Model Free Reinforcement Learning

I-Chen Wu

- Sutton, R.S. and Barto, A.G., Reinforcement Learning: An Introduction, MIT Press, Cambridge, MA, 1998. (Bible for RL)
  - http://webdocs.cs.ualberta.ca/~sutton/book/ebook/the-book.html
  - Chapters 5-6
- David Silver, Online Course for Deep Reinforcement Learning.
  - http://www.cs.ucl.ac.uk/staff/D.Silver/web/Teaching.html
  - Chapters 4-5

Outline

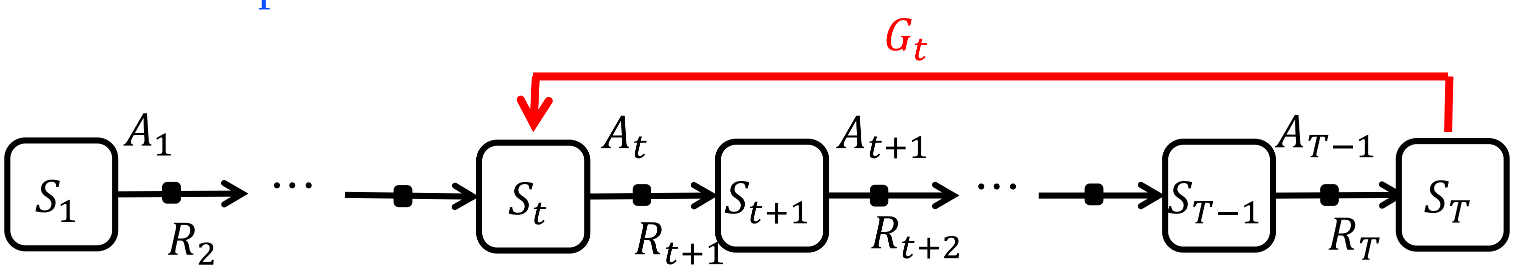
- Model-free prediction: Estimate the value function of an unknown MDP
  – Monte-Carlo (MC) Learning
  – Temporal Difference (TD) Learning
- Model-free control: Approximate optimal policies based on the estimation of the value function.
  – On-Policy Monte-Carlo Control
  – On-Policy Temporal-Difference Learning
  – Off-Policy Learning

The purpose of this chapter:

- Learn model-free RL: MC + TD

Monte-Carlo Reinforcement Learning

- MC methods learn directly from episodes of experience
- MC is model-free:
  - no knowledge of MDP transitions / rewards
- MC learns from complete episodes:
  - no bootstrapping
- MC uses the simplest possible idea:
  - value = mean return
- Caveat: can only apply MC to episodic MDPs
  - All episodes must terminate



Monte-Carlo Policy Evaluation

- Goal: learn v_π from episodes of experience under policy π

  $S_1, A_1, R_2, \dots, S_T \sim \pi$

- Recall that the return is the total discounted reward:

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

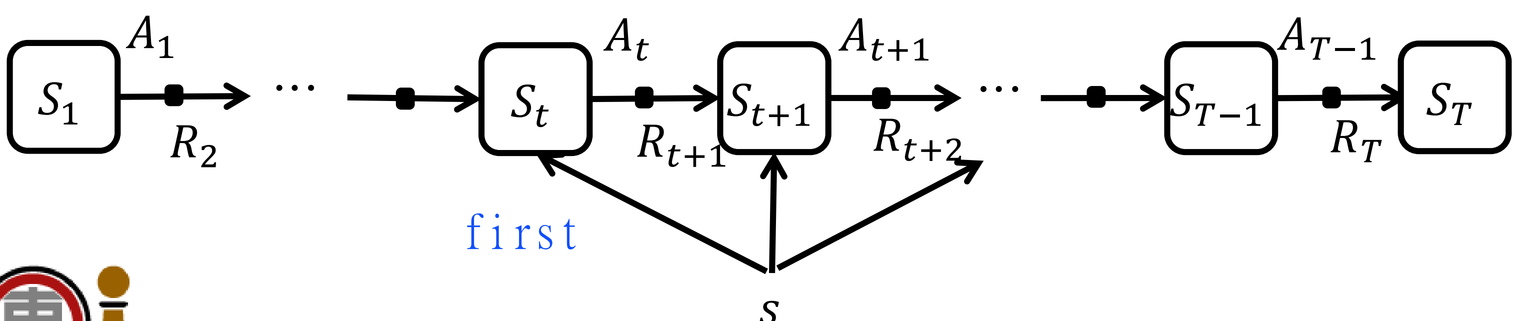
- Recall that the value function is the expected return:

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

- Monte-Carlo policy evaluation uses empirical mean return instead of expected return

Monte-Carlo Policy Evaluation (cont.)

- To evaluate $v_\pi(s)$ at state s
  - Every time-step t that state s is visited in an episode,
    - Sometimes, we also consider the first time-step t.
    - Both converge quadratically, so this issue is ignored in this course.
  - Increment counter $N(s) \leftarrow N(s) + 1$
  - Increment total return $S(s) \leftarrow S(s) + G_t$
  - Value is estimated by mean return $V(s) \leftarrow S(s)/N(s)$
- By law of large numbers, $V(s) \rightarrow v_\pi(s)$ *as* $N(s) \rightarrow \infty$



Incremental Mean

The mean $\mu_1, \mu_2, \ldots$ of a sequence $x_1, x_2, \ldots$ can be computed incrementally,

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

Incremental Monte-Carlo Updates

- Update $V(s)$ incrementally after episode $S_1, A_1, R_2, \ldots, S_T$
- For each state S_t with return G_t

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

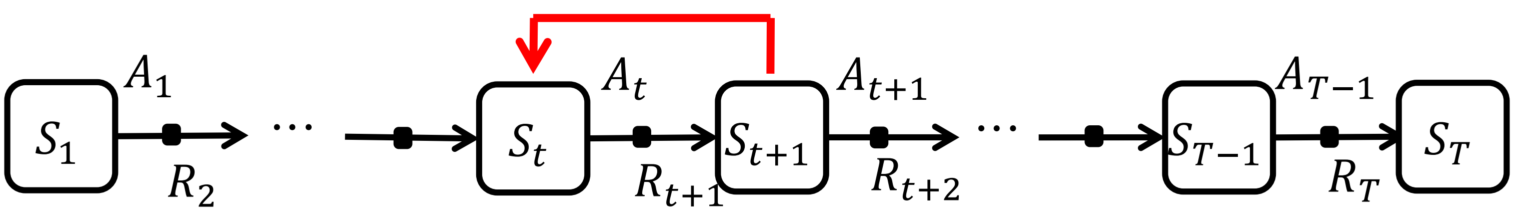
$$V(S_t) \leftarrow V(S_t) + \frac{1}{N(S_t)}\left(G_t - V(S_t)\right)$$

- In non-stationary problems, it can be useful to track a running mean, i.e. forget old episodes.

