Lecture 4: Model-Free Prediction and Control

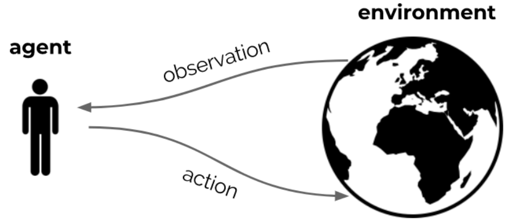
Hado van Hasselt

February 1, 2018, UCL

Background

Sutton & Barto 2018, Chapters 5 + 6

Recap



- Reinforcement learning is the science of learning to make decisions
- Agents can learn a policy, value function and/or a model
- The general problem involves taking into account time and consequences
- Decisions affect the reward, the agent state, and environment state

Sample-based reinforcement learning

- Last lecture:
  - Planning by dynamic programming to solve a known MDP
- This lecture:
  - Model-free prediction to estimate values in an unknown MDP
  - Model-free control to optimise values in an unknown MDP
- Not yet:
  - Learning policies directly in sequential problems (policy gradients)
  - Continuous MDPs
  - Deep reinforcement learning

Sample-based reinforcement learning

- We can use experience samples to learn without a model
- We call direct sampling of episodes Monte Carlo
- MC is model-free: no knowledge of MDP required, only samples

Sample-based reinforcement learning

- Simple example, multi-armed bandit:
  - For each action, average reward samples

$$q_t(a) \;=\; \frac{\sum_{i=0}^{t} \mathcal{I}(A_i = a) R_{i+1}}{\sum_{i=0}^{t} \mathcal{I}(A_i = a)} \quad \approx \quad \mathbb{E}\left[R_{t+1} | A_t = a\right] \;=\; q(a)$$

  - Equivalently:

$$\begin{aligned} q_t(A_t) &= q_{t-1}(A_t) + \alpha_t (R_t - q_{t-1}(A_t)) \\ q_t(a) &= q_{t-1}(a) & \forall a \neq A_t \end{aligned}$$

with $\alpha_t = \frac{1}{N_t(A_t)} = \frac{1}{\sum_{i=0}^{t} \mathcal{I}(A_i = a)}$

Note: we changed notation from $A_t \rightarrow R_t$ to $A_t \rightarrow R_{t+1}$
In MDPs, the reward is said to arrive on the time step after the action

Contextual bandits

- Consider bandits with different states ('context')
  - episodes still end after one step
  - actions do not affect the states
  - e.g., different visitors to a website
- Then, we want to estimate

$$q(s, a) = \mathbb{E}\left[R_{t+1} | S_t = s, A_t = a\right]$$

- q could be a parametric function, e.g., neural network, and we could use loss

$$l_t(\theta) = \frac{1}{2}(q_\theta(S_t, A_t) - R_{t+1})^2$$

- Also works for large (continuous) state spaces $\mathcal{S}$ — this is just regression

Monte-Carlo Policy Evaluation

- Now consider sequential decision problems
- Goal: learn v_π from episodes of experience under policy π

$$S_1, A_1, R_2, ..., S_k \sim \pi$$

- The return is the total discounted reward (for an episode ending at time $T > t$):

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

- The value function is the expected return:

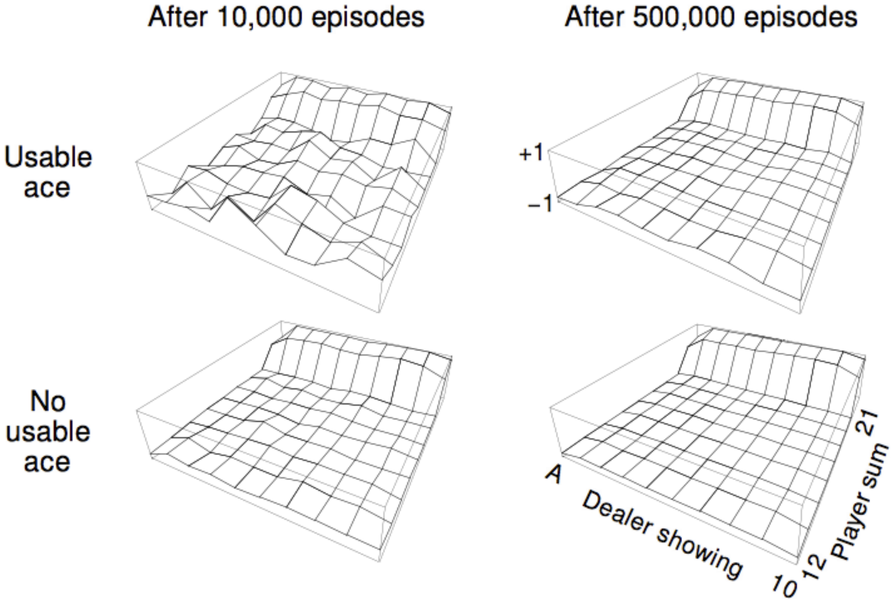
$$v_\pi(s) = \mathbb{E}\left[G_t \mid S_t = s, \pi\right]$$

