535514: Reinforcement Learning

Lecture 19 — Value-Based Methods and Stochastic Approximation

Ping-Chun Hsieh

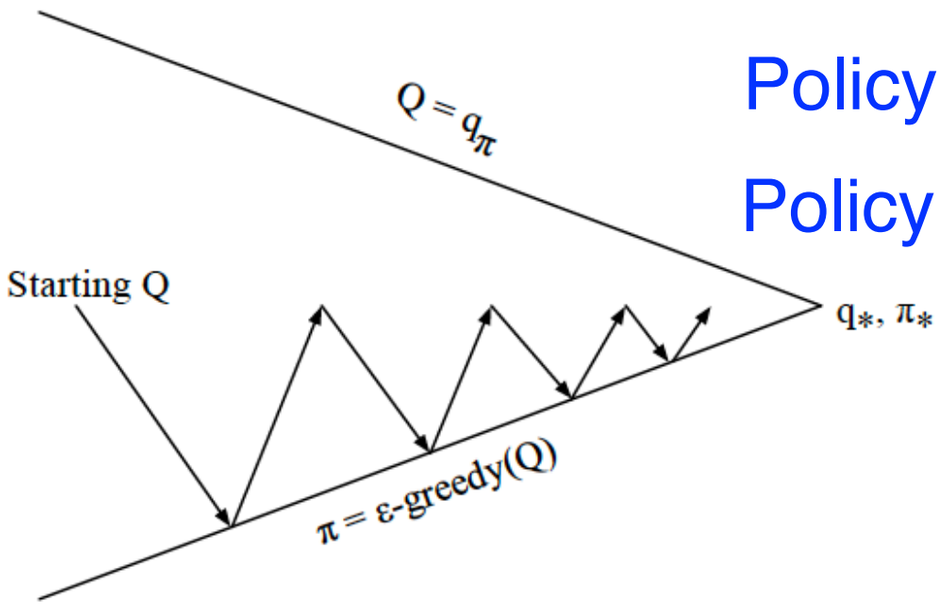
April 29, 2024

On-Policy vs Off-Policy Methods

	Policy Optimization	Value-Based	Model-Based	Imitation-Based
On-Policy	Exact PG REINFORCE (w/i baseline) A2C On-policy DAC TRPO Natural PG (NPG) PPO-KL & PPO-Clip RLHF by PPO-KL	Epsilon-Greedy MC Sarsa Expected Sarsa	Model-Predictive Control (MPC) PETS	IRL GAIL IQ-Learn
Off-Policy	Off-policy DPG & DDPG Twin Delayed DDPG (TD3)	Q-learning Double Q-learning DQN & DDQN C51 / QR-DQN / IQN Soft Actor-Critic (SAC)		

Review: ε-Greedy + MC / TD

- Idea: MC / TD + one-step ε-greedy policy improvement



Policy Evaluation: MC or TD policy evaluation

Policy Improvement: ε-Greedy policy improvement

- With probability $1 - \varepsilon$: choose the greedy action
- With probability ε: choose an action uniformly at random

$$\pi(a \mid s) = \begin{cases} \frac{\varepsilon}{|\mathcal{A}|} + 1 - \varepsilon, & \text{if } a = \arg\max_{a \in \mathcal{A}} Q(s, a) \\ \frac{\varepsilon}{|\mathcal{A}|}, & \text{otherwise} \end{cases}$$

(Slide Credit: David Silver)

ε-Greedy Monte-Carlo Control (Formally)

- ε-Greedy MC Control:

Step 1: Initialize $Q(s,a)$ and $N(s,a) = 0$ for each (s,a)

Step 2: In episode k, sample a trajectory $\tau = (s_0, a_0, r_1, \cdots) \sim \pi_k$

For each first-visit (s_t, a_t) in the current episode, update

$$N(s_t, a_t) \leftarrow N(s_t, a_t) + 1$$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \frac{1}{N(s_t, a_t)}(G_t - Q(s_t, a_t))$$

Improve the policy by using ε-greedy w.r.t. $Q(\,\cdot\,,\cdot\,)$

Question: Is ε-Greedy Monte-Carlo control *on-policy* or *off-policy*?

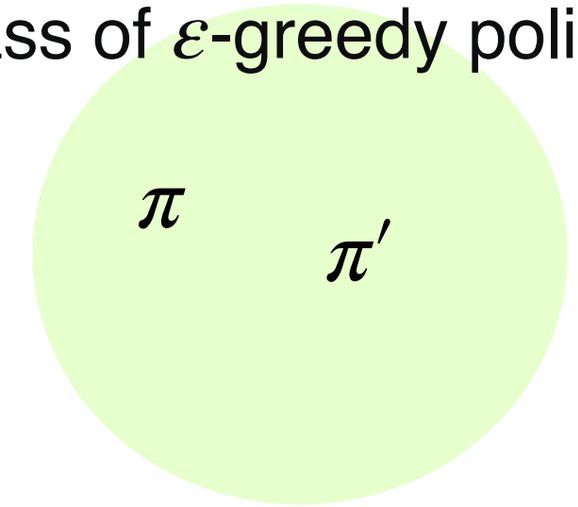
ε-Greedy Policy Improvement Theorem

- **ε-Greedy Policy Improvement Theorem**:

  For any ε-greedy policy π, the new ε-greedy policy π' with respect to Q^{π} is an improvement, i.e.

$$V^{\pi'}(s) \geq V^{\pi}(s), \quad \forall s \in S$$

- Proof: The proof is very similar to one-step policy improvement in Lecture 3 (see the appendix)



Sarsa: ε-Greedy TD Control (Formally)

- Sarsa: ε-Greedy TD control

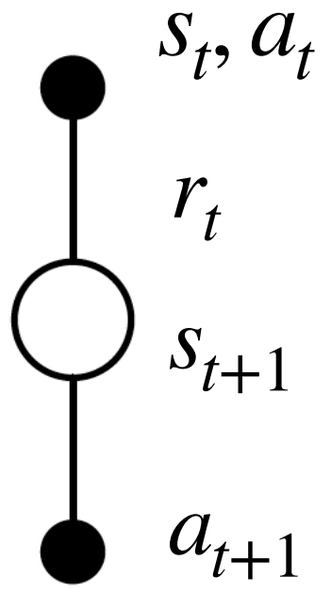
Step 1: Initialize $Q(s, a)$

Step 2: In each step t, repeat the following:

In state s_t, apply $a_t \sim \varepsilon$-greedy$(Q(s_t, \cdot))$

Observe s_{t+1} and draw $a_{t+1} \sim \varepsilon$-greedy$(Q(s_{t+1}, \cdot))$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t(s_t, a_t)(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$$



Question: Why is it called “Sarsa”?

Question: Is “Sarsa” *on-policy* or *off-policy*?

Rummery and Niranjan, “Online Q-Learning Using Connectionist Systems”, 1994

- Question: Does Sarsa enjoy convergence to the optimal Q?

Yes, under proper conditions

- **Greedy in the Limit with Infinite Exploration (GLIE)**:

(G1) All state-action pairs are explored **infinitely many** times

(G2) The policy converges on a greedy policy (no exploration in the limit)

- **Theorem**: Sarsa converges to the optimal action-value function, i.e., $Q(s,a) \rightarrow Q^*(s,a)$, under the following conditions:

(1) GLIE (2) $\sum_{t=1}^{\infty} \alpha_t(s,a) = \infty$, $\sum_{t=1}^{\infty} \alpha_t(s,a)^2 < \infty$, for all (s,a)

- Question: How to choose $\alpha_t(s,a)$?

- Proof Technique: *Stochastic approximation*

Singh et al., Convergence Results for Single-Step On-Policy Reinforcement-Learning Algorithms, ML 2000

20-Minute Intro to *Stochastic Approximation*

1. SA is Classic: First Proposed in 1951

A STOCHASTIC APPROXIMATION METHOD[1]

By Herbert Robbins and Sutton Monro
University of North Carolina

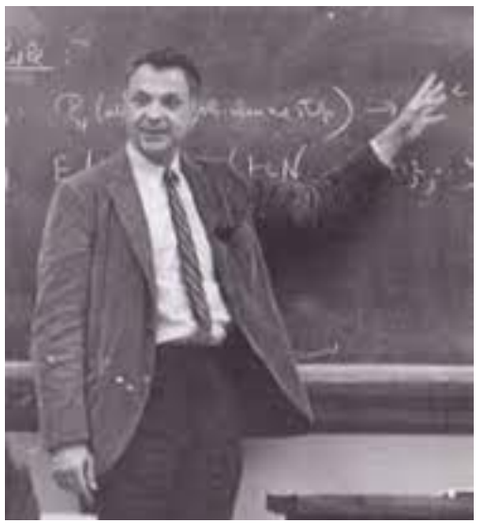
1. Summary. Let $M(x)$ denote the expected value at level x of the response to a certain experiment. $M(x)$ is assumed to be a monotone function of x but is unknown to the experimenter, and it is desired to find the solution $x = \theta$ of the equation $M(x) = \alpha$, where α is a given constant. We give a method for making successive experiments at levels $x_1, x_2, \cdots$ in such a way that x_n will tend to θ in probability.