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

Temporal-Difference Learning

- TD methods learn directly from episodes of experience
- TD is model-free:
  - no knowledge of MDP transitions / rewards
- TD learns from incomplete episodes,
  - by bootstrapping
- TD updates a guess towards a guess



MC vs. TD

- Goal: learn v_π online from experience under policy π
- Incremental every-visit Monte-Carlo
  - Update value $V(S_t)$ toward actual return G_t
    $$V(S_t) \leftarrow V(S_t) + \alpha\big(G_t - V(S_t)\big)$$
- Simplest temporal-difference learning algorithm: TD(0)
  - Update value $V(S_t)$ toward estimated return $R_{t+1} + \gamma V(S_{t+1})$
    $$V(S_t) \leftarrow V(S_t) + \alpha\big(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\big)$$
  - $R_{t+1} + \gamma V(S_{t+1})$ is called the TD target
  - $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ is called the TD error

TD vs. MC (I)

- TD can learn before knowing the final outcome
  - TD can learn online after every step
  - MC must wait until end of episode before return is known
- TD can learn without the final outcome
  - TD can learn from incomplete sequences
  - MC can only learn from complete sequences
  - TD works in continuing (non-terminating) environments
  - MC only works for episodic (terminating) environments

Bias/Variance Trade-Off

- TD target $R_{t+1} + \gamma V(S_{t+1})$ is biased estimate of $v_\pi(S_t)$
  - Return $G_t = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-1} R_T$ is unbiased estimate of $v_\pi(S_t)$
  - True TD target $R_{t+1} + \gamma v_\pi(S_{t+1})$ is unbiased estimate of $v_\pi(S_t)$
- TD target is much lower variance than the return:
  - Return depends on many random actions, transitions, rewards
  - TD target depends on only one random action, transition, reward

MC vs. TD (II)

- MC has high variance, zero bias
  - Good convergence properties (even with function approximation)
  - Not very sensitive to initial value
  - Very simple to understand and use
- TD has low variance, some bias
  - Usually more efficient than MC
  - TD(0) converges to $v_\pi(s)$ (but not always with function approximation)
  - More sensitive to initial value

Batch MC and TD

- MC and TD converge: $V(s) \rightarrow v_\pi(s)$ as experience $\rightarrow \infty$
- But what about batch solution for finite experience?

$$s_1^1, a_1^1, r_2^1, \ldots, s_{T_1}^1$$
$$\vdots$$
$$s_1^k, a_1^k, r_2^k, \ldots, s_{T_k}^k$$

  - e.g. Repeatedly sample episode $k \in [1, K]$
  - Apply MC or TD(0) to episode k

AB Example

Two states A, B; no discounting; 8 episodes of experience

A, 0, B, 0

B, 1

B, 1

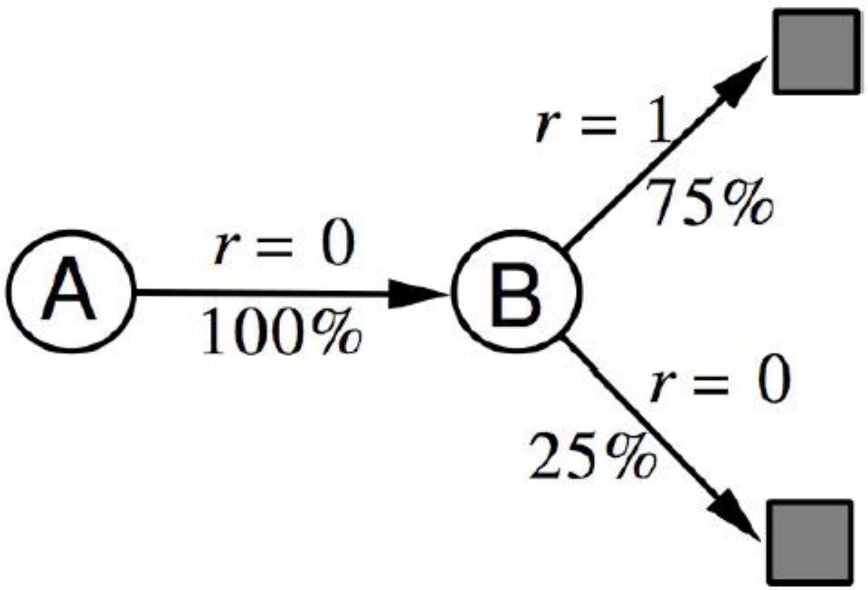
B, 1

B, 1

B, 1

B, 1

B, 0



What is $V(A), V(B)$?

Both MC and TD will obtain different values!!

Certainty Equivalence

- MC converges to solution with minimum mean-squared error
  - Best fit to the observed returns

$$\sum_{k=1}^{K}\sum_{t=1}^{T_k}\left(G_t^k - V(s_t^k)\right)^2$$

  - In the AB example, $V(A) = 0$, $V(B) = 0.75$
- TD(0) converges to solution of max likelihood Markov model
  - Solution to the MDP <$\mathcal{S}$, $\mathcal{A}$, $\hat{\mathcal{P}}$, $\hat{\mathcal{R}}$, γ> that best fits the data

$$\hat{\mathcal{P}}_s^a = \frac{1}{N(s,a)}\sum_{k=1}^{K}\sum_{t=1}^{T_k} 1\left(s_t^k, a_t^k, s_{t+1}^k = s, a, s'\right)$$

$$\hat{\mathcal{R}}_s^a = \frac{1}{N(s,a)}\sum_{k=1}^{K}\sum_{t=1}^{T_k} 1\left(s_t^k, a_t^k = s, a\right) r_t^k$$

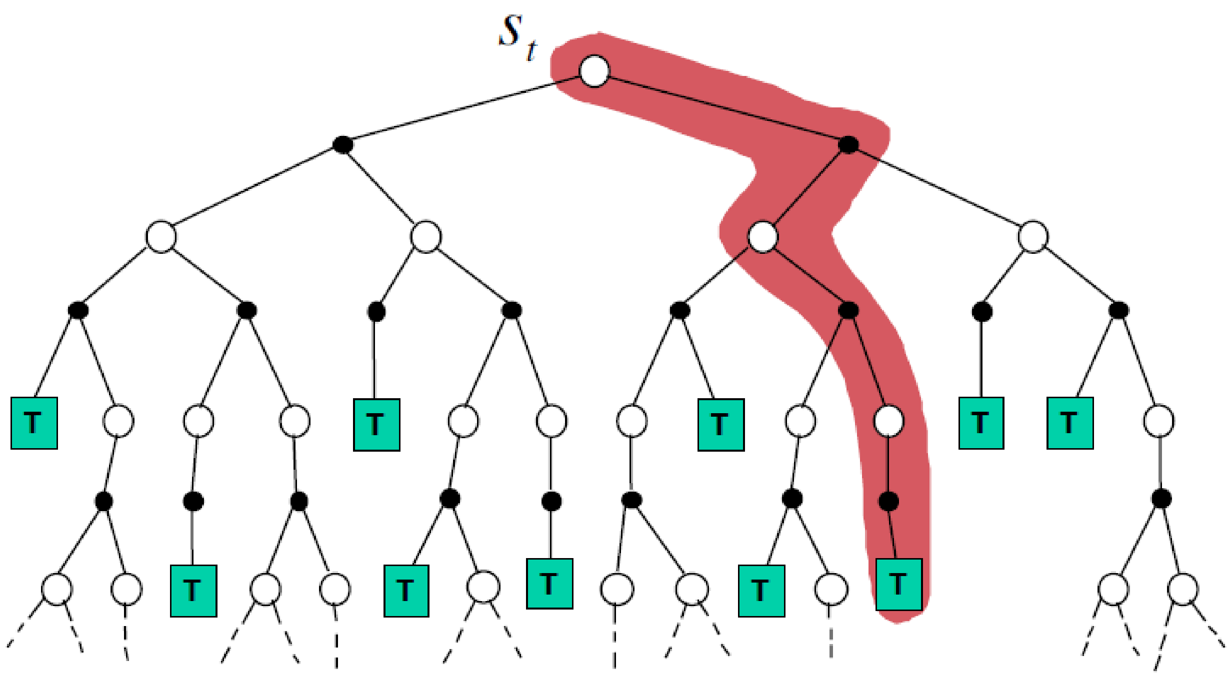
  - In the AB example, $V(A) = 0.75$, $V(B) = 0.75$

MC vs. TD (III)

- TD exploits Markov property
  - Usually more efficient in Markov environments
    - So, TD works well for MDP problems like 2048.
- MC does not exploit Markov property
  - Usually more effective in non-Markov environments
    - MC works fine for non-MDP too.