- We can just use sample average return instead of expected return
- We call this Monte Carlo policy evaluation

Blackjack Example

- States (200 of them):
  - Current sum (12-21)
  - Dealer's showing card (ace-10)
  - Do I have a "useable" ace? (yes-no)
- Action stick: Stop receiving cards (and terminate)
- Action draw: Take another card (random, no replacement)
- Reward for stick:
  - $+1$ if sum of cards $>$ sum of dealer cards
  - 0 if sum of cards $=$ sum of dealer cards
  - -1 if sum of cards $<$ sum of dealer cards
- Reward for draw:
  - -1 if sum of cards > 21 (and terminate)
  - 0 otherwise
- Transitions: automatically draw if sum of cards < 12

Blackjack Value Function after Monte-Carlo Learning



Temporal Difference Learning by Sampling Bellman Equations

- Previous lecture: Bellman equations,

$$v_\pi(s) = \mathbb{E}\left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t \sim \pi(S_t)\right]$$

- Previous lecture: Approximate by iterating,

$$v_{k+1}(s) = \mathbb{E}\left[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s, A_t \sim \pi(S_t)\right]$$

- We can sample this!

$$v_{t+1}(S_t) = R_{t+1} + \gamma v_t(S_{t+1})$$

- This is likely quite noisy — better to average:

$$v_{t+1}(S_t) = v_t(S_t) + \alpha_t \left(\underbrace{R_{t+1} + \gamma v_t(S_{t+1})}_{\text{target}} - v_t(S_t) \right)$$

Temporal difference learning

- In (approximate) DP: we use one step of the model and bootstrap

$$\text{target} \quad = \quad \mathbb{E}\left[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s, \pi\right]$$

- In Monte Carlo, we sample, and use:

$$\text{target} \quad = \quad G_t \quad (= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots + R_T)$$

- Alternatively, we could sample **and** bootstrap, and use

$$\text{target} \quad = \quad R_{t+1} + \gamma v_t(S_{t+1}) \, .$$

- This is called temporal-difference learning

Temporal-Difference Learning

- TD is model-free (no knowledge of MDP) and learn directly from experience
- TD can learn from incomplete episodes, by bootstrapping
- TD can learn during each episode

MC and TD

- Goal: learn v_π online from experience under policy π
- Incremental Monte-Carlo
  - Update value $v(S_t)$ towards sampled return G_t

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

- Simplest temporal-difference learning algorithm: TD(0)
  - Update value $v(S_t)$ towards estimated return $R_{t+1} + \gamma v(S_{t+1})$

$$v(S_t) \leftarrow v(S_t) + \alpha\left(\overbrace{\underbrace{R_{t+1} + \gamma v(S_{t+1})}_{\text{target}} - v(S_t)}^{\text{TD error}}\right)$$

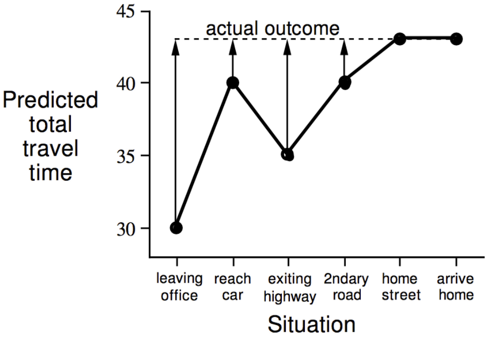
  - $\delta_t = R_{t+1} + \gamma v(S_{t+1}) - v(S_t)$ is called the TD error

Driving Home Example

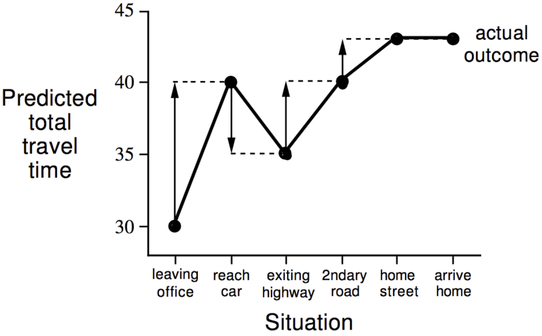
State	Elapsed Time (minutes)	Predicted Time to Go	Predicted Total Time
leaving office	0	30	30
reach car, raining	5	35	40
exit highway	20	15	35
behind truck	30	10	40
home street	40	3	43
arrive home	43	0	43

Driving Home Example: MC vs. TD

Changes recommended by Monte Carlo methods (α=1)



Changes recommended by TD methods (α=1)