2. Introduction. Let $M(x)$ be a given function and α a given constant such that the equation

$$M(x) = \alpha \tag{1}$$

has a unique root $x = \theta$. There are many methods for determining the value of θ by successive approximation. With any such method we begin by choosing one or more values $x_1, \cdots, x_r$ more or less arbitrarily, and then successively obtain new values x_n as certain functions of the previously obtained $x_1, \cdots, x_{n-1}$, the values $M(x_1), \cdots, M(x_{n-1})$, and possibly those of the derivatives $M'(x_1), \cdots, M'(x_{n-1})$, etc. If

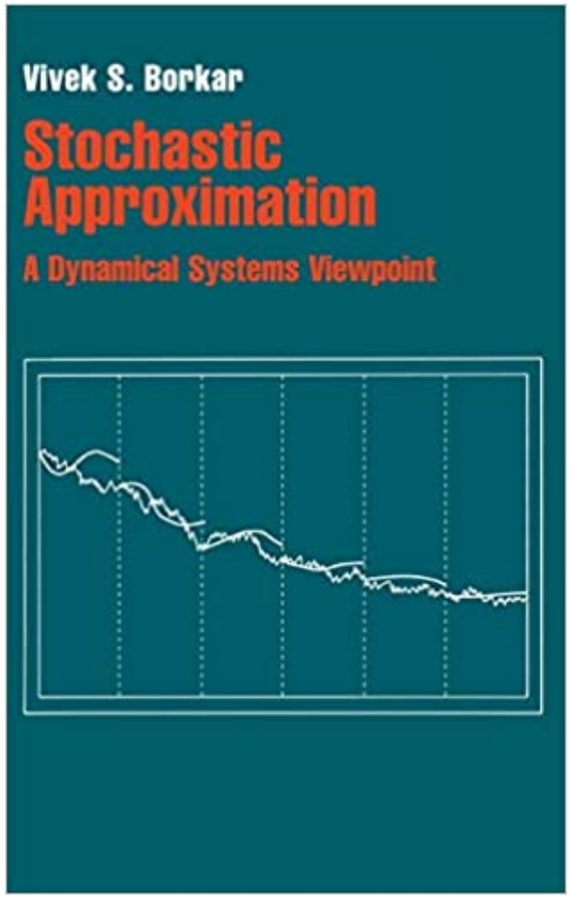
$$\lim_{n\to\infty} x_n = \theta, \tag{2}$$



Herbert Robbins

Sutton Monro

Vivek Borkar
(IIT Bombay)



2. SA is Powerful: A Lot of Recent RL Papers that Use SA for Establishing Convergence!

Addressing Function Approximation Error in Actor-Critic Methods

Scott Fujimoto [1] Herke van Hoof [2] David Meger [1]

Abstract

In value-based reinforcement learning methods such as deep Q-learning, function approximation errors are known to lead to overestimated value estimates and suboptimal policies. We show that this problem persists in an actor-critic setting and propose novel mechanisms to minimize its effects on both the actor and the critic. Our algorithm builds on Double Q-learning, by taking the minimum value between a pair of critics to limit overestimation. We draw the connection between target networks and overestimation bias, and suggest delaying policy updates to reduce per-update error and further improve performance. We evaluate our method on the suite of OpenAI gym tasks, outperforming the state of the art in every environment tested.

means using an imprecise estimate within each update will lead to an accumulation of error. Due to overestimation bias, this accumulated error can cause arbitrarily bad states to be estimated as high value, resulting in suboptimal policy updates and divergent behavior.

This paper begins by establishing this overestimation property is also present for deterministic policy gradients (Silver et al., 2014), in the continuous control setting. Furthermore, we find the ubiquitous solution in the discrete action setting, Double DQN (Van Hasselt et al., 2016), to be ineffective in an actor-critic setting. During training, Double DQN estimates the value of the current policy with a separate target value function, allowing actions to be evaluated without maximization bias. Unfortunately, due to the slow-changing policy in an actor-critic setting, the current and target value estimates remain too similar to avoid maximization bias. This can be dealt with by adapting an older variant, Double

TD3 (ICML 2018)

Reward Constrained Policy Optimization

Chen Tessler[1], Daniel J. Mankowitz[2], and Shie Mannor[1]

[1] *Technion Israel Institute of Technology, Haifa, Israel*
[2] *DeepMind, London, England*
chen.tessler@campus.technion.ac.il, dmankowitz@google.com, shie@ee.technion.ac.il

Abstract

Solving tasks in Reinforcement Learning is no easy feat. As the goal of the agent is to maximize the accumulated reward, it often learns to exploit loopholes and misspecifications in the reward signal resulting in unwanted behavior. While constraints may solve this issue, there is no closed form solution for general constraints. In this work, we present a novel multi-timescale approach for constrained policy optimization, called 'Reward Constrained Policy Optimization' (RCPO), which uses an alternative penalty signal to guide the policy towards a constraint satisfying one. We prove the convergence of our approach and provide empirical evidence of its ability to train constraint satisfying policies.

RCPO (ICLR 2019)

Maxmin Q-learning: Controlling the Estimation Bias of Q-learning

Qingfeng Lan, Yangchen Pan, Alona Fyshe, Martha White
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada
{qlan3,pan6,alona,whitem}@ualberta.ca

Maxmin Q-learning (ICLR 2020)

Abstract

Q-learning suffers from overestimation bias, because it approximates the maximum action value using the maximum estimated action value. Algorithms have been proposed to reduce overestimation bias, but we lack an understanding of how bias interacts with performance, and the extent to which existing algorithms mitigate bias. In this paper, we 1) highlight that the effect of overestimation bias on learning efficiency is environment-dependent; 2) propose a generalization of Q-learning, called *Maxmin Q-learning*, which provides a parameter to flexibly control bias; 3) show theoretically that there exists a parameter choice for Maxmin

Distributional Policy Optimization: An Alternative Approach for Continuous Control

Chen Tessler*, Guy Tennenholtz* and Shie Mannor

* Equal Contribution
chen.tessler@campus.technion.ac.il, guytenn@gmail.com, shie@ee.technion.ac.il
Technion Institute of Technology, Haifa, Israel

Abstract

We identify a fundamental problem in policy gradient-based methods in continuous control. As policy gradient methods require the agent's underlying probability distribution, they limit policy representation to parametric distribution classes. We show that optimizing over such sets results in local movement in the action space and thus convergence to sub-optimal solutions. We suggest a novel distributional framework, able to represent arbitrary distribution functions over the continuous action space. Using this framework, we construct a generative scheme, trained using an off-policy actor-critic paradigm, which we call the Generative Actor Critic (GAC). Compared to policy gradient methods, GAC does not require knowledge of the underlying probability distribution, thereby overcoming these limitations. Empirical evaluation shows that our approach is comparable and often surpasses current state-of-the-art baselines in continuous domains.

Generative Actor-Critic (NeurIPS 2019)

Review: Second-Order Taylor Expansion

- **Taylor's Theorem**:

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a twice continuously differentiable function.

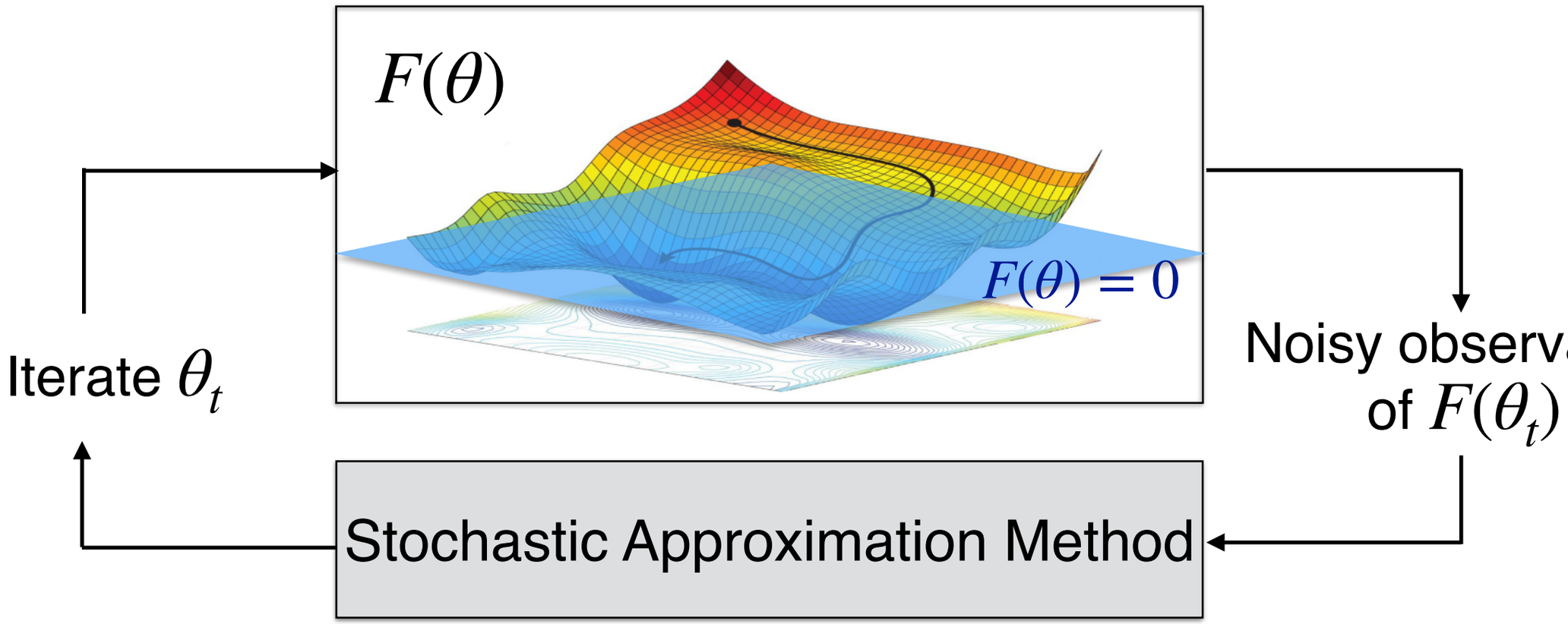
(1) For all $x \in \mathbb{R}^n$ and for all $\delta \in \mathbb{R}^n$, we have

$$f(x+\delta) = f(x) + \delta^T \nabla f(x) + \frac{1}{2}\delta^T \nabla^2 f(x)\delta + o(\|\delta\|^2)$$

