-1} x_j\right) \\ &= \frac{1}{k}\left(x_k + (k-1)\mu_{k-1}\right) \\ &= \mu_{k-1} + \frac{1}{k}\left(x_k - \mu_{k-1}\right)\end{aligned}$$

$$N(S_t) \leftarrow N(S_t) + 1$$

$$V(S_t) \leftarrow V(S_t) + \frac{1}{N(S_t)}\left(G_t - V(S_t)\right)$$

Constant LR is useful to forget old episodes (for non-stationary setup):

$$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t - V(S_t)\right)$$

Incremental MC Policy Evaluation

Running mean updates

$$\begin{aligned}\mu_k &= \frac{1}{k}\sum_{j=1}^{k} x_j \\ &= \frac{1}{k}\left(x_k + \sum_{j=1}^{k-1} x_j\right) \\ &= \frac{1}{k}\left(x_k + (k-1)\mu_{k-1}\right) \\ &= \mu_{k-1} + \frac{1}{k}\left(x_k - \mu_{k-1}\right)\end{aligned}$$

Q.: Any problems?

$$N(S_t) \leftarrow N(S_t) + 1$$

$$V(S_t) \leftarrow V(S_t) + \frac{1}{N(S_t)}\left(G_t - V(S_t)\right)$$

Constant LR is useful to forget old episodes (for non-stationary setup):

$$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t - V(S_t)\right)$$

Temporal Difference (TD) updates

Idea:

1. **Monte-Carlo updates a V(s) guess towards a sample from true V(s) distribution**

$$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t - V(S_t)\right)$$

2. **Let's rollout only one step ahead and update a guess towards a slightly more precise guess**

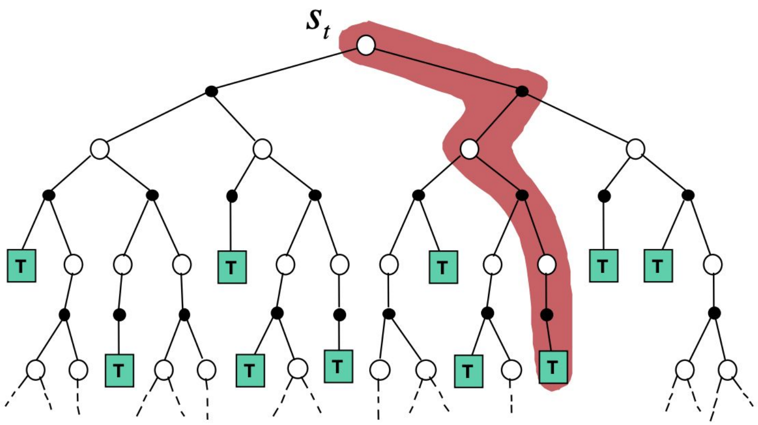
$$V(S_t) \leftarrow V(S_t) + \alpha\left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right)$$