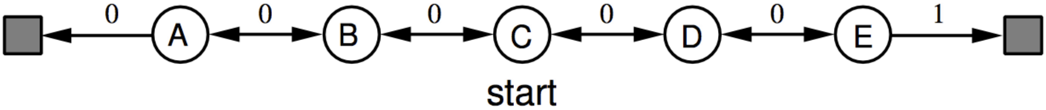
Advantages and Disadvantages of MC vs. TD

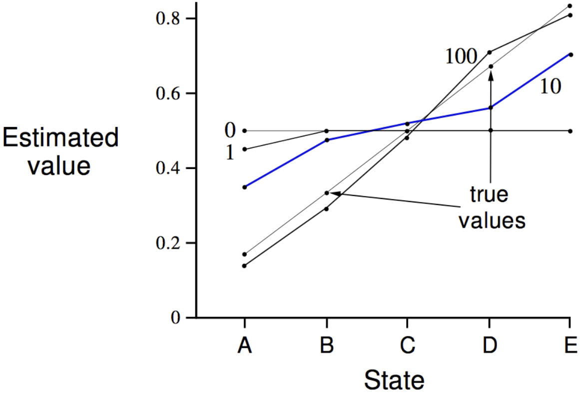
- TD can learn before knowing the final outcome
  - TD can learn online after every step
  - MC must wait until end of episode before return is known
- TD can learn without the final outcome
  - TD can learn from incomplete sequences
  - MC can only learn from complete sequences
  - TD works in continuing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

Bias/Variance Trade-Off

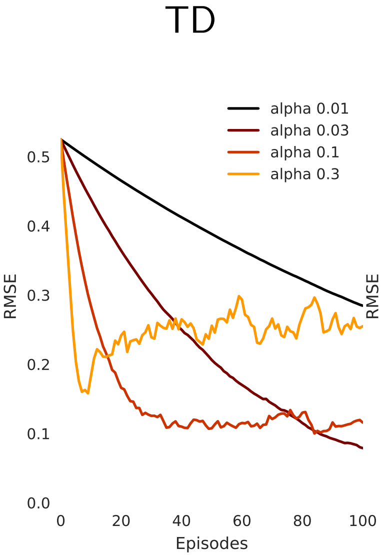
- Return $G_t = R_{t+1} + \gamma R_{t+2} + \ldots$ is an unbiased estimate of $v_\pi(S_t)$
- TD target $R_{t+1} + \gamma v(S_{t+1})$ is a biased estimate of $v_\pi(S_t)$
  - Unless $v(S_{t+1}) = v_\pi(S_{t+1})$
- But the TD target has much lower variance:
  - Return depends on many random actions, transitions, rewards
  - TD target depends on one random action, transition, reward

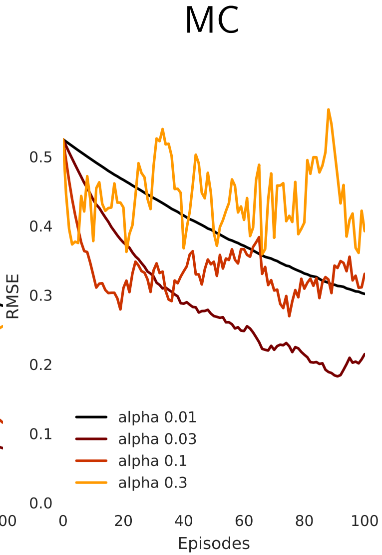
Random Walk Example





Random Walk: MC vs. TD





Batch MC and TD

- Tabular MC and TD converge: $v \to v_\pi$ as experience $\to \infty$ and $\alpha \to 0$
- But what about finite experience?

$$\begin{array}{ll} \text{episode 1:} & S_1^1, A_1^1, R_2^1, ..., S_{T_1}^1 \\ & \vdots \\ \text{episode K:} & S_1^K, A_1^K, R_2^K, ..., S_{T_K}^K \end{array}$$

  - Repeatedly sample each episodes $k \in [1, K]$ and apply MC or TD(0)
  - $=$ sampling from an empirical model

AB Example

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

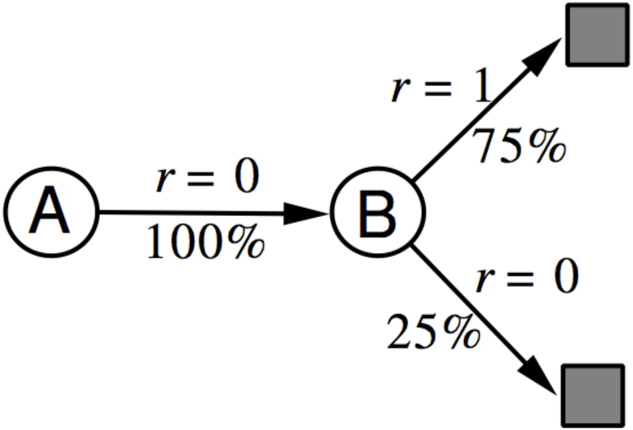
$$\mathrm{A}, 0, \mathrm{B}, 0$$
$$\mathrm{B}, 1$$
$$\mathrm{B}, 1$$
$$\mathrm{B}, 1$$
$$\mathrm{B}, 1$$
$$\mathrm{B}, 1$$
$$\mathrm{B}, 1$$
$$\mathrm{B}, 0$$

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

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 $v(A), v(B)$?