(2) For all $x \in \mathbb{R}^n$ and for all $\delta \in \mathbb{R}^n$, there exists an $\alpha \in [0,1]$ such that

$$f(x+\delta) = f(x) + \delta^T \nabla f(x) + \frac{1}{2}\delta^T \nabla^2 f(x+\alpha\delta)\delta$$

- Goal of Stochastic Approximation: Given $F(\cdot) : \mathbb{R}^d \to \mathbb{R}^n$, iteratively solve an equation $F(\theta) = 0$ based on "noisy" measurements

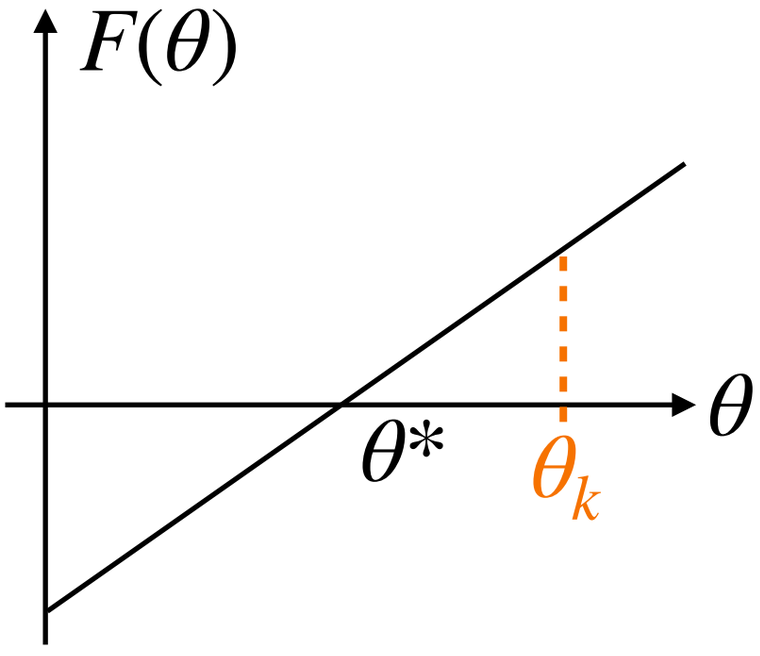


Question: For Sarsa, what is the corresponding $F(\theta)$ and θ?

- Goal of Stochastic Approximation: Given $F(\cdot): \mathbb{R}^d \to \mathbb{R}^n$, iteratively solve an equation $F(\theta) = 0$ based on "noisy" measurements

---

- Toy Example: Suppose $d = n = 1$ and $F(\theta)$ satisfies that:
  - (1) $F'(\theta) \in (0, M), \forall \theta \in \mathbb{R}$ (and hence $F(\theta)$ is strictly increasing)
  - (2) $|F''(\theta)| \leq L, \forall \theta \in \mathbb{R}$
  - Moreover, we can only get "noisy" samples, i.e., $Y(\theta; \varepsilon) = F(\theta) + \varepsilon$ (ε is an i.i.d. noise with $\mathbb{E}[\varepsilon] = 0$ and $\mathbb{V}[\varepsilon] = 1$)



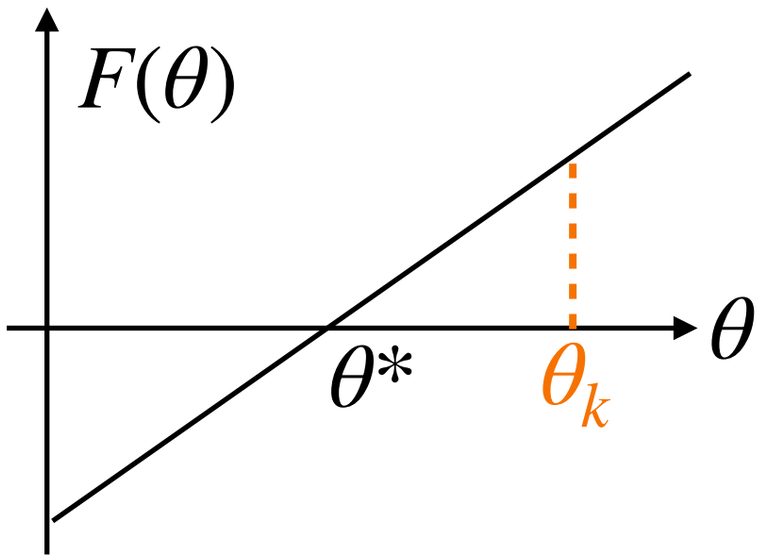
- Stochastic approximation

Let θ_k be the sample point of k-th iteration

$$\theta_{k+1} = \theta_k - \alpha_k Y(\theta_k; \varepsilon)$$

Will θ_k converge to θ^* ?

- Toy Example: Suppose $d = n = 1$ and $F(\theta)$ satisfies that:
  - $F'(\theta) = 1, \forall \theta \in \mathbb{R}$ (and hence $F(\theta)$ is strictly increasing)
  - Moreover, we can only get "noisy" samples, i.e., $Y(\theta; \varepsilon) = F(\theta) + \varepsilon$ (ε is an i.i.d. noise with $\mathbb{E}[\varepsilon] = 0$ and $\mathbb{V}[\varepsilon] = 1$)



$$\theta_{k+1} = \theta_k - \alpha_k Y(\theta_k; \varepsilon) \quad \text{(This is the SA update)}$$

$$= \theta_k + \underbrace{(-\alpha_k F(\theta_k) - \alpha_k \varepsilon_k)}_{=:\Delta}$$

$$F(\theta_{k+1}) = F(\theta_k) + \underbrace{\qquad\qquad}_{\text{First-order term}} + \underbrace{\qquad\qquad}_{\text{Second-order term}}$$

$$= F(\theta_k)\big(1 - \alpha_k F'(\theta_k)\big) - \alpha_k \varepsilon_k F'(\theta_k) + O\big(\alpha_k^2 (F(\theta_k) + \varepsilon_k)^2\big)$$

$$F(\theta_{k+1}) = F(\theta_k)(1 - \alpha_k F'(\theta_k)) - \alpha_k \varepsilon_k F'(\theta_k) + O(\alpha_k^2 (F(\theta_k) + \varepsilon_k)^2)$$

$$\mathbb{E}[F(\theta_{k+1})|\theta_k] = \mathbb{E}[F(\theta_k)(1 - \alpha_k F'(\theta_k))|\theta_k]$$

$$\underbrace{-\mathbb{E}[\alpha_k \varepsilon_k F'(\theta_k)|\theta_k]}_{=0}$$

(First-order term has no effect in expectation)

$$+\mathbb{E}[O(\alpha_k^2 (F(\theta_k) + \varepsilon_k)^2)|\theta_k]$$

shall be made small

(Second-order term can be made small by choosing proper α_k)

Next Question: How to choose the step size α_k?

Review: High-School Math

Q1: Consider a sequence $a_{k+1} = (1-\alpha)a_k, \alpha \in (0,1), a_1 = 1$

Then, $\lim_{k\to\infty} a_k = ?$

Q2: Consider a sequence $a_{k+1} = (1-\alpha_k) \cdot a_k, a_0 = 1$

If $\alpha_k = 1/(k+2)$, then $\lim_{k\to\infty} a_k = ?$

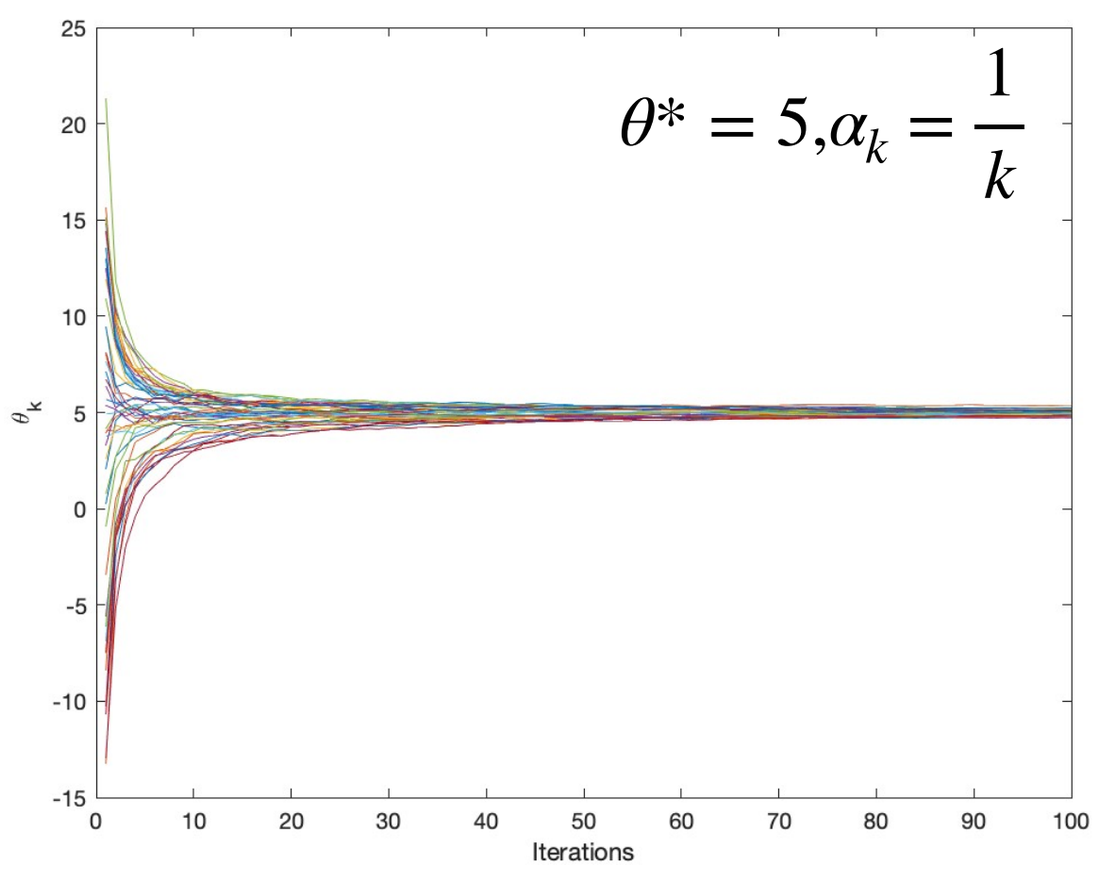
$$\mathbb{E}[F(\theta_{k+1})|\theta_k] = \mathbb{E}[F(\theta_k)(1 - \alpha_k F'(\theta_k))|\theta_k] + \mathbb{E}[O(\alpha_k^2 (F(\theta_k) + \varepsilon_k)^2)|\theta_k]$$