Temporal Difference (TD) updates

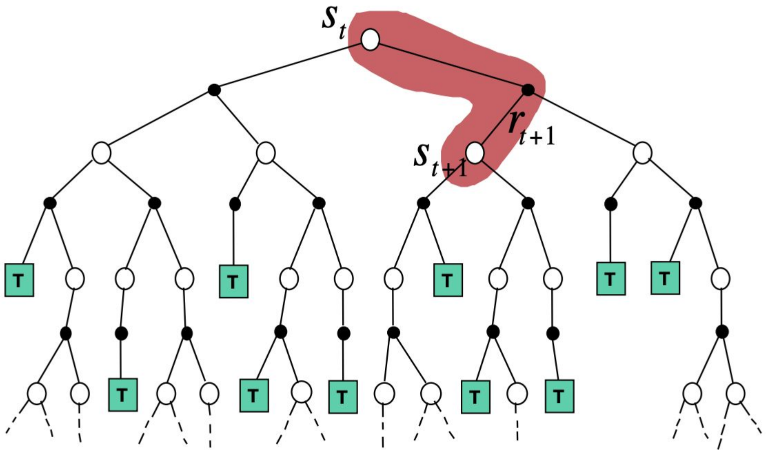
Monte-Carlo backup

Temporal Difference backup

$$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t - V(S_t)\right)$$



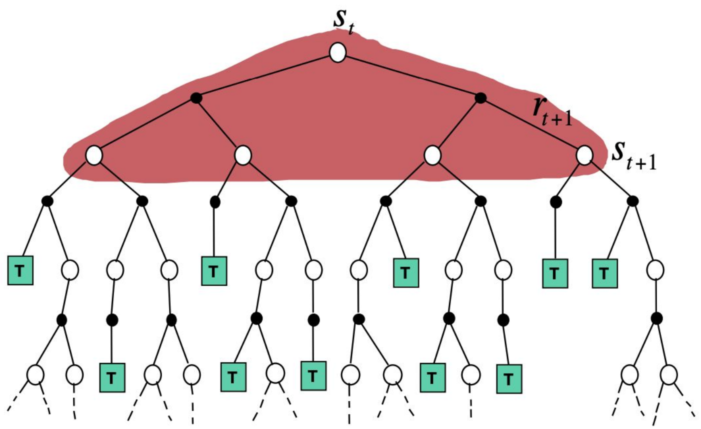
$$V(S_t) \leftarrow V(S_t) + \alpha\left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right)$$



Temporal Difference (TD) updates

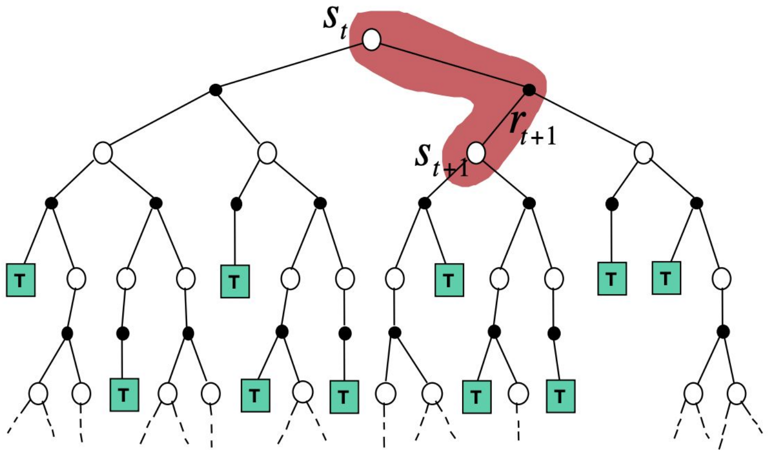
Dynamic Programming backup

$$V(S_t) \leftarrow \mathbb{E}_\pi\left[R_{t+1} + \gamma V(S_{t+1})\right]$$



Temporal Difference backup

$$V(S_t) \leftarrow V(S_t) + \alpha\left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right)$$



TD vs. MC

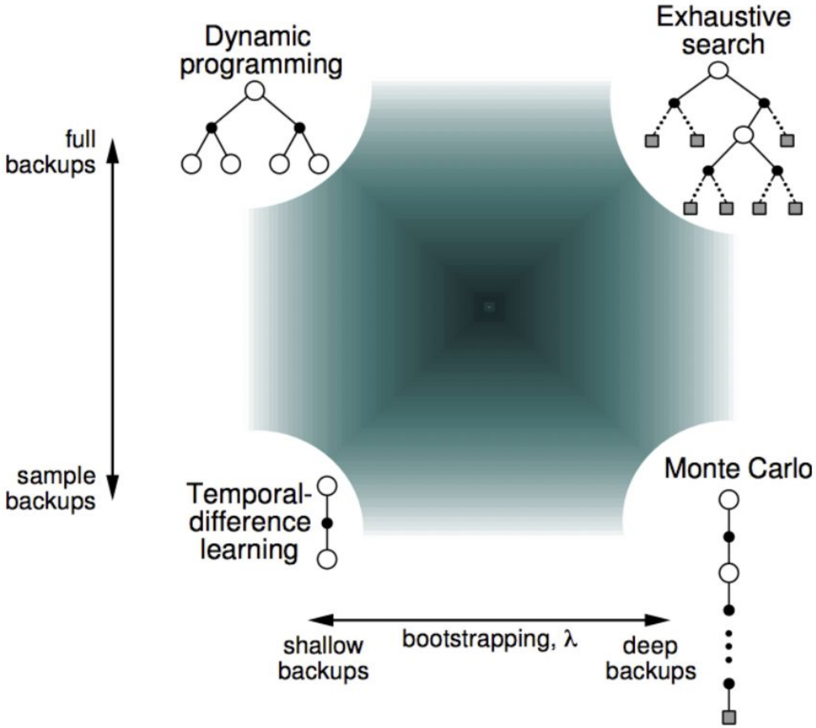
Monte-Carlo:

- honest sampling
- unbiasedly estimates expectation
- converges to MSE solution with G_t targets

TD:

- bootstrapping
- biased estimator (as “target” involves error)
- converges to solution of max likelihood Markov model

Value-based methods. Unified view.



TD(n)

- Use n-step rollouts instead of 1-step
- Slightly more accurate bootstraps

Q.: what is TD(inf)?

TD(n)

- Use n-step rollouts instead of 1-step
- Slightly more accurate bootstraps

$$
\begin{array}{lll}
n = 1 & (TD) & G_t^{(1)} = R_{t+1} + \gamma V(S_{t+1}) \\
n = 2 & & G_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 V(S_{t+2}) \\
\vdots & & \vdots \\
n = \infty & (MC) & G_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{T-1} R_T
\end{array}
$$

$$V(S_t) \leftarrow V(S_t) + \alpha \left(G_t^{(n)} - V(S_t) \right)$$

Model-free control

Model-free control

Model-free prediction

- ***Estimate*** *the value function of unknown MDP*

Model-free control

- ***Optimize*** *the value function of unknown MDP*

Model-free control

Q.: What to learn?

$V(S)$ or $Q(S, A)$

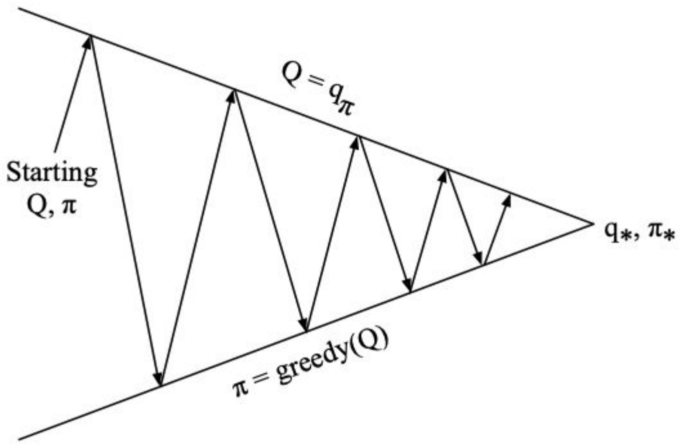
Model-free control

Q.: What to learn?

V(S) or **<u>Q(S, A)</u>**

V(S) is useless for action prediction w/o model dynamics

Policy Iteration with Q-function



1. Policy evaluation
   estimate Q using MC sampling
2. Policy improvement
   greedy improvement

Exploration / exploitation tradeoff

- Two doors
- Try left door, V(left) = 0
- Try right door, V(right) = 1

Q.: Is the right door optimal?

Exploration / exploitation tradeoff

- Two doors
- Try left door, V(left) = 0
- Try right door, V(right) = 1

<u>Q.: Is the right door optimal?</u>

- *Stochasticity in environment*
- *Need to ensure continual exploration of options*

"Behind one door is tenure - behind the other is flipping burgers at McDonald's."

Exploration / exploitation tradeoff

Epsilon-greedy exploration

- with probability epsilon, pick random action uniformly
- with probability (1 - epsilon), pick current best choice