Certainty Equivalence

- MC converges to best mean-squared fit for the observed returns

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

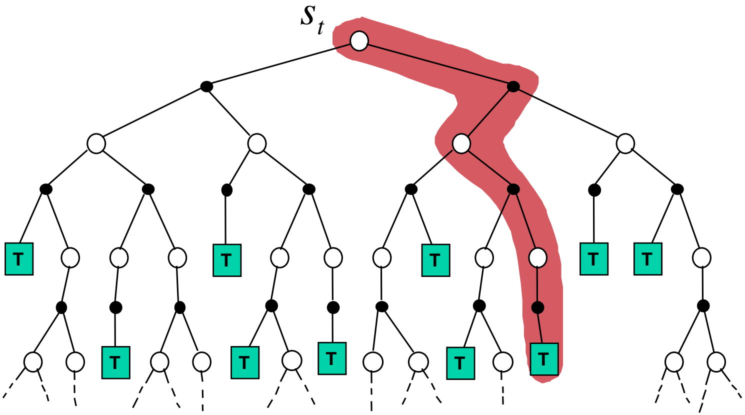
- In the AB example, $v(A) = 0$
- TD converges to solution of max likelihood Markov model
  - Solution to the empirical MDP $\langle \mathcal{S}, \mathcal{A}, \hat{p}, \gamma \rangle$ that best fits the data
  - In the AB example: $\hat{p}(S_{t+1} = B \mid S_t = A) = 1$, and therefore $v(A) = v(B) = 0.75$

Advantages and Disadvantages of MC vs. TD (3)

- TD exploits Markov property
  - Usually more efficient in Markov environments
- MC does not exploit Markov property
  - Usually more accurate in non-Markov environments