---

1. Suppose we choose $\alpha_k = \frac{1}{k}$, then what is $F(\theta_k)$ as $k \to \infty$?

$$\prod_{k=1}^{\infty} (1 - \frac{1}{k}) =$$

$$\sum_{k=1}^{\infty} \mathbb{E}\left[\frac{1}{k^2}\varepsilon_k^2\right] =$$



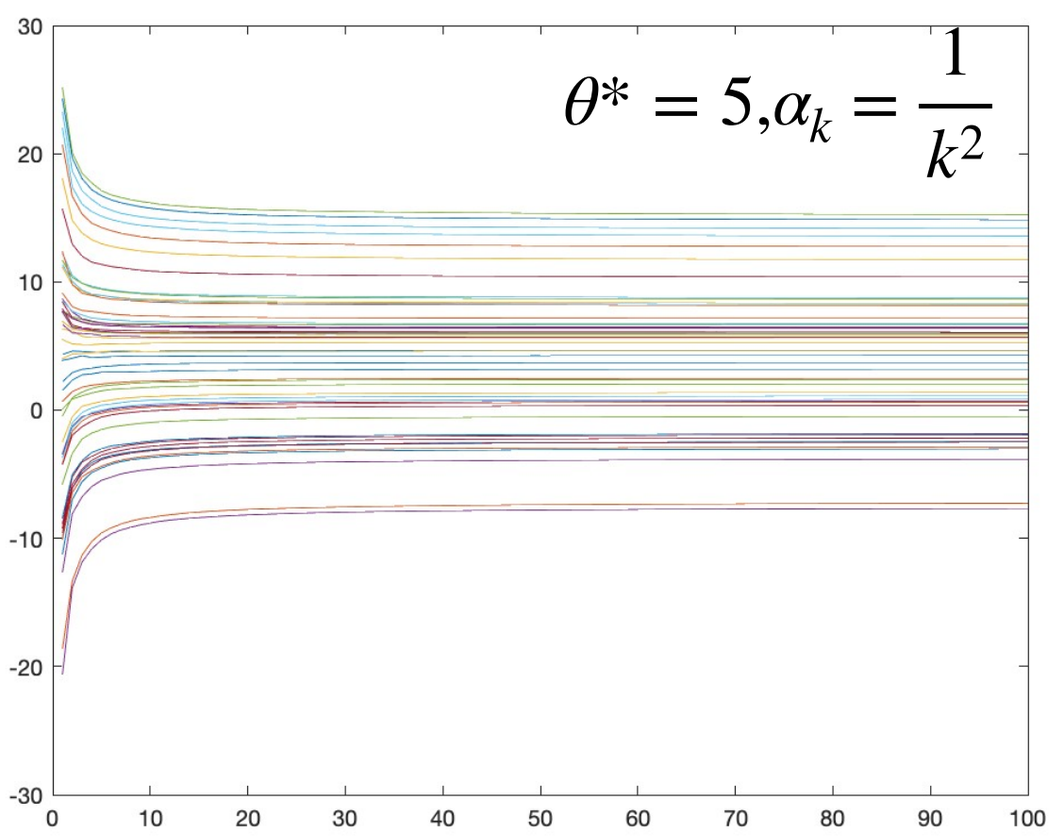
$$\mathbb{E}[F(\theta_{k+1})|\theta_k] = \mathbb{E}[F(\theta_k)(1 - \alpha_k F'(\theta_k))|\theta_k] + \mathbb{E}[O(\alpha_k^2 (F(\theta_k) + \varepsilon_k)^2)|\theta_k]$$

---

2. What if we choose $\alpha_k = \frac{1}{k^2}$, then what is $F(\theta_k)$ as $k \to \infty$?

$$\prod_{k=1}^{\infty} (1 - \frac{1}{k^2}) =$$

$$\sum_{k=1}^{\infty} \mathbb{E}\left[\frac{1}{k^4}\varepsilon_k^2\right] =$$



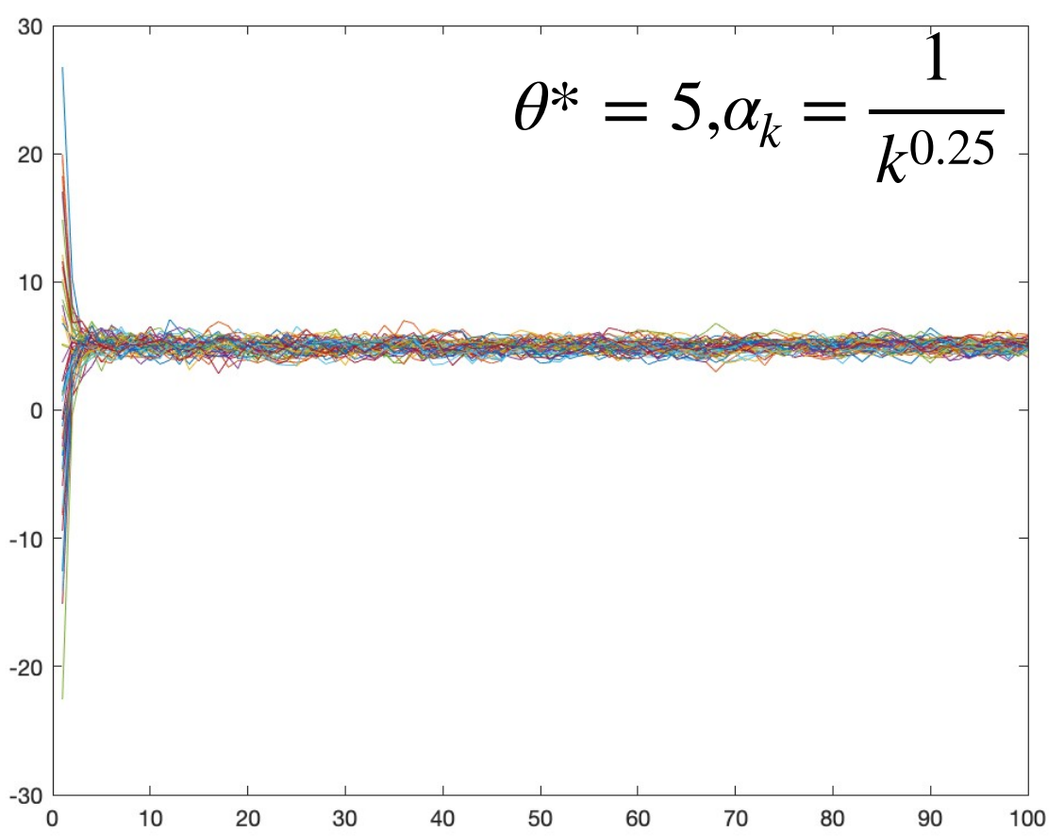
$$\mathbb{E}[F(\theta_{k+1})|\theta_k] = \mathbb{E}[F(\theta_k)(1-\alpha_k F'(\theta_k))|\theta_k] + \mathbb{E}[O(\alpha_k^2(F(\theta_k)+\varepsilon_k)^2)|\theta_k]$$

---

3. What if we choose $\alpha_k = \frac{1}{k^{0.25}}$, then what is $F(\theta_k)$ as $k \to \infty$?

$$\prod_{k=1}^{\infty}(1-\frac{1}{k^{0.25}}) =$$

$$\sum_{k=1}^{\infty}\mathbb{E}\left[\frac{1}{k^{0.5}}\varepsilon_k^2\right] =$$



A Popular Variant of SA Theorem

$$\Delta_{n+1}(x) = (1 - \alpha_n(x)) \cdot \Delta_n(x) + \alpha_n(x) \cdot \varepsilon_n(x)$$

- **Stochastic Approximation (Jaakkola, Jordan, Singh, 1993)**:

  If the following conditions are satisfied for all $x \in \mathcal{X}$:

  (SA1) $0 \leq \alpha_n(x) \leq 1, \sum_{n=1}^{\infty} \alpha_n(x) = \infty, \sum_{n=1}^{\infty} \alpha_n(x)^2 < \infty$, w.p.1

  (SA2) $\left\| \mathbb{E}\left[\varepsilon_n \mid \mathcal{H}_n\right] \right\|_\infty \leq \rho \|\Delta_n\|_\infty + c_n, \quad \rho \in [0,1)$ and $c_n \to 0$ w.p.1

  (SA3) $\mathbb{V}\left[\varepsilon_n(x) \mid \mathcal{H}_n\right] \leq C(1 + \|\Delta_n\|_\infty)^2$, where C is some constant

  then $\Delta_n \to 0$, w.p.1

Jaakkola et al., “On the Convergence of Stochastic Iterative Dynamic Programming Algorithms”, 1993

How to Interpret SARSA as Stochastic Approximation?