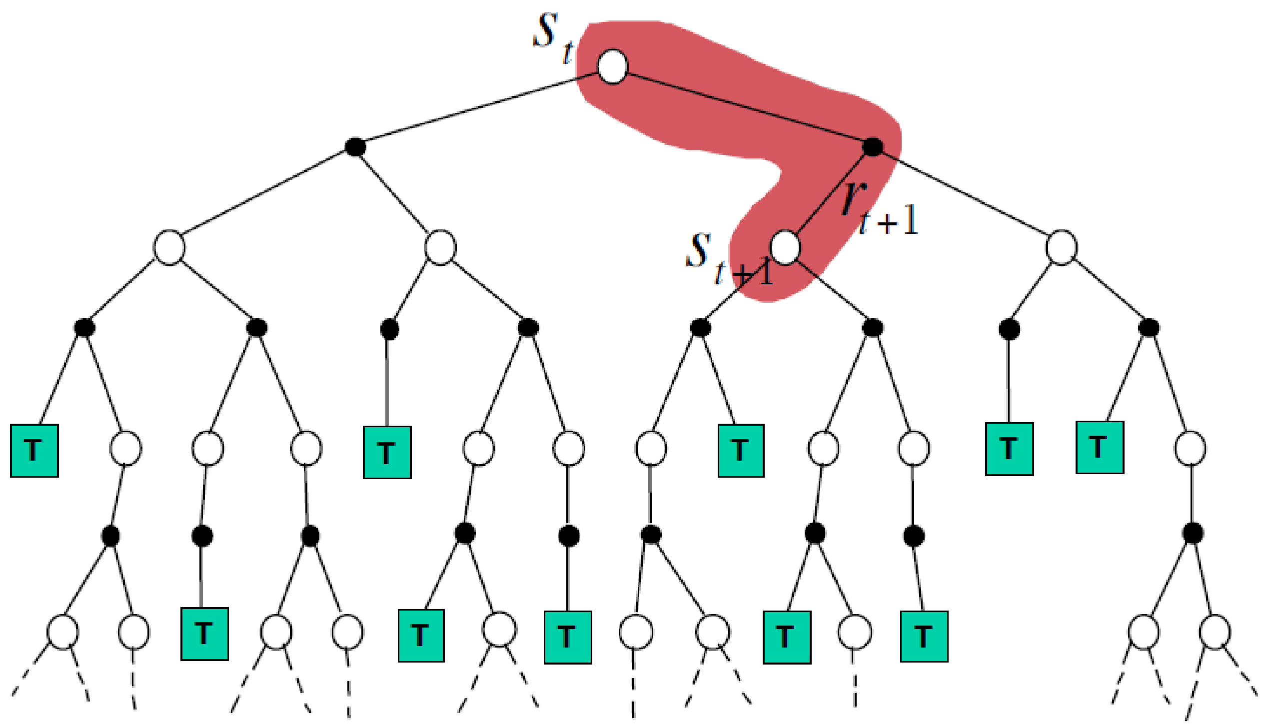
Monte-Carlo Backup

$$V(S_t) \leftarrow V(S_t) + \alpha\big(G_t - V(S_t)\big)$$



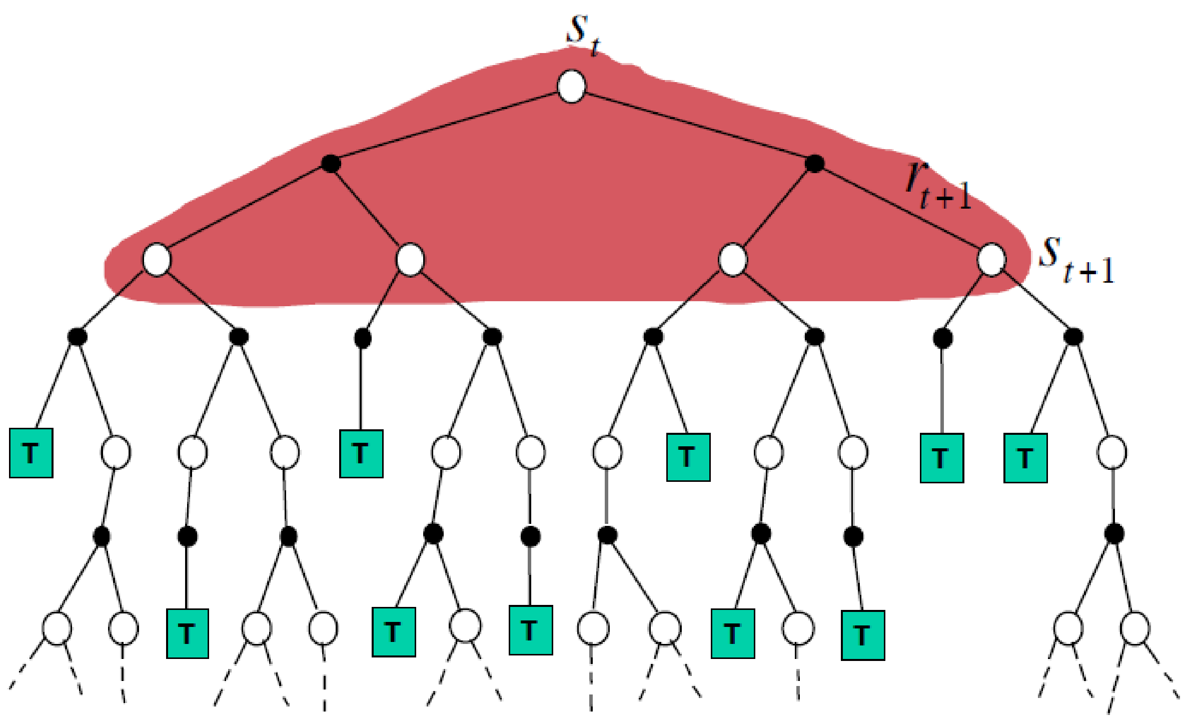
Temporal-Difference Backup

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



Dynamic Programming Backup

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



Bootstrapping and Sampling

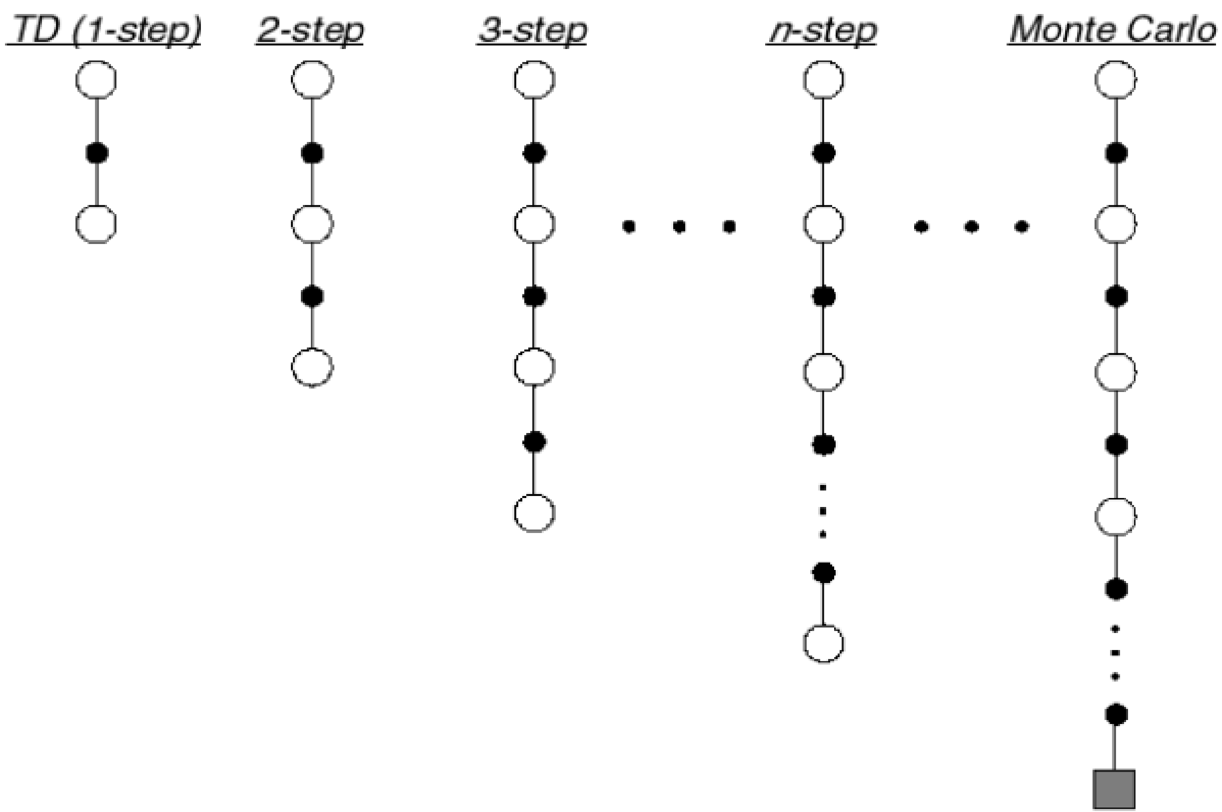
- Bootstrapping: update involves an estimate
  - MC does not bootstrap
  - DP bootstraps
  - TD bootstraps
- Sampling: update samples an expectation
  - MC samples
  - DP does not sample
  - TD samples

General TD Learning

- Review TD
  - Update value $V(S_t)$ toward estimated return $R_{t+1} + \gamma V(S_{t+1})$
    $$V(S_t) \leftarrow V(S_t) + \alpha\left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right)$$
  - $R_{t+1} + \gamma V(S_{t+1})$ is called the TD target
- Question: a more general TD target?
- Investigate TD in a more general manner.
- A typical one: TD(λ)

n-Step Prediction

- Let TD target look n steps into the future



n-Step Return

- Consider the following n-step returns for $\text{n} = 1,2,\infty$

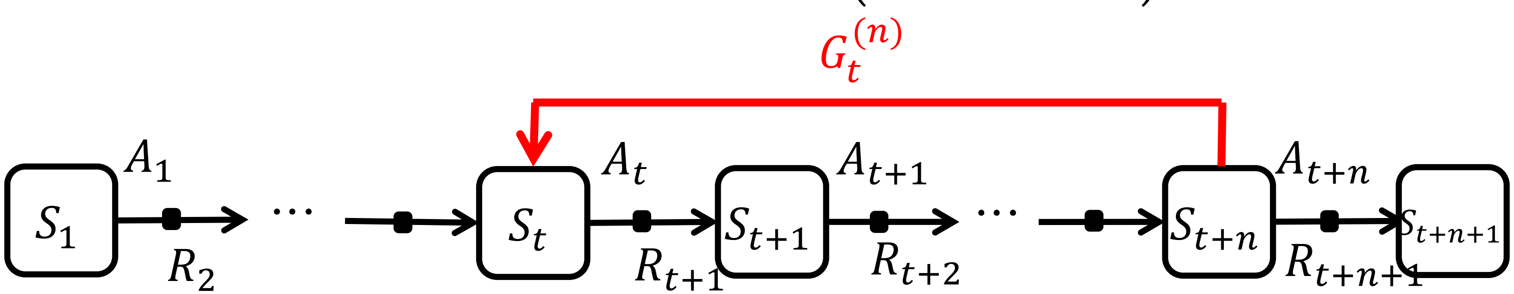
$$