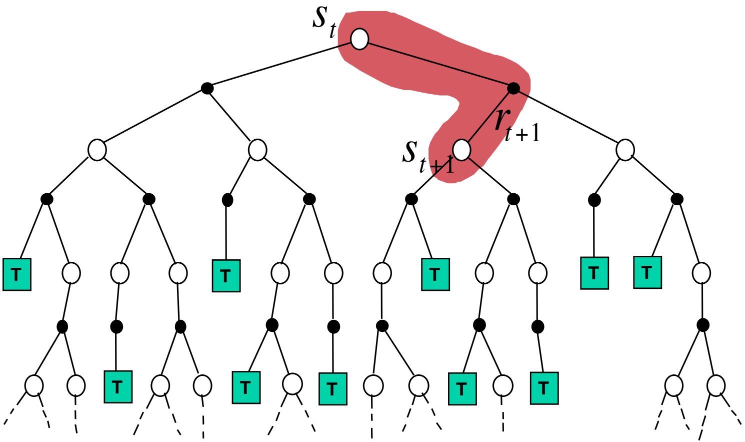
Monte-Carlo Backup

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



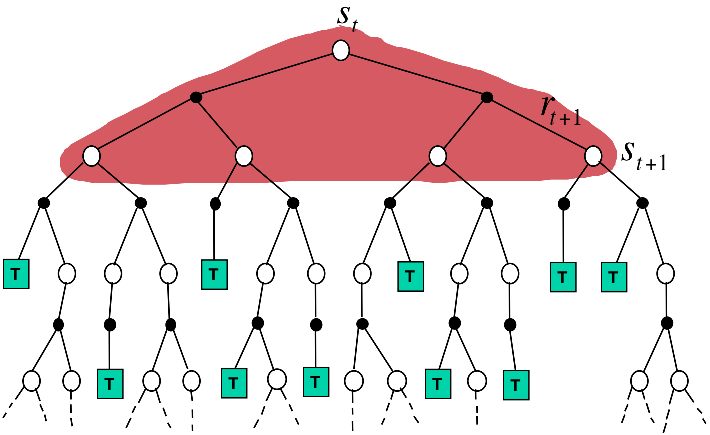
Temporal-Difference Backup

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



Dynamic Programming Backup

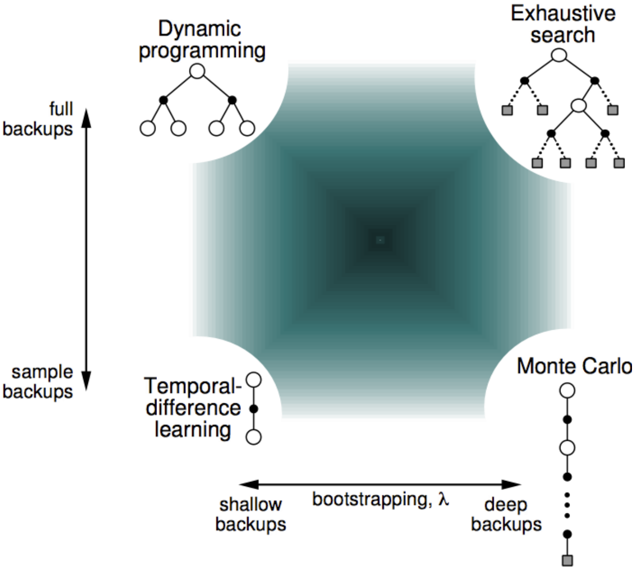
$$v(S_t) \leftarrow \mathbb{E}\left[R_{t+1} + \gamma v(S_{t+1}) \mid A_t \sim \pi(S_t)\right]$$



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

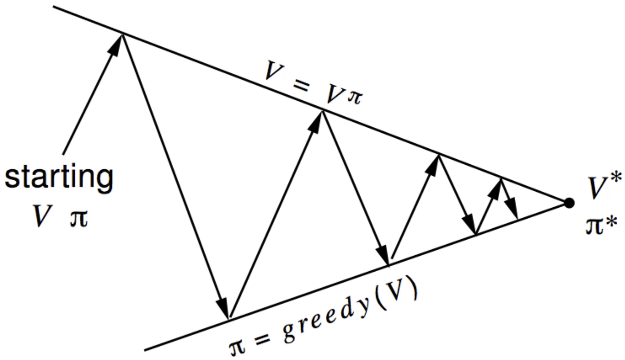
Unified View of Reinforcement Learning

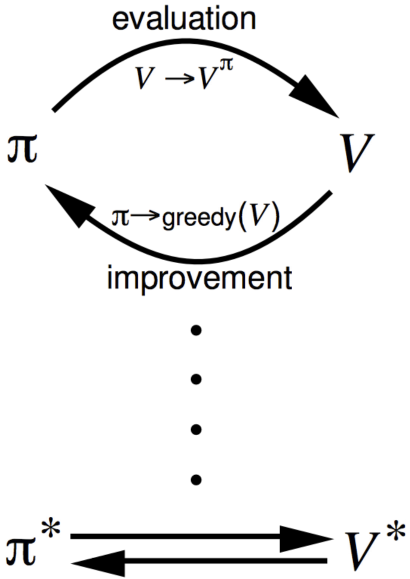


Model-Free Control

- Previous: Model-free prediction:
  Estimate the value function of an unknown MDP
- Next: Model-free control:
  Optimise the value function of an unknown MDP

Generalized Policy Iteration (Refresher)





Policy evaluation Estimate v_π

e.g. Iterative policy evaluation

Policy improvement Generate $\pi' \geq \pi$

e.g. Greedy policy improvement

Monte Carlo

- Recall, Monte Carlo estimate from state S_t is

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots$$

- $\mathbb{E}[G_t] = v_\pi$
- So, we can average multiple estimates to get v_π

Model-Free Policy Iteration Using Action-Value Function

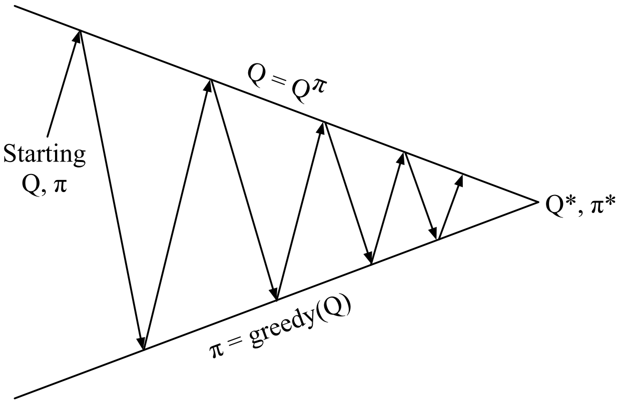
- But, greedy policy improvement over $v(s)$ requires model of MDP

$$\pi'(s) = \underset{a}{\operatorname{argmax}}\, \mathbb{E}\left[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s, A_t = a\right]$$