SA Update $\Delta_{n+1}(x) = (1 - \alpha_n(x)) \cdot \Delta_n(x) + \alpha_n(x) \cdot \varepsilon_n(x)$

SARSA $Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t(s_t, a_t)(r_t + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t))$

(Note that $\Delta_n, \alpha_n, \varepsilon_n$ are vectors)

$x \Leftrightarrow$

$\Delta_n(x) \Leftrightarrow$

$\alpha_n(x) \Leftrightarrow$

$\varepsilon_n(x) \Leftrightarrow$

Step 1: Interpret SARSA as SA (Formally)

SA Update $\Delta_{n+1}(x) = (1 - \alpha_n(x)) \cdot \Delta_n(x) + \alpha_n(x) \cdot \varepsilon_n(x)$

SARSA $Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t(s_t, a_t)(r_t + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t))$

$Q_{t+1}(s, a) = Q_t(s, a)$, for other $(s, a) \neq (s_t, a_t)$

$x \Leftrightarrow$ (s, a) pairs

$\Delta_n(x) \Leftrightarrow$ $Q_t(s, a) - Q^*(s, a)$

$\alpha_n(x) \Leftrightarrow$ For (s_t, a_t): $\alpha_t(s_t, a_t)$
But for other $(s, a) \neq (s_t, a_t)$: 0

$\varepsilon_n(x) \Leftrightarrow$ For (s_t, a_t): $r_t + \gamma Q_t(s_{t+1}, a_{t+1}) - Q^*(s_t, a_t) =: \varepsilon_t(s_t, a_t)$ (TD error w.r.t. Q^*)
But for other $(s, a) \neq (s_t, a_t)$: 0

Step 2: A Closer Look at $\varepsilon_t(s_t, a_t)$

$$\varepsilon_t(s_t, a_t) = r_t + \gamma Q_t(s_{t+1}, a_{t+1}) - Q^*(s_t, a_t)$$

$$= \underbrace{r_t + \gamma \max_{b \in A} Q_t(s_{t+1}, b) - Q^*(s_t, a_t)}_{=: X_t(s_t, a_t)} + \underbrace{\left[\gamma Q_t(s_{t+1}, a_{t+1}) - \gamma \max_{b \in A} Q_t(s_{t+1}, b)\right]}_{=: Y_t(s_t, a_t)}$$

Next step: Check (SA2) by evaluating $\mathbb{E}[\varepsilon_t(s_t, a_t) | \mathscr{H}_t]$

- Q1: Given $\mathscr{H}_t$, what is random in $\varepsilon_t(s_t, a_t)$?
- Q2: Would $\mathbb{E}[X_t(s_t, a_t) | \mathscr{H}_t]$ be small?
- Q3: Would $\mathbb{E}[Y_t(s_t, a_t) | \mathscr{H}_t]$ be small?

Step 3: Verify the Condition (SA2)

(SA2) $\left|\mathbb{E}\left[\varepsilon_t \mid \mathscr{H}_t\right]\right| \leq \rho \|\Delta_t\|_\infty + c_t, \quad \rho \in [0,1)$ and $c_t \to 0$ w.p.1

(We only need to check $\mathbb{E}\left[\varepsilon_t(s_t, a_t) \mid \mathscr{H}_t\right]$)

$$\varepsilon_t(s_t, a_t) = \underbrace{r_t + \gamma \max_{b \in A} Q_t(s_{t+1}, b) - Q^*(s_t, a_t)}_{=: X_t(s_t, a_t)} + \underbrace{\left[\gamma Q_t(s_{t+1}, a_{t+1}) - \gamma \max_{b \in A} Q_t(s_{t+1}, b)\right]}_{=: Y_t(s_t, a_t)}$$

Fact 1: $\|\mathbb{E}[X_t|\mathscr{H}_t]\|_\infty \leq \gamma \cdot \|\Delta_t\|_\infty$ (by "contraction mapping" of one-step greedy policy improvement in Lec3)

Fact 2: $\|\mathbb{E}[Y_t|\mathscr{H}_t]\|_\infty \leq \epsilon_t(\gamma \cdot \frac{1}{1-\gamma})$ (by epsilon-greedy policies and reward boundedness)

Step 4: Verify the Condition (SA3)

(SA3) $\mathbb{V}\left[\varepsilon_n(s,a) \mid \mathscr{H}_n\right] \leq C(1 + \|\Delta_n\|_\infty)^2$, where C is some constant

---

(We only need to check $\mathbb{V}\left[\varepsilon_t(s_t,a_t) \mid \mathscr{H}_t\right]$

Recall that

$$\varepsilon_t(s_t,a_t) = \underbrace{r_t + \gamma \max_{b\in A} Q_t(s_{t+1},b) - Q^*(s_t,a_t)}_{=: X_t(s_t,a_t)} + \underbrace{\left[\gamma Q_t(s_{t+1},a_{t+1}) - \gamma \max_{b\in A} Q_t(s_{t+1},b)\right]}_{=: Y_t(s_t,a_t)}$$

Assignment for this lecture:

- Spend 30 minutes going through the idea of SA
- Spend 30 minutes verifying that SARSA satisfies (SA1)-(SA3)
  - Reference: https://link.springer.com/content/pdf/10.1023/A:1007678930559.pdf

Expected Sarsa

Rethinking Sarsa

- Recall: Bellman expectation equation for $Q^\pi(s,a)$

$$Q^\pi(s,a) = \sum_{s',r} P(s',r \mid s,a)\left(r + \gamma \sum_{a'} \pi(a' \mid s') Q^\pi(s',a')\right)$$

Sarsa:

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

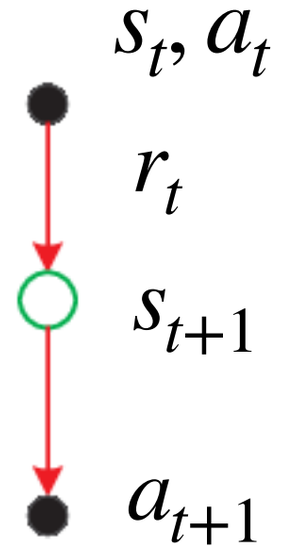
$$s' \sim P(\,|\,s,a)$$
$$a' \sim \pi(\,\cdot\,|\,s)$$

- Question: π is known to us. Do we still need sampling for a'?

Expected Sarsa vs Sarsa

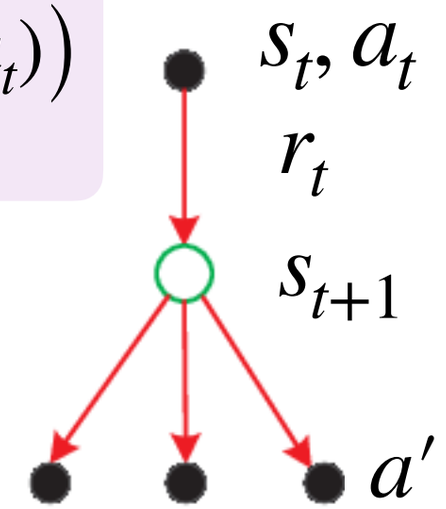
Sarsa:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\Big(r_t + \gamma Q(\underbrace{s_{t+1}, a_{t+1}}_{\substack{s_{t+1} \sim P(\cdot \mid s_t, a_t) \\ a_{t+1} \sim \pi(\cdot \mid s_{t+1})}}) - Q(s_t, a_t)\Big)$$



Expected Sarsa:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\Big(r_t + \gamma \sum_{a'} \pi(a' \mid s_{t+1}) Q(s_{t+1}, a') - Q(s_t, a_t)\Big)$$

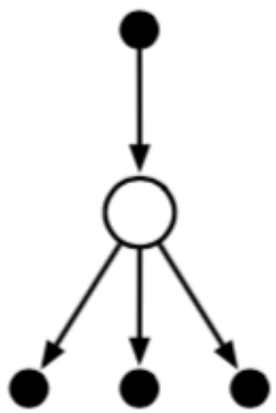


- Expected Sarsa has a lower variance
- Expected Sarsa requires slightly more computation

Stability of Expected Sarsa (Compared to Sarsa)

- Example: Suppose $R_{t+1} = 1.0$ and the following

$Q(S_{t+1}, a') =$	0.0	−1.0	2.0	1.0
$\pi(a' \mid S_{t+1}) =$	0.1	0.1	0.7	0.1



TD target under Sarsa:

$$1.0 + \gamma(2.0), \; w.p.0.1$$
$$1.0 + \gamma(-1.0), \; w.p.0.1$$
$$1.0 + \gamma(2.0), \; w.p.0.7$$
$$1.0 + \gamma(1.0), \; w.p.0.1$$

TD target under Expected Sarsa:

$$1.0 + \gamma \cdot 1.4$$

In practice, Expected Sarsa has lower variance than Sarsa

(Example Source: Martha White)

Expected Sarsa (Formally)

- Expected Sarsa With ε-Greedy policies:

  Step 1: Initialize $Q(s,a)$ for all (s,a), and initial state s_0

  Step 2: For each step $t = 0,1,2,\cdots$

  Select a_t using $\pi_t \sim \varepsilon$-greedy w.r.t $Q(s_t, \cdot)$

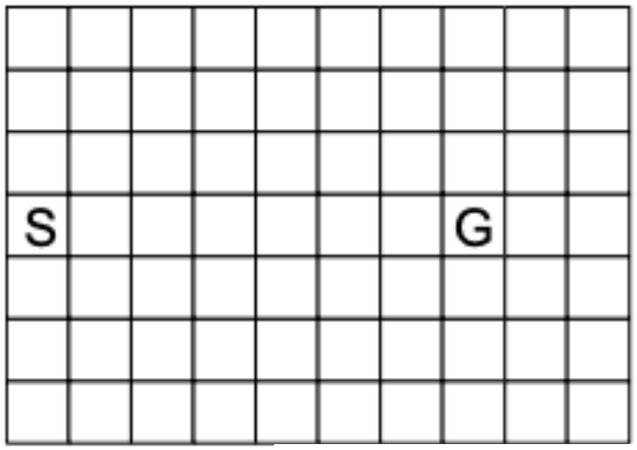
  Observe (r_{t+1}, s_{t+1})

  $$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\Big(r_{t+1} + \gamma \sum_{a'} \pi_t(a' \mid s_{t+1}) Q(s_{t+1},a') - Q(s_t,a_t)\Big)$$

- Question: Is “Expected Sarsa” *on-policy* or *off-policy*?