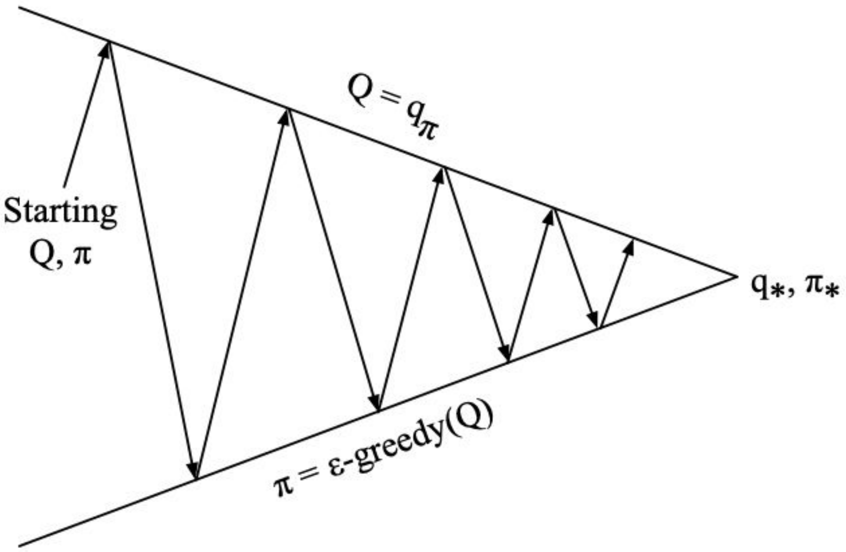
$$\pi(a|s) = \begin{cases} \epsilon/m + 1 - \epsilon & \text{if } a^* = \underset{a \in \mathcal{A}}{\text{argmax}}\ Q(s, a) \\ \epsilon/m & \text{otherwise} \end{cases}$$

Epsilon-greedy policy improvement

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

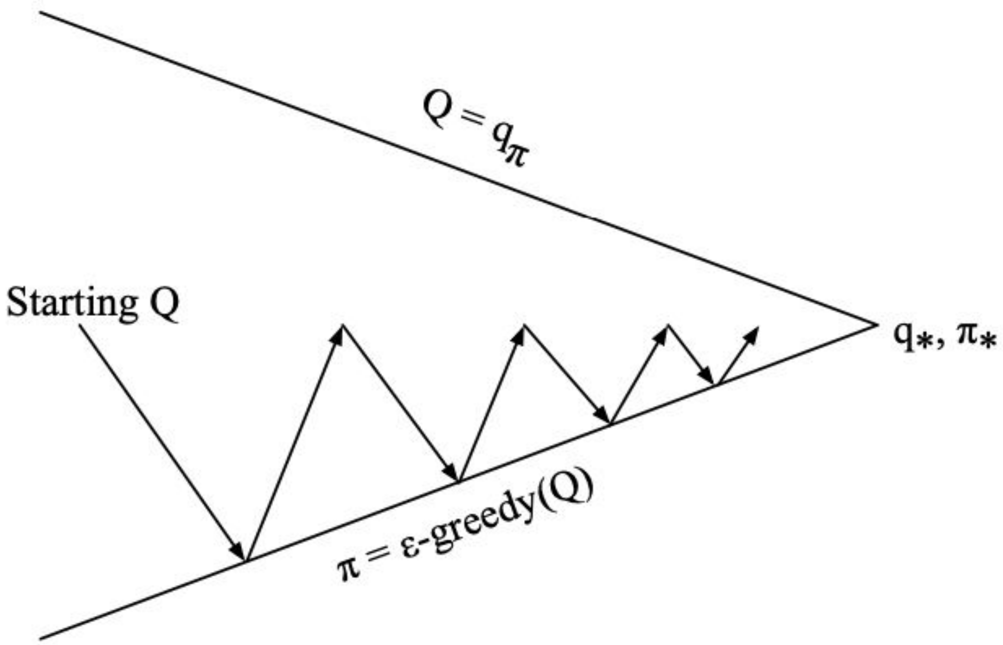
$$\boxed{v_{\pi'}(s) \geq v_\pi(s)}$$

Monte-Carlo Policy Iterarion



1. Policy evaluation
   estimate Q using MC sampling
2. Policy improvement
   eps-greedy improvement

Monte-Carlo Control



Just like in value iteration

<u>on every episode</u> *(!)*

1. *Approx. policy evaluation*
2. *Policy improvement*

Model Free Control: SARSA

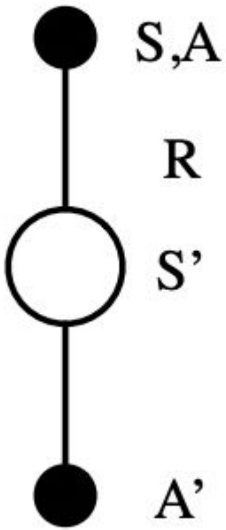
Basically, SARSA is

- One-step Temporal Difference Policy Evaluation
- Epsilon-greedy Policy Improvement