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

$$\pi'(s) = \underset{a}{\operatorname{argmax}}\, q(s, a)$$

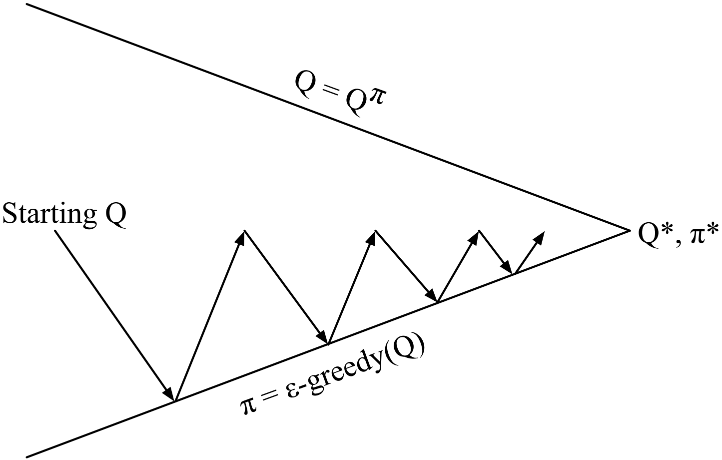
Generalised Policy Iteration with Action-Value Function



Policy evaluation Monte-Carlo policy evaluation, $q \approx q_{\pi}$

Policy improvement Greedy policy improvement? No exploration!

Monte-Carlo Generalized Policy Iteration



Every episode:

Policy evaluation Monte-Carlo policy evaluation, $q \approx q_\pi$

Policy improvement ϵ-greedy policy improvement

GLIE

Definition

Greedy in the Limit with Infinite Exploration (GLIE)

- All state-action pairs are explored infinitely many times,

$$\lim_{k \to \infty} N_k(s, a) = \infty$$

- The policy converges to a greedy policy,

$$\lim_{k \to \infty} \pi_k(s, a) = \mathcal{I}(a = \operatorname*{argmax}_{a'} q_k(s, a'))$$

- For example, ϵ-greedy with $\epsilon_k = \frac{1}{k}$

GLIE Monte-Carlo Control

- Sample kth episode using π: $\{S_1, A_1, R_2, ..., S_T\} \sim \pi$
- For each state S_t and action A_t in the episode,

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

- Improve policy based on new action-value function

$$\epsilon \leftarrow 1/k$$
$$\pi \leftarrow \epsilon\text{-greedy}(q)$$

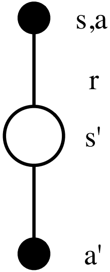
Theorem

GLIE Monte-Carlo control converges to the optimal action-value function, $q(s, a) \rightarrow q_*(s, a)$

MC vs. TD Control

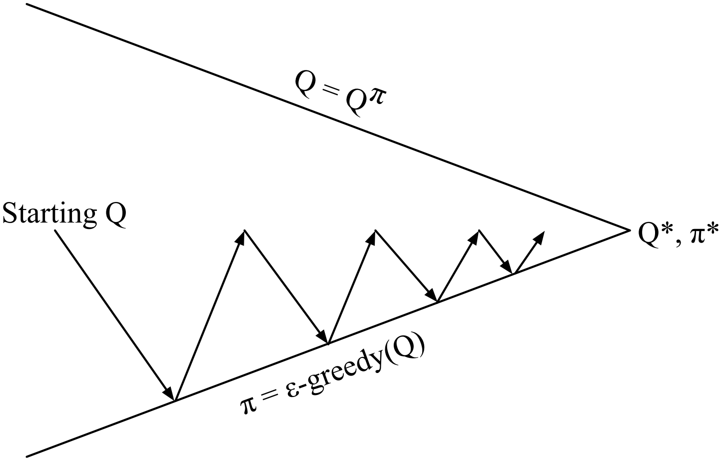
- Temporal-difference (TD) learning has several advantages over Monte-Carlo (MC)
  - Lower variance
  - Online
  - Can learn from incomplete sequences
- Natural idea: use TD instead of MC for control
  - Apply TD to $q(s, a)$
  - Use, e.g., ϵ-greedy policy improvement
  - Update every time-step

Updating Action-Value Functions with Sarsa



$$q(s,a) \leftarrow q(s,a) + \alpha \left(r + \gamma q(s',a') - q(s,a) \right)$$

Sarsa



Every time-step:

Policy evaluation Sarsa, $q \approx q_\pi$

Policy improvement ϵ-greedy policy improvement

Tabular Sarsa

Initialize $Q(s, a)$ arbitrarily
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)
    $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$
    $s \leftarrow s'$; $a \leftarrow a'$;
  until s is terminal

On and Off-Policy Learning

- 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 b

Dynamic programming

- We discussed other algorithms:

$$
\begin{aligned}
v_{k+1}(s) &= \mathbb{E}\left[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s, A_t \sim \pi(S_t)\right] && \text{(policy evaluation)} \\
v_{k+1}(s) &= \max_a \mathbb{E}\left[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s, A_t = a\right] && \text{(value iteration)} \\
q_{k+1}(s,a) &= \mathbb{E}\left[R_{t+1} + \gamma q_k(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a\right] && \text{(policy evaluation)} \\
q_{k+1}(s,a) &= \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q_k(S_{t+1}, a') \mid S_t = s, A_t = a\right] && \text{(value iteration)}
\end{aligned}
$$