Comparison: Expected Sarsa vs Sarsa

- Example: Stochastic Windy Gridworld

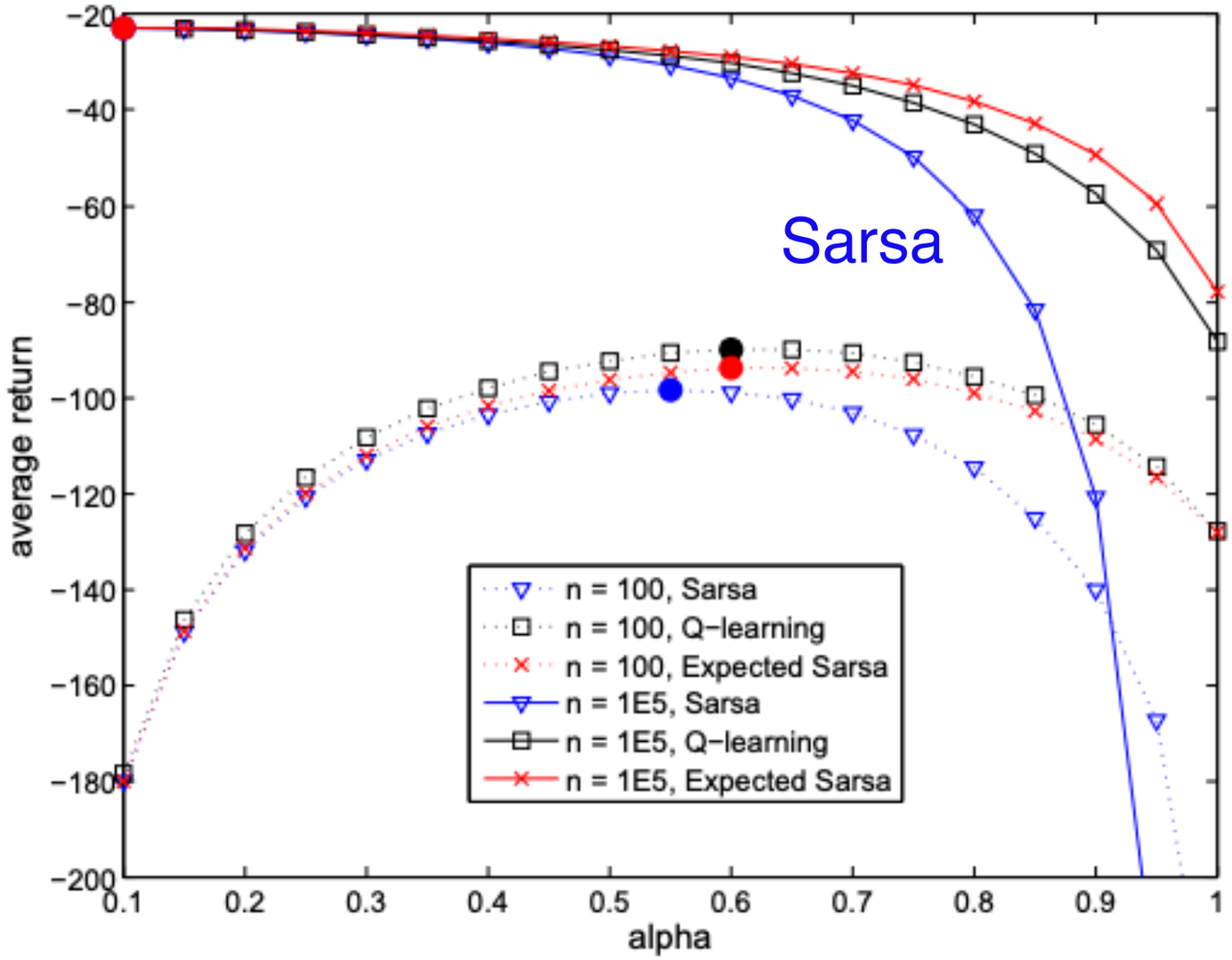


Actions: Left, Right, Up, Down

State transition:

- No wind at all, w.p. 0.8
- Wind in a random direction, w.p. 0.2

$(\varepsilon = 0.1)$



α = learning rate

Harm van Seijen et al., A Theoretical and Empirical Analysis of Expected Sarsa, 2009

Convergence of Expected Sarsa

- **Theorem**: Expected Sarsa converges to the optimal action-value function, i.e., $Q(s,a) \to Q_*(s,a)$, under the following conditions:

  (1) GLIE (2) $\sum_{t=1}^{\infty} \alpha_t(s,a) = \infty$, $\sum_{t=1}^{\infty} \alpha_t(s,a)^2 < \infty$, for all (s,a)

- Proof: Stochastic approximation (similar to the proof for Sarsa)

Harm van Seijen et al., A Theoretical and Empirical Analysis of Expected Sarsa, 2009

Q-Learning

Watkins and Dayan,”Q-Learning,” Machine Learning, 1992

Some Historical Accounts on Q-Learning

Technical Note

Q-Learning

CHRISTOPHER J.C.H. WATKINS
25b Framfield Road, Highbury, London N5 1UU, England

PETER DAYAN
Centre for Cognitive Science, University of Edinburgh, 2 Buccleuch Place, Edinburgh EH8 9EH, Scotland

Abstract. Q-learning (Watkins, 1989) is a simple way for agents to learn how to act optimally in controlled Markovian domains. It amounts to an incremental method for dynamic programming which imposes limited computational demands. It works by successively improving its evaluations of the quality of particular actions at particular states.

This paper presents and proves in detail a convergence theorem for Q-learning based on that outlined in Watkins (1989). We show that Q-learning converges to the optimum action-values with probability 1 so long as all actions are repeatedly sampled in all states and the action-values are represented discretely. We also sketch extensions to the cases of non-discounted, but absorbing, Markov environments, and where many Q values can be changed each iteration, rather than just one.

Keywords. Q-learning, reinforcement learning, temporal differences, asynchronous dynamic programming

(A paper in ML Journal in published 1992)

Chris Watkins
(Professor@U of London)

Peter Dayan
(Professor@Max Planck Institute)

Rethinking Expected Sarsa: Deterministic Policies?

Expected Sarsa in general:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\Big(r + \gamma \sum_{a'} \pi(a'|s')Q(s',a') - Q(s,a)\Big)$$

Expected Sarsa when π is deterministic:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\Big(r + \gamma Q(s', \pi(s')) - Q(s,a)\Big)$$

We have actually seen similar update!

(From Lec14) Off-policy deterministic actor-critic (OPDAC)

$$\Delta w_k \leftarrow \Delta w_k + \alpha_w \Big(r_t + \gamma Q_{w_k}(s_{t+1}, \pi_\theta(s_{t+1})) - Q_{w_k}(s_t, a_t)\Big) \nabla_w Q_w(s_t, a_t)|_{w=w_k}$$

From Expected Sarsa to Q-Learning

Expected Sarsa when π is deterministic:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\Big(r + \gamma Q(s', \pi(s')) - Q(s,a)\Big)$$

- Idea: Let's allow both behavior and target policies to improve

Target policy π: Greedy w.r.t. $Q(s,a)$

Behavior policy β: ε-Greedy w.r.t. $Q(s,a)$

Q-Learning under the above π, β:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\Big(r + \gamma \cdot \max_{a' \in \mathcal{A}} Q(s', a') - Q(s,a)\Big)$$

Q-learning is an "off-policy" version of Expected Sarsa!

Q-Learning Algorithm (With ε-Greedy Exploration)

- Q-Learning:

Step 1: Initialize $Q(s, a)$ for all (s, a), and initial state s_0

Step 2: For each step $t = 0,1,2,\cdots$

Select a_t using ε-greedy w.r.t $Q(s_t, \cdot)$

Observe (r_{t+1}, s_{t+1})

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t(s_t, a_t)\left(r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)\right)$$

- Question: Is “Q-learning” *on-policy* or *off-policy*?

Convergence of Q-Learning

- **Theorem**: Q-learning converges to the optimal action-value function, i.e., $Q(s,a) \rightarrow Q_*(s,a)$, under the following conditions:

(1) GLIE (2) $\sum_{t=1}^{\infty} \alpha_t(s,a) = \infty$, $\sum_{t=1}^{\infty} \alpha_t(s,a)^2 < \infty$, for all (s,a)

- Proof: Stochastic approximation (similar to the proof for Sarsa)

Watkins and Dayan,"Q-Learning," Machine Learning, 1992

How to Interpret Q-Learning as SA?

Recall: A Popular Variant of SA Theorem

$$\Delta_{n+1}(x) = (1 - \alpha_n(x)) \cdot \Delta_n(x) + \alpha_n(x) \cdot \varepsilon_n(x)$$

- **Stochastic Approximation (Jaakkola, Jordan, Singh, 1993)**:

  If the following conditions are satisfied for all $x \in \mathcal{X}$:

  (SA1) $0 \leq \alpha_n(x) \leq 1, \sum_{n=1}^{\infty} \alpha_n(x) = \infty, \sum_{n=1}^{\infty} \alpha_n(x)^2 < \infty$, w.p.1

  (SA2) $\left\| \mathbb{E}\left[\varepsilon_n \mid \mathcal{H}_n\right] \right\|_\infty \leq \rho \|\Delta_n\|_\infty + c_n, \quad \rho \in [0,1)$ and $c_n \to 0$ w.p.1

  (SA3) $\mathbb{V}\left[\varepsilon_n(x) \mid \mathcal{H}_n\right] \leq C(1 + \|\Delta_n\|_\infty)^2$, where C is some constant

  then $\Delta_n \to 0$, w.p.1

Jaakkola et al., “On the Convergence of Stochastic Iterative Dynamic Programming Algorithms”, 1993

How to Interpret Q-Learning as SA?