\begin{array}{ll}
\text{n} = 1 & G_t^{(1)} = R_{t+1} + \gamma V(S_{t+1}) \\
\text{n} = 2 & G_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 V(S_{t+2}) \\
\vdots & \vdots \\
\text{n} = \infty & G_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{T-1} R_T
\end{array}
$$

- Define the n-step return

$$G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})$$

- n-step temporal-difference learning

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

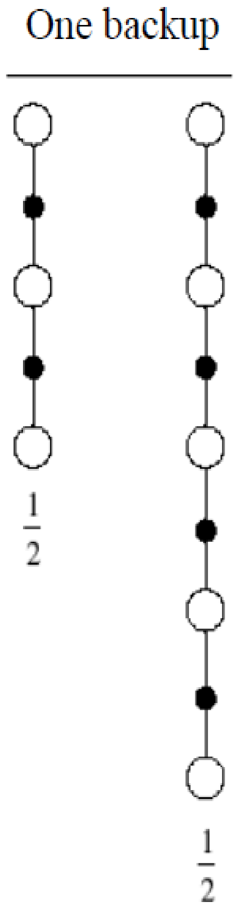


Example of Averaging n-Step Returns

- We can average n-step returns over different n
- Example:
  - average the 2-step and 4-step returns

$$\frac{1}{2}G^{(2)} + \frac{1}{2}G^{(4)}$$

  - Combines information from two different time-steps
- Next:
  - combine information from all time-steps?



λ-return

- λ-return G_t^{λ}:
  - combines all n-step returns $G_t^{(n)}$
- Using weight$(1-\lambda)\,\lambda^{n-1}$

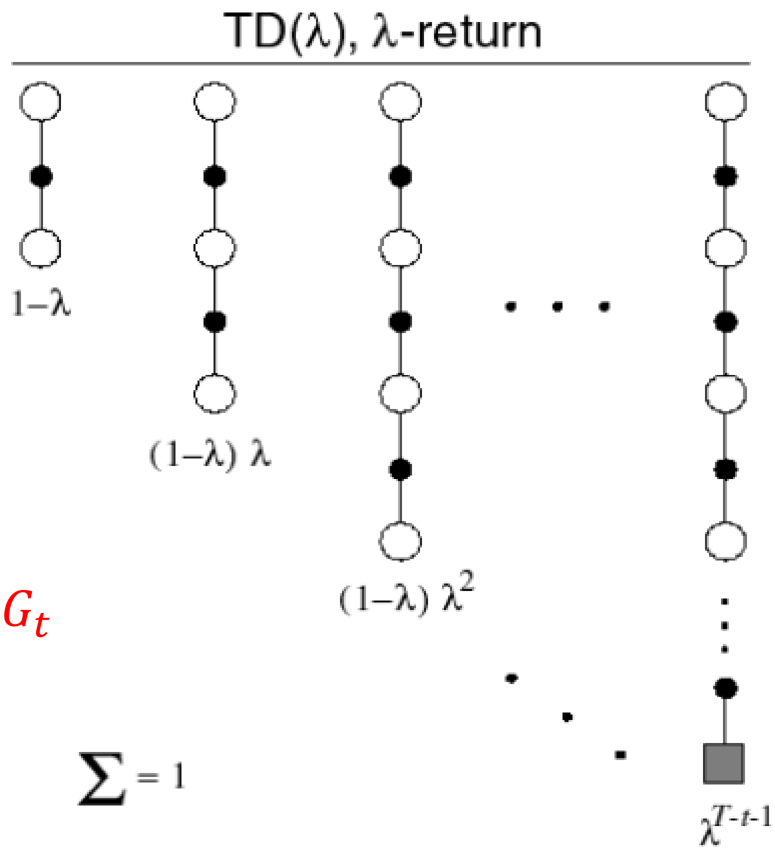
$$G_t^{\lambda} = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}\,G_t^{(\mathrm{n})}$$

or (in the case of termination)

$$G_t^{\lambda} = (1-\lambda)\sum_{n=1}^{T-t-1}\lambda^{n-1}\,G_t^{(\mathrm{n})} + \lambda^{T-t-1}G_t$$