TD learning

- Analogous TD algorithms

$$v_{t+1}(S_t) = v_t(S_t) + \alpha_t \left(R_{t+1} + \gamma v_t(S_{t+1}) - v_t(S_t)\right) \qquad \text{(TD)}$$

$$q_{t+1}(s,a) = q_t(S_t, A_t) + \alpha_t \left(R_{t+1} + \gamma q_t(S_{t+1}, A_{t+1}) - q_t(S_t, A_t)\right) \qquad \text{(Sarsa)}$$

$$q_{t+1}(s,a) = q_t(S_t, A_t) + \alpha_t \left(R_{t+1} + \gamma \max_{a'} q_t(S_{t+1}, a') - q_t(S_t, A_t)\right) \qquad \text{(Q-learning)}$$

- Note, no trivial analogous version of value iteration

$$v_{k+1}(s) = \max_a \mathbb{E}\left[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s, A_t = a\right]$$

Can you explain why? We do not have a model and can not sample.

Off-Policy Learning

- Evaluate target policy $\pi(s,a)$ to compute $v_\pi(s)$ or $q_\pi(s,a)$
- While following behaviour policy $b(s,a)$

$$\{S_1, A_1, R_2, ..., S_T\} \sim b$$

- Why is this important?
  - Learn from observing humans or other agents
  - Re-use experience from old policies
  - Learn about optimal policy while following exploratory policy
  - Learn about multiple policies while following one policy
- Q-learning estimates the value of the greedy policy

$$q_{t+1}(s,a) = q_t(S_t, A_t) + \alpha_t \left(R_{t+1} + \gamma \max_{a'} q_t(S_{t+1}, a') - q_t(S_t, A_t) \right)$$

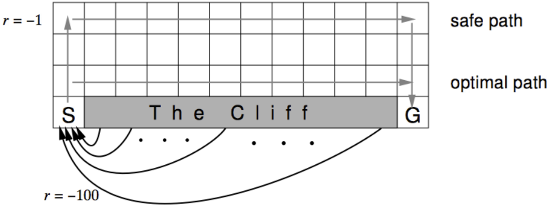
Q-Learning Control Algorithm

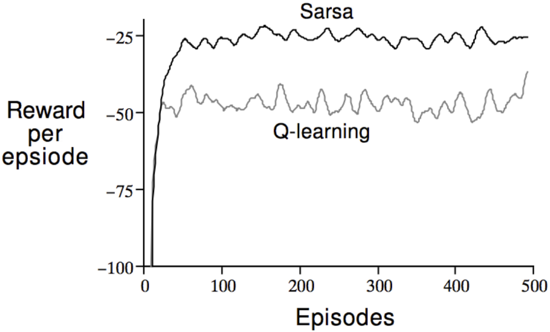
Theorem

Q-learning control converges to the optimal action-value function, $q \to q^*$, as long as we take each action in each state infinitely often.

Note: no need for greedy behaviour!

Cliff Walking Example





Q-learning overestimation

- Classical Q-learning has potential issues
- Recall

$$\max_a q_t(S_{t+1}, a) = q_t(S_{t+1}, \operatorname*{argmax}_a q_t(S_{t+1}, a))$$

- Uses same values to select and to evaluate
- ... but values are approximate
  - more likely to select overestimated values
  - less likely to select underestimated values
- This causes upward bias

Double Q-learning

- Q-learning uses same values to select and to evaluate

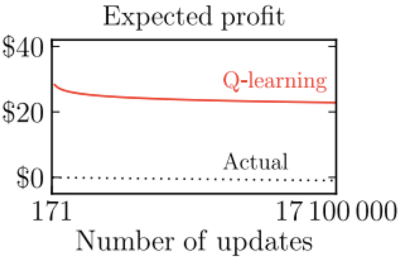
$$R_{t+1} + \gamma q_t(S_{t+1}, \underset{a}{\operatorname{argmax}}\ q_t(S_{t+1}, a))$$

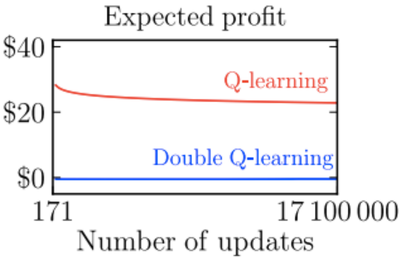
- Solution: decouple selection from evaluation, using the
- Double Q-learning:
  - Store two q functions: q, q'

$$R_{t+1} + \gamma q'_t(S_{t+1}, \underset{a}{\operatorname{argmax}}\ q_t(S_{t+1}, a)) \tag{1}$$
$$R_{t+1} + \gamma q_t(S_{t+1}, \underset{a}{\operatorname{argmax}}\ q'_t(S_{t+1}, a)) \tag{2}$$