Model Free Control: SARSA

Basically, SARSA is

- One-step Temporal Difference Policy Evaluation
- Epsilon-greedy Policy Improvement



$$Q(S,A) \leftarrow Q(S,A) + \alpha \left(R + \gamma Q(S',A') - Q(S,A)\right)$$

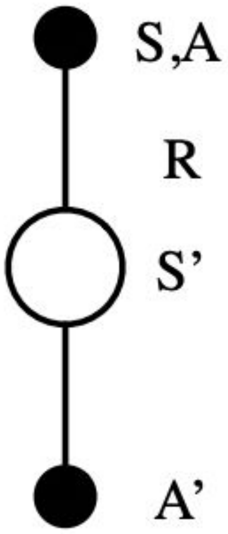
Model Free Control: SARSA

Basically, SARSA is

- One-step Temporal Difference Policy Evaluation
- Epsilon-greedy Policy Improvement

$$Q(S,A) \leftarrow Q(S,A) + \alpha \left(R + \gamma Q(S',A') - Q(S,A)\right)$$

Policy evaluation



Sample from experience

Model Free Control: n-step SARSA

$$
\begin{array}{lll}
n = 1 & (Sarsa) & q_t^{(1)} = R_{t+1} + \gamma Q(S_{t+1}) \\
n = 2 & & q_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 Q(S_{t+2}) \\
\vdots & & \vdots \\
n = \infty & (MC) & q_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{T-1} R_T
\end{array}
$$

Use n-step partial rollouts and TD(n) estimates

$$
Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left(q_t^{(n)} - Q(S_t, A_t) \right)
$$

Off-policy and on-policy learning

Off-policy vs. on-policy

On-policy learning

- Training episodes are sampled from the current policy we are optimizing

Off-policy learning

- Training episodes are taken elsewhere (e.g. old policy, external policy, some data with unknown origin)

Off-policy model-free control: Q-Learning

Key idea: optimize greedy policy

- Our policy is greedy (or eps-greedy) w.r.t. current q-values
- Hence, no A' is required in (S, A, R, S', A') updates

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

Off-policy model-free control: Q-Learning

Key idea: optimize greedy policy

- Our policy is greedy (or eps-greedy) w.r.t. current q-values
- Hence, no A' is required in (S, A, R, S', A') updates

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

$$\pi(S_{t+1}) = \underset{a'}{\operatorname{argmax}}\, Q(S_{t+1}, a')$$

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

Off-policy model-free control: Q-Learning

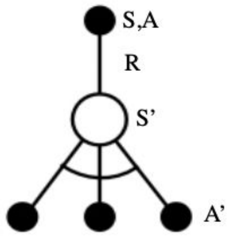
<u>Key idea:</u> optimize greedy policy

- Our policy is greedy (or eps-greedy) w.r.t. current q-values
- Hence, no A’ is required in (S, A, R, S’, A’) updates

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

$$\pi(S_{t+1}) = \underset{a'}{\operatorname{argmax}}\ Q(S_{t+1}, a')$$

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



Off-policy model-free control: EV-SARSA

EV-SARSA (Expected Value SARSA)

Use current policy to get expectation of V(s')

Q-learning: greedy policy // EV-SARSA: some policy

$$\hat{Q}(s,a) = r(s,a) + \boxed{\gamma \underset{a_i \sim \pi(a|s')}{E} Q(s',a_i)}$$

"Target" Q from trajectory

Off-policy vs. on-policy

On-policy

- Agent trains on experience generated with its own policy
- Can't learn off-policy

Examples:
- Cross-entropy method
- SARSA

Off-policy

- Agent trains on any kind of experience
- Can still learn on-policy

Examples:
- Q-learning
- EV-SARSA

Outro

- Model-based and model-free RL
- Recap: solving mdp with dynamic programming
- Model-free prediction
  - Monte-Carlo vs. TD
- Model-free control
  - SARSA
  - Q-Learning
- Exploration / exploitation tradeoff

Acknowledgements

This lecture uses materials from
(1) RL Lectures by David Siver (licensed CC-BY-NC 4.0)
(2) Practical_RL lectures by Yandex Data School (Unlicense license)

Questions?