SA Update $\Delta_{n+1}(x) = (1 - \alpha_n(x)) \cdot \Delta_n(x) + \alpha_n(x) \cdot \varepsilon_n(x)$

Q-Learning $Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t(s_t, a_t)(r_t + \gamma \max_{a'} Q_t(s_{t+1}, a') - Q_t(s_t, a_t))$

$Q_{t+1}(s, a) = Q_t(s, a), \quad \text{for other} (s, a) \neq (s_t, a_t)$

$x \Leftrightarrow$

$\Delta_n(x) \Leftrightarrow$

$\alpha_n(x) \Leftrightarrow$

$\varepsilon_n(x) \Leftrightarrow$

$\mathscr{H}_n \Leftrightarrow$

Step 1: Interpret Q-Learning as SA (Formally)

SA Update $\Delta_{n+1}(x) = (1 - \alpha_n(x)) \cdot \Delta_n(x) + \alpha_n(x) \cdot \varepsilon_n(x)$

Q-Learning $Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t(s_t, a_t)(r_t + \gamma \max_{a'} Q_t(s_{t+1}, a') - Q_t(s_t, a_t))$

$Q_{t+1}(s, a) = Q_t(s, a), \quad$ for other $(s, a) \neq (s_t, a_t)$

$x \Leftrightarrow$ (s, a) pairs

$\Delta_n(x) \Leftrightarrow Q_t(s, a) - Q^*(s, a)$

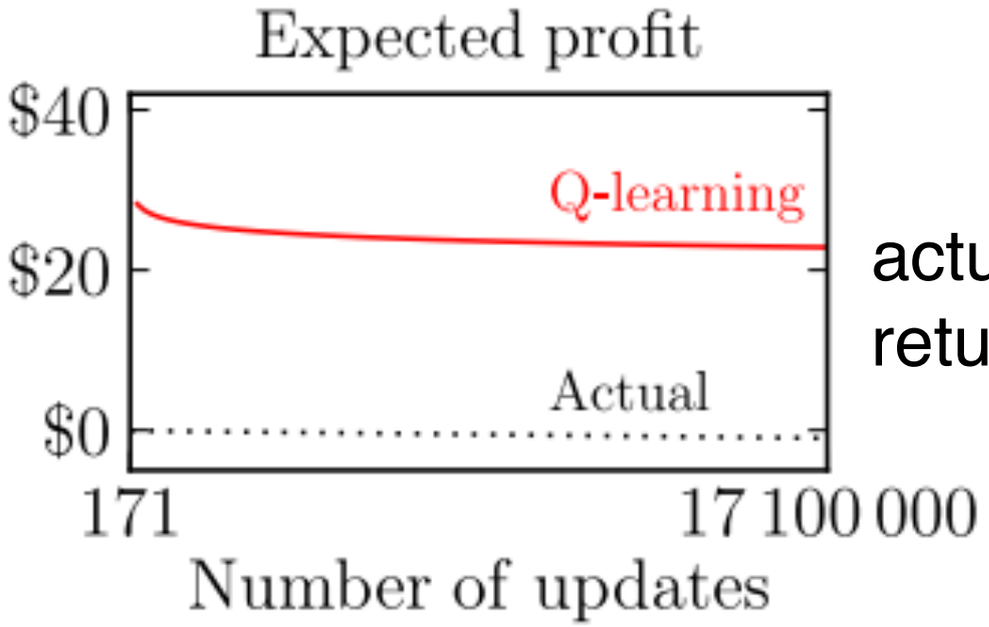
$\alpha_n(x) \Leftrightarrow$ For (s_t, a_t): $\alpha_t(s_t, a_t)$
But for other $(s, a) \neq (s_t, a_t)$: 0

$\varepsilon_n(x) \Leftrightarrow$ For (s_t, a_t): $r_t + \gamma \max_{a'} Q_t(s_{t+1}, a') - Q^*(s_t, a_t) =: \varepsilon_t(s_t, a_t)$
But for other $(s, a) \neq (s_t, a_t)$: 0

$\mathcal{H}_n \Leftrightarrow \{s_0, a_0, r_1, \cdots, s_t, a_t\}$

An Issue With Q-Learning: Overestimation Bias

- Example: Roulette with 1 state and 171 actions (assume $1 for each bet)



Average action values for Q-learning on roulette with synchronous updates, $\alpha = 1/n$ and $\gamma = 0.95$.

actual expected return = -$0.053

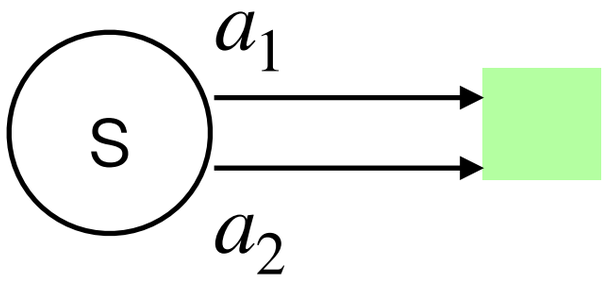
Q-learning after 10^5 trials: Each dollar yields ≈ $22

Q-learning can suffer from overestimation (with finite samples)!

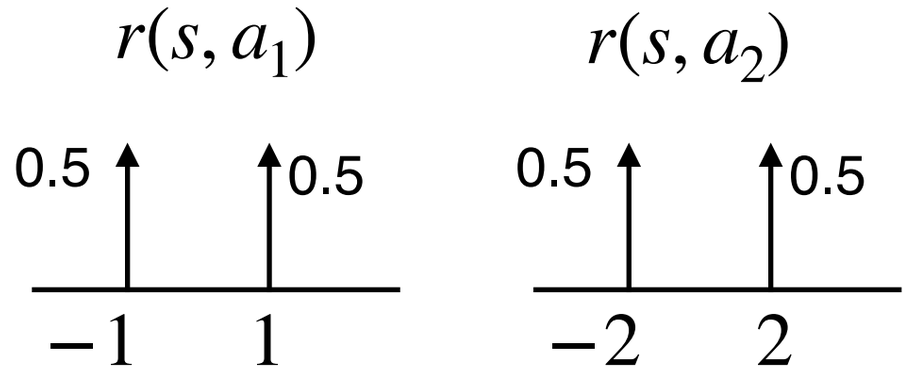
Hado van Hasselt et al., Double Q Learning, NIPS 2010

Overestimation Bias: A Motivating Example

- Example: 1-state MDP with 2 actions



terminal state



- Let $\hat{Q}(s, a_1)$, $\hat{Q}(s, a_2)$ be unbiased estimators (based on 1 reward sample)

- Overestimation:

$$\mathbb{E}\left[\max\{\hat{Q}(s, a_1), \hat{Q}(s, a_2)\}\right] > \max\left\{\mathbb{E}[\hat{Q}(s, a_1)], \mathbb{E}[\hat{Q}(s, a_2)]\right\} = \max_a Q(s, a)$$

$$\max\left\{\mathbb{E}[\hat{Q}(s, a_1)], \mathbb{E}[\hat{Q}(s, a_2)]\right\} =$$

$$\mathbb{E}\left[\max\{\hat{Q}(s, a_1), \hat{Q}(s, a_2)\}\right] =$$

Double Q-Learning

Hado van Hasselt et al., Double Q Learning, NIPS 2010

How to Mitigate Overestimation Bias: Double Estimators

- Observation: Overestimation bias can occur under a greedy policy w.r.t the estimated Q function
- Idea: Avoid using "max of estimates" as "estimate of max"

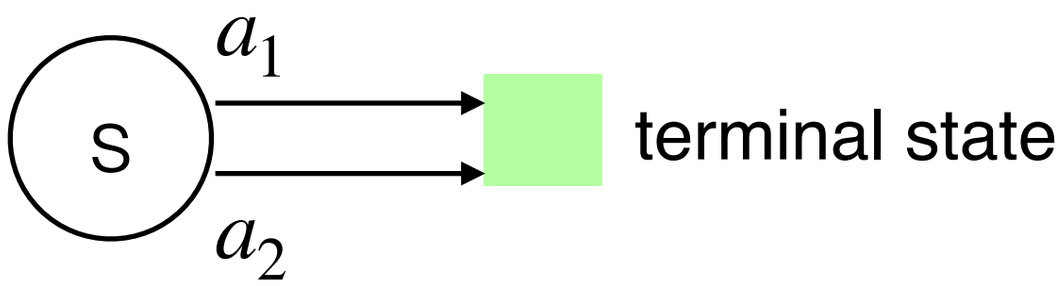
  - Create 2 independent unbiased estimates $\hat{Q}_1(s,a), \hat{Q}_2(s,a)$
  - Use one estimate to select action: $a^* = \arg\max_a Q_1(s,a)$
  - Use the other estimate for evaluate a^*: $\hat{Q}_2(s,a^*)$
  - Obtain an unbiased estimate: $\mathbb{E}[\hat{Q}_2(s,a^*)] = Q(s,a^*)$

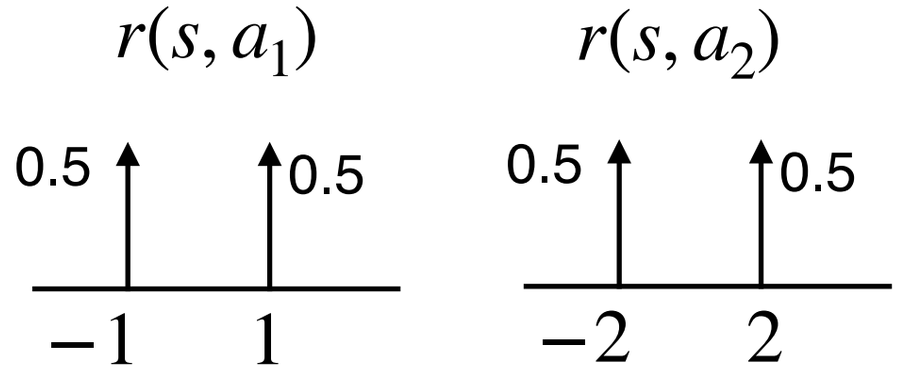
(Unfortunately, unbiased only for $Q(s,a^*)$, not for $\max_a Q(s,a)$)