  - Each step, pick one (e.g., randomly) and update, using update (1) for q or (2) for q'
  - Can use both to act (e.g., use $q + q'$)

Roulette example

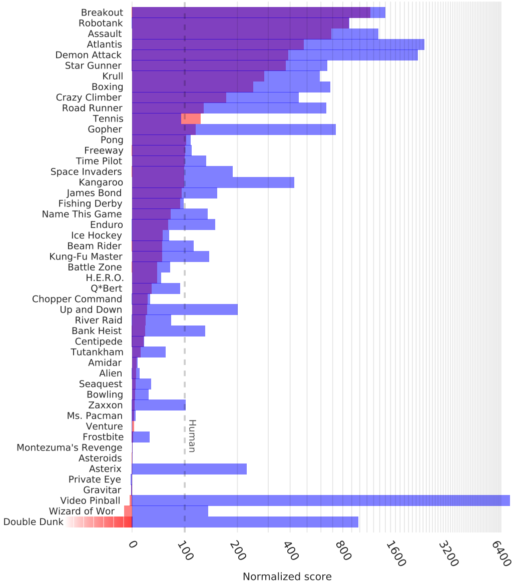




Double DQN on Atari

DQN

Double DQN



Importance Sampling

- Estimate the expectation of a different distribution

$$
\begin{aligned}
\mathbb{E}_{x\sim d}[f(x)] &= \sum d(x)f(x) \\
&= \sum d'(x)\frac{d(x)}{d'(x)}f(x) \\
&= \mathbb{E}_{x\sim d'}\left[\frac{d(x)}{d'(x)}f(x)\right]
\end{aligned}
$$

Importance Sampling

- Estimate the expectation of a different distribution

$$\begin{aligned}\mathbb{E}\left[R_{t+1} \mid S_t = s, A_t \sim \pi\right] &= \sum_a \pi(a|s) r(s,a) \\ &= \sum b(a|s) \frac{\pi(a|s)}{b(a|s)} r(s,a) \\ &= \mathbb{E}\left[\frac{\pi(A_t|S_t)}{b(A_t|S_t)} R_{t+1} \mid S_t = s, A_t \sim b\right]\end{aligned}$$

- Ergo, when following b, can use $\frac{\pi(A_t|S_t)}{b(A_t|S_t)} R_{t+1}$ as unbiased sample

Importance Sampling for Off-Policy Monte-Carlo

- Use returns generated from b to evaluate π
- Weight return G_t according to similarity between policies

$$G_t^{\pi/b} = \frac{\pi(S_t, A_t)}{b(S_t, A_t)} \frac{\pi(S_{t+1}, A_{t+1})}{b(S_{t+1}, A_{t+1})} \cdots \frac{\pi(S_{T-1}, A_{T-1})}{b(S_{T-1}, A_{T-1})} G_t$$

- Update towards corrected return

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

- $\mathbb{E}\left[G_t^{\pi/b} \mid S_t = s, b \right] = v_\pi(s)$ — no bias!
- ...but importance sampling can dramatically increase variance...

Importance Sampling for Off-Policy TD Updates

- Use TD targets generated from b to evaluate π
- Weight TD target $r + \gamma v(s')$ by importance sampling
- Only need a single importance sampling correction

$$v(S_t) \leftarrow v(S_t) \ + \alpha \left(\frac{\pi(S_t, A_t)}{b(S_t, A_t)} \left(R_{t+1} + \gamma v(S_{t+1})\right) - v(S_t) \right)$$

- Much lower variance than Monte-Carlo importance sampling
- 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 may be chosen using behaviour policy $A_{t+1} \sim b(S_{t+1}, \cdot)$
- But we consider probabilities under $\pi(S_t, \cdot)$
- Update $q(S_t, A_t)$ towards value of alternative action

$$q(S_t, A_t) \leftarrow q(S_t, A_t) + \alpha \left(R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) q(S_{t+1}, a) - q(S_t, A_t) \right)$$

- Called Expected Sarsa (when $b = \pi$) or General Q-learning

Off-Policy Control with Q-Learning

- We want behaviour and target policies to improve
- E.g., 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 b is e.g. ϵ-greedy w.r.t. $q(s, a)$
- The Q-learning target then simplifies:

$$\begin{aligned} & R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) q(S_{t+1}, a) \\ = \; & R_{t+1} + \gamma \max_a q(S_{t+1}, a) \end{aligned}$$

Questions?

The only stupid question is the one you were afraid to ask but never did.
-Rich Sutton

For questions that arise outside of class, please use Moodle