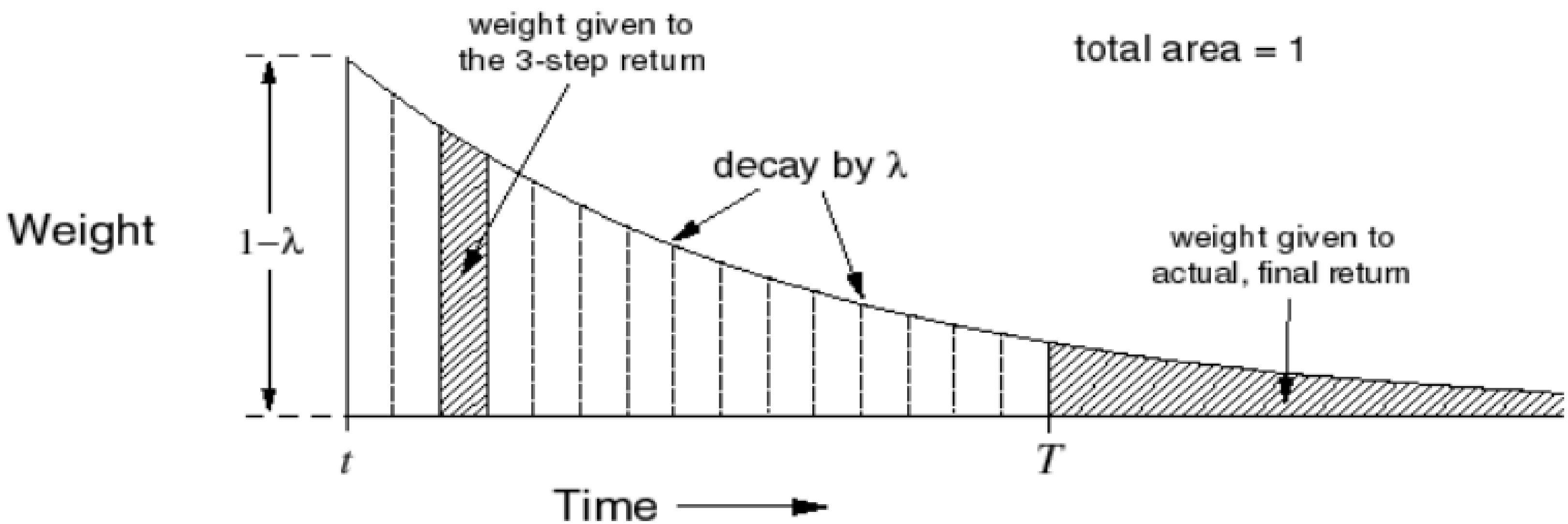
- Forward-view TD(λ)

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



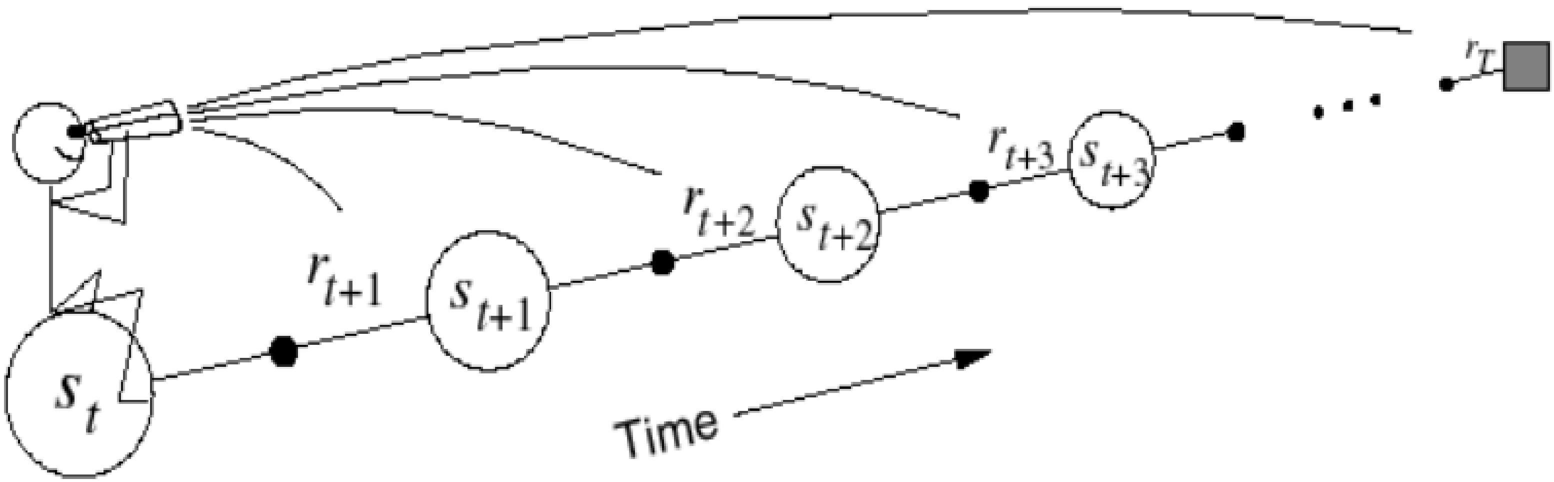
TD(λ) Weighting Function

$$G_t^{\lambda} = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1}\, G_t^{(\mathrm{n})}$$



Forward-view TD(λ)

- Update value function towards the λ-return
- Forward-view looks into the future to compute G_t^λ
- Like MC, can only be computed from complete episodes

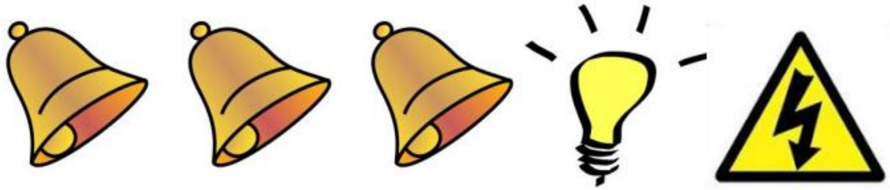


Backward View TD(λ)

- Forward view provides theory
- Backward view provides mechanism
  - Update online, every step, from incomplete sequences

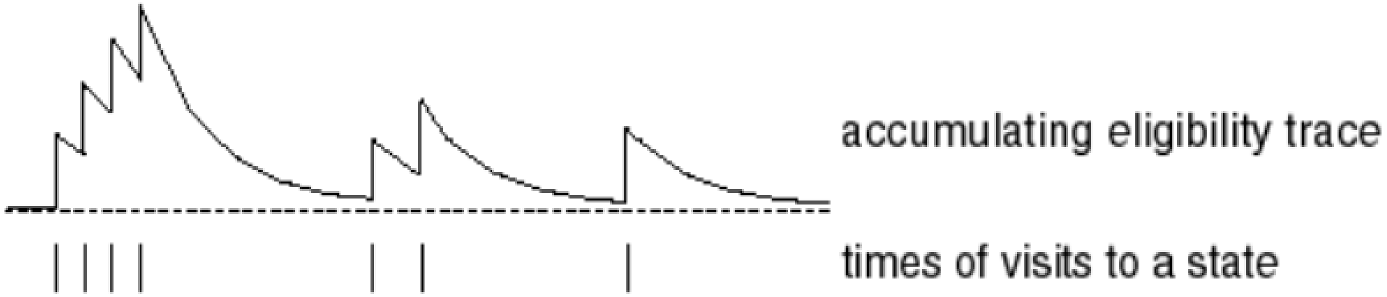
- Notes:
  - Consider backward (eligible traces) only when you try to implement it online. Otherwise, you can ignore it now.

Eligibility Traces



- Credit assignment problem: did bell or light cause shock?
- Frequency heuristic: assign credit to most frequent states
- Recency heuristic: assign credit to most recent states
- Eligibility traces combine both heuristics

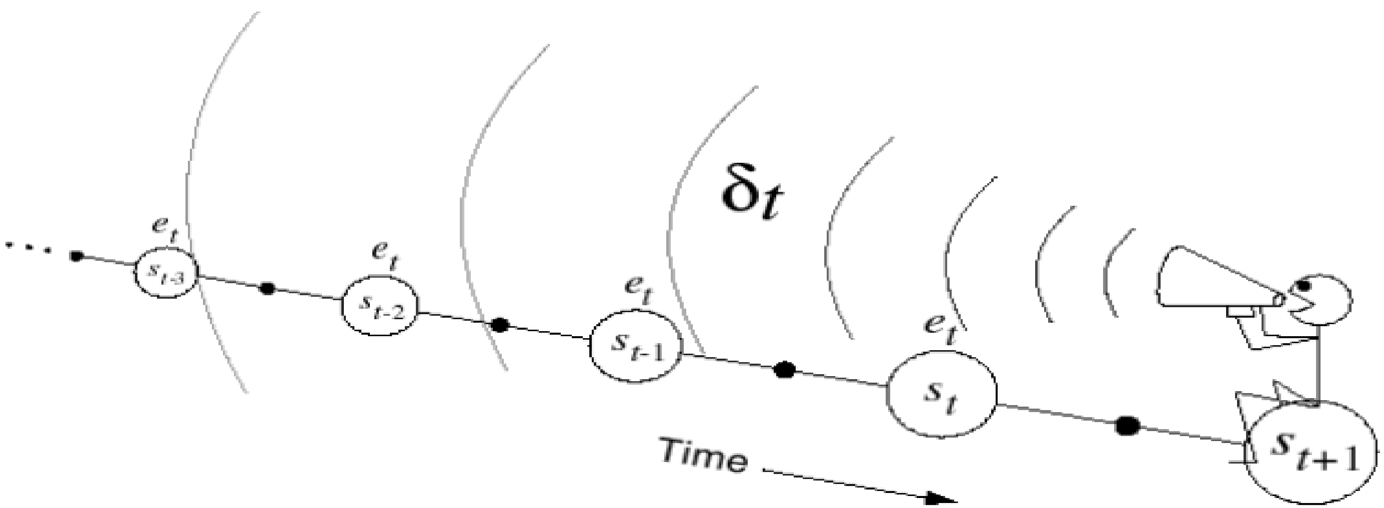
$$E_0(s) = 0$$
$$E_t(s) = \gamma\,\lambda E_{t-1}(s) + 1(S_t = s)$$



Backward View TD(λ)

- Keep an eligibility trace for every state *s*
- Update value $V(s)$ for every state *s*
- In proportion to TD-error δ_t and eligibility trace $E_t(s)$

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$$
$$V(s) \leftarrow V(s) + \alpha \delta_t E_t(s)$$



TD(λ) and TD(0)

- When $\lambda = 0$, only current state is updated

$$E_t(s) = 1(S_t = s)$$

$$V(s) \leftarrow V(s) + \alpha\delta_t E_t(s)$$

- This is exactly equivalent to TD(0) update

$$V(S_t) \leftarrow V(S_t) + \alpha\delta_t$$

TD(λ) and MC

- When $\lambda = 1$, TD(1) = MC

$$G_t^{\lambda} = (1-\lambda)\sum_{n=1}^{T-t-1} \lambda^{n-1} G_t^{(n)} + \lambda^{T-t-1} G_t = G_t$$

  – Consider episodic environments with offline updates
    - Over the course of an episode, total update for TD(1) is the same as total update for MC
  – Consider an episode where s is visited once at time-step k,
    - TD(1) updates accumulate error online

$$\sum_{t=1}^{T-1} \alpha\delta_t E_t(s) = \alpha \sum_{t=k}^{T-1} \gamma^{t-k}\delta_t = \alpha(G_k - V(S_k))$$

    - By end of episode it accumulates total error

$$\delta_k + \gamma\delta_{k+1} + \gamma^2\delta_{k+2} + \cdots + \gamma^{T-1-k}\delta_{T-1}$$

Outline

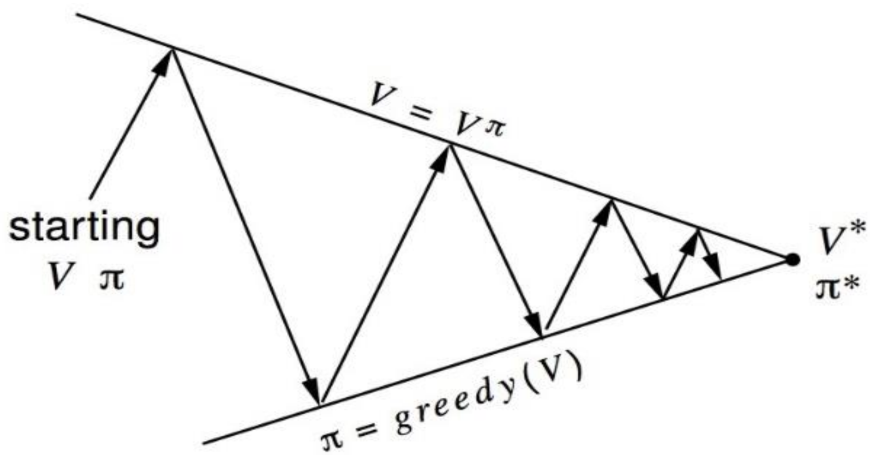
- Model-free prediction: Estimate the value function of an unknown MDP
  - Monte-Carlo (MC) Learning
  - Temporal Difference (TD) Learning
- Model-free control: Approximate optimal policies based on the estimation of the value function.
  - On-Policy Monte-Carlo Control
  - On-Policy Temporal-Difference Learning
  - Off-Policy Learning

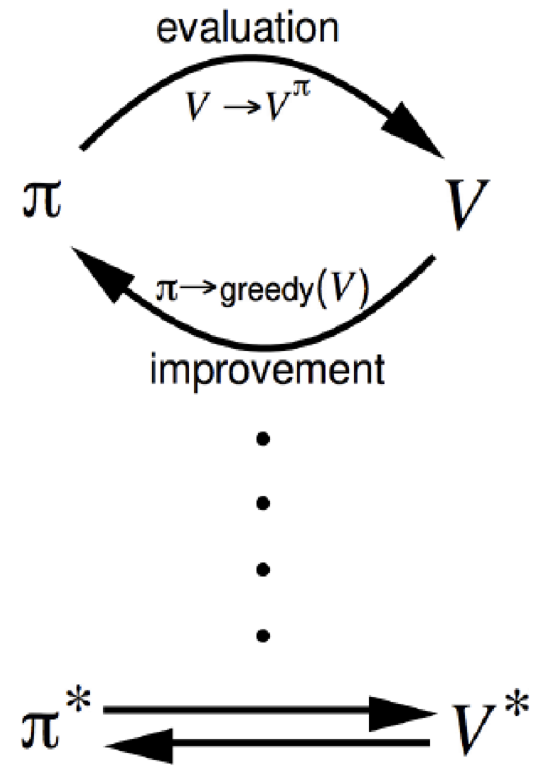
Model-Free Control

- For most of MDP problems, either:
  - MDP model is unknown, but experience can be sampled
  - MDP model is known, but is too big to use, except by samples
- Model-free control can solve these problems
- On-policy vs. off-policy
  - On-policy learning
    - "Learn on the job"
    - Learn about policy π from experience sampled from π
  - Off-policy learning
    - "Look over someone's shoulder"
    - Learn about policy π from experience sampled from μ

Generalized Policy Iteration (Review)

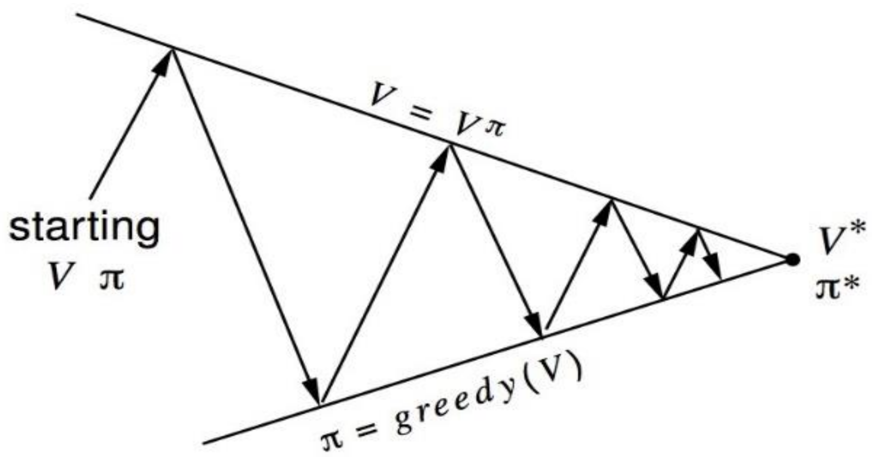
- Policy evaluation → Estimate v_π
  - e.g. Iterative policy evaluation
- Policy improvement → Generate $\pi' \geq \pi$
  - e.g. Greedy policy improvement





Generalized Policy Iteration With Monte-Carlo Evaluation

- Policy evaluation
  - Monte-Carlo policy evaluation, $V = v_\pi$? Problems: Model free?
- Policy improvement
  - Greedy policy improvement?
    - Problems: Always choose the same one (the best one)?



Model-Free Policy Iteration Using Action-Value Function

- Greedy policy improvement over $V(s)$ requires model of MDP

$$\pi'(s) = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \left(\mathcal{R}_s^a + \mathcal{P}_{ss'}^a V(s') \right)$$

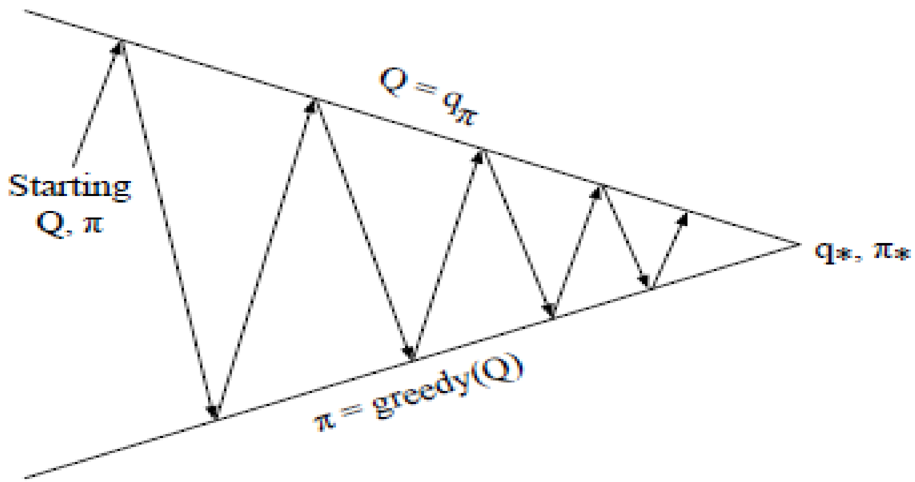
- Greedy policy improvement over $Q(s,a)$ is model-free

$$\pi'(s) = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\ Q(s,a)$$

- Note: this is very important!

Generalized Policy Iteration With Action-Value Function

- Policy evaluation
  - Monte-Carlo policy evaluation, $Q = q_\pi$
- Policy improvement
  - Greedy policy improvement?
    - Problems: Always choose the same one (the best one)?



Example of Greedy Action Selection

- There are two doors in front of you,
  Always apply the greedy action selection:
  - You open the left door and get reward 0
    $V(left) = 0$
  - You open the right door and get reward +1
    $V(right) = +1$
  - You open the right door and get reward +3
    $V(right) = +2$
  - You open the right door and get reward +2
    $V(right) = +2$
  - $\vdots$
- Are you sure you've chosen the best door?

ε-Greedy Exploration

- ε-greedy policy:

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

- Exploration
  – If you always try the best, you don't explore a real better one.
  – With probability ε choose an action at random
    - Simplest idea for ensuring continual exploration
  – All m actions are tried with non-zero probability
- Exploitation
  – If you always choose at random, you don't exploit the best
  – With probability $1 - \varepsilon$ choose the greedy action

ε-Greedy Policy Improvement

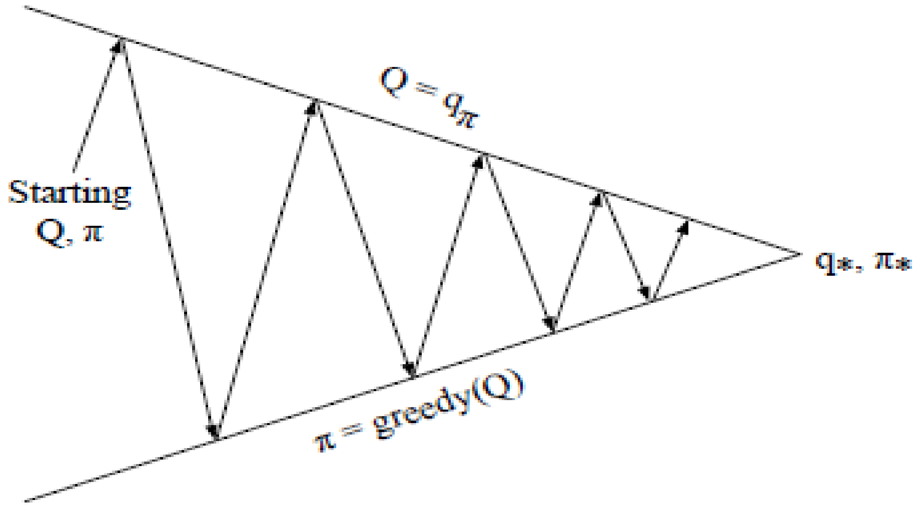
- Theorem
  - For any ε-greedy policy π, the ϵ-greedy policy π' with respect to q_π is an improvement, $v_{\pi'}(s) \geq v_\pi(s)$
- Proof:

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

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

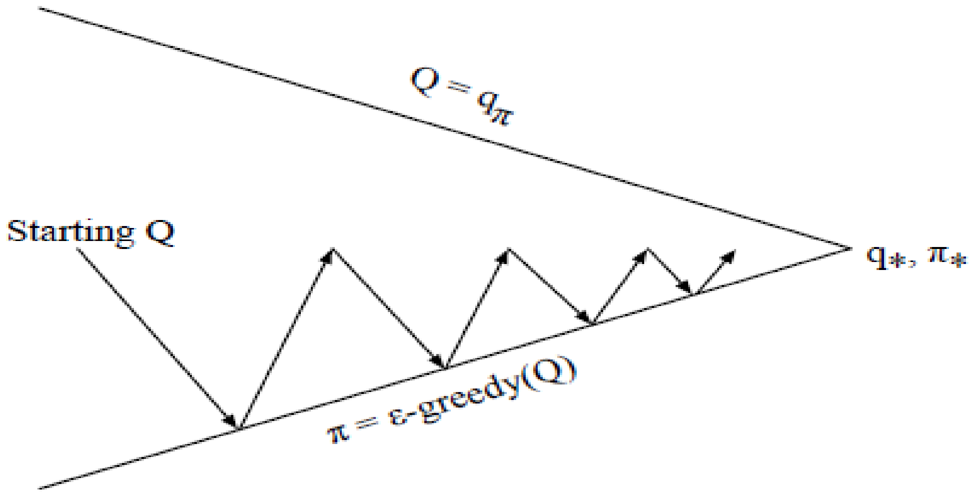
Monte-Carlo Policy Iteration

- Policy evaluation
  - Monte-Carlo policy evaluation, $Q = q_\pi$
- Policy improvement
  - ε-greedy policy improvement
    - Converge too, but the proof is not given here.



Monte-Carlo Control

- Policy evaluation
  - Monte-Carlo policy evaluation, $Q \approx q_\pi$
- Policy improvement
  - ε-greedy policy improvement
    - For every episode, improve more slowly by at most a factor of ε.



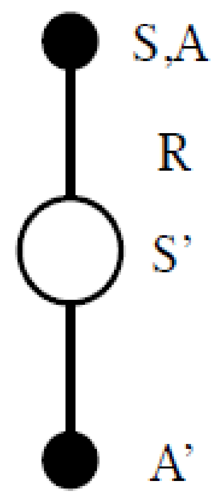
TD vs. MC Control

- Temporal-difference (TD) learning has several advantages over Monte-Carlo (MC)
  - Lower variance
  - Online
  - Incomplete sequences
- Natural idea: use TD instead of MC in our control loop
  - Apply TD to $Q(s, a)$
  - Use ε-greedy policy improvement
  - Update every time-step

Updating Action-Value Functions with Sarsa

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

- Notice: Interesting naming



Sarsa Algorithm for On-Policy Control

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Repeat (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

- Sarsa converges to the optimal action-value function
- n-step Sarsa – like n-step return
- Sarsa(λ) – like TD(λ)

Off-Policy Learning

- Evaluate target policy $\pi(a|s)$ to compute $V_\pi(s)$ or $q_\pi(s,a)$
- While following behaviour policy $\mu(a|s)$

$$\{S_1, A_1, R_2, \ldots, S_T\} \sim \mu$$

- Why is this important?
  - Learn from observing humans or other agents
  - Re-use experience generated from old policies $\pi_1, \pi_2, \ldots, \pi_{t-1}$
  - Learn about optimal policy while following exploratory policy
  - Learn about multiple policies while following one policy

Importance Sampling

- Estimate the expectation of a different distribution

$$\mathbb{E}_{X\sim P}[f(X)]$$
$$= \sum P(X) f(X)$$
$$= \sum Q(X) \frac{P(X)}{Q(X)} f(X)$$
$$= \mathbb{E}_{X\sim Q}\left[\frac{P(X)}{Q(X)} f(X)\right]$$

Importance Sampling for Off-Policy Monte-Carlo

- Use returns generated from μ to evaluate π
- Weight return G_t according to similarity between policies
- Multiply importance sampling corrections along whole episode

$$G_t^{\pi/\mu} = \frac{\pi(A_t|S_t)\pi(A_{t+1}|S_{t+1})}{\mu(A_t|S_t)\mu(A_{t+1}|S_{t+1})} \dots \frac{\pi(A_T|S_T)}{\mu(A_T|S_T)} G_t$$

- Update value towards corrected return

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

- Cannot use if μ is zero when π is non-zero
- Importance sampling can dramatically increase variance

Importance Sampling for Off-Policy TD

- Use TD targets generated from μ to evaluate π
- Weight TD target $R + \gamma V(S')$ by importance sampling
- Only need a single importance sampling correction

$$V(S_t) \leftarrow V(S_t) + \alpha\left(\frac{\pi(A_t|S_t)}{\mu(A_t|S_t)}(R_{t+1}+\gamma V(S_{t+1})) - V(S_t)\right)$$

- Much lower variance than Monte-Carlo importance sampling (since just one step)
- Policies only need to be similar over a single step

Q-Learning

- We now consider off-policy learning of action-values $Q(s, a)$
- No importance sampling is required
- Next action is chosen using behaviour policy

  $$A_{t+1} \sim \mu(\cdot \mid S_{t+1})$$
- But we consider alternative successor action

  $$A' \sim \pi(\cdot \mid S_{t+1})$$
- And update $Q(S_t, A_t)$ towards value of alternative action

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

Off-Policy Control with Q-Learning

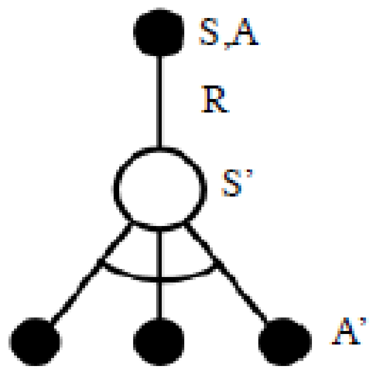
- We now allow both behaviour and target policies to improve
- The target policy π is greedy w.r.t. $Q(s,a)$

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

- The behaviour policy μ is e.g. ε-greedy w.r.t. $Q(s,a)$
- The Q-learning target then simplifies:

$$\begin{aligned} & R_{t+1} + \gamma Q(S_{t+1}, A') \\ & = R_{t+1} + \gamma Q\left(S_{t+1}, \underset{a'}{\operatorname{argmax}}\; Q(S_{t+1}, a')\right) \\ & = R_{t+1} + \max_{a'} \gamma Q(S_{t+1}, a') \end{aligned}$$

Q-Learning Control Algorithm



- $Q(S,A) \leftarrow Q(S,A) + \alpha\left(R + \gamma \max_{a'} Q(S',a') - Q(S,A)\right)$
- Theorem
  - Q-learning control converges to the optimal action-value function, $Q(s,a) \rightarrow q_*(s,a)$

Q-Learning Algorithm for Off-Policy Control

Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
    Initialize S
    Repeat (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