Hado van Hasselt et al., Double Q Learning, NIPS 2010

Estimation Bias of Double Estimators

- Example: 1-state MDP with 2 actions





- Create 2 independent unbiased estimates $\hat{Q}_A(s,a), \hat{Q}_B(s,a)$
  - Use one estimate to select action: $\bar{a} = \arg\max_a \hat{Q}_A(s,a)$
  - Use the other estimate for evaluate $\bar{a}$: $\hat{Q}_B(s,\bar{a})$
  - Obtain an unbiased estimate: $\mathbb{E}[\hat{Q}_B(s,\bar{a})] = Q(s,\bar{a})$

$$\max_a Q(s,a) = \max\left\{\mathbb{E}[\hat{Q}_{A/B}(s,a_1)], \mathbb{E}[\hat{Q}_{A/B}(s,a_2)]\right\} = 0$$

$$\mathbb{E}[\hat{Q}_B(s,\bar{a})] =$$

How to extend such idea to general MDPs?

Double Q-Learning Algorithm (Formally)

- Double Q-Learning:

Step 1: Initialize $Q^A(s,a), Q^B(s,a)$ for all (s,a), and initial state s_0

Step 2: For each step $t = 0,1,2,\cdots$

Select a_t using ε-greedy w.r.t $Q^A(s_t,a) + Q^B(s_t,a)$

Observe (r_{t+1}, s_{t+1})

Choose one of the following updates uniformly at random

$$Q^A(s_t,a_t) \leftarrow Q^A(s_t,a_t) + \alpha\left(r_{t+1} + \gamma Q^B(s_{t+1}, \arg\max_a Q^A(s_{t+1},a)) - Q^A(s_t,a_t)\right)$$

$$Q^B(s_t,a_t) \leftarrow Q^B(s_t,a_t) + \alpha\left(r_{t+1} + \gamma Q^A(s_{t+1}, \arg\max_a Q^B(s_{t+1},a)) - Q^B(s_t,a_t)\right)$$

- Question: *Memory & computation per step (compared to Q-learning)*?

van Hasselt et al., Double Q Learning, NIPS 2010

Convergence of Double Q-Learning

- Technical assumptions:

(A1) Both Q^A, Q^B receive infinite number of updates

(A2) Q^A, Q^B are stored in a lookup table

(A3) Each state-action pair is visited an infinite number of times

(A4) Step sizes: $\sum_k \alpha_k = \infty, \sum_k \alpha_k^2 < \infty$ (Robbins-Monro conditions)

- **Convergence Result**:

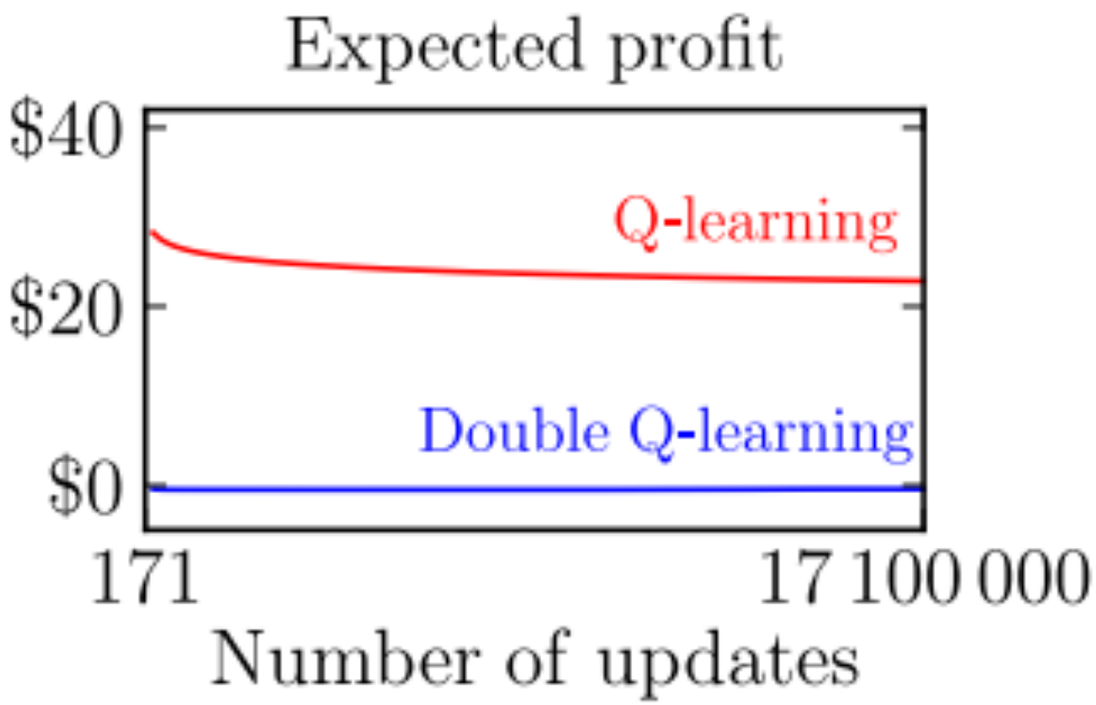
  Under the assumptions (A1)-(A4), both Q^A and Q^B converge to the optimal Q function, almost surely.

- Proof: Stochastic approximation (similar to Q-learning)

van Hasselt et al., Double Q Learning, NIPS 2010

Evaluation: Q-Learning and Double Q-Learning

- Example: Roulette with 1 state and 171 actions
  (assume $1 for each bet)

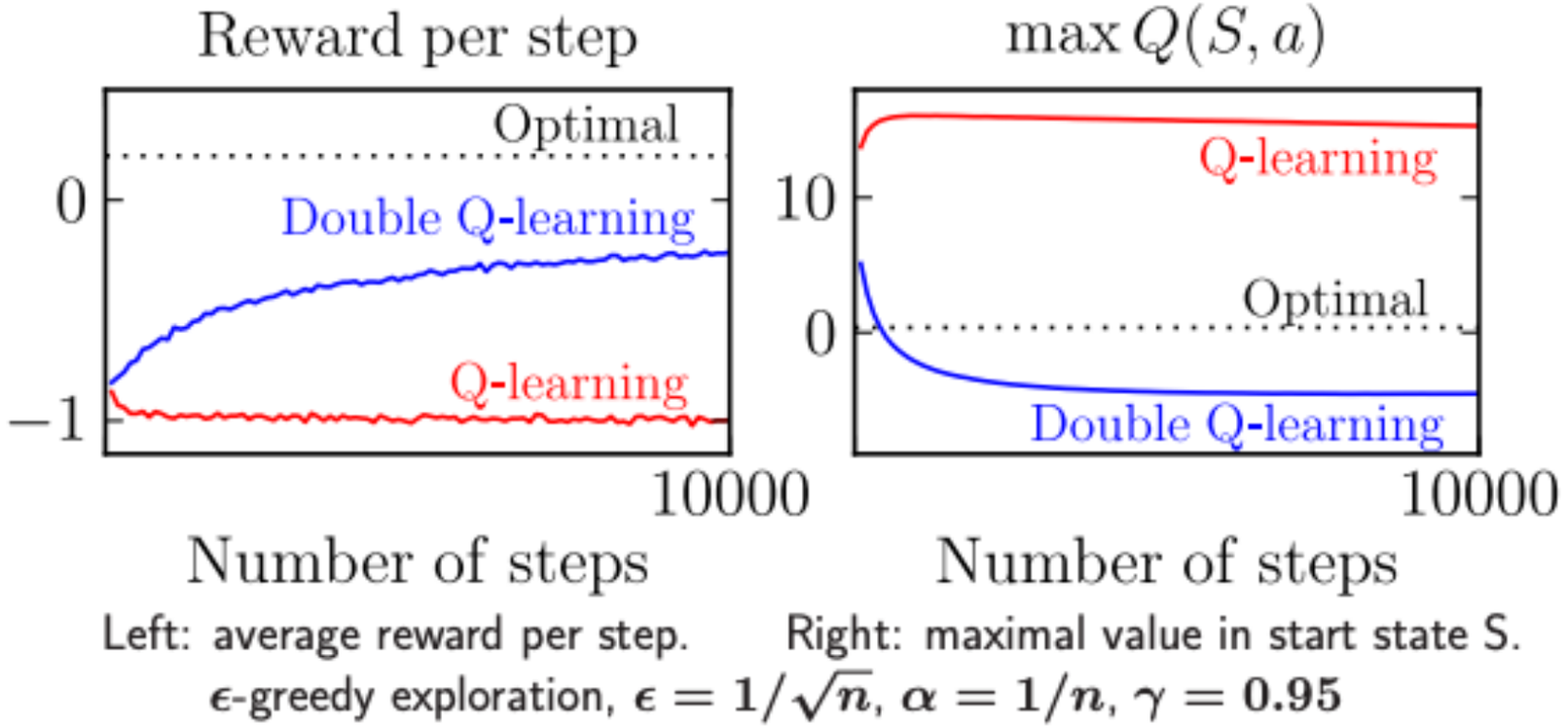


van Hasselt et al., Double Q Learning, NIPS 2010

Evaluation: Q-Learning & Double Q-Learning (Cont.)

- Example: Grid World with 9 states and 4 actions
  - Reward at the terminal state G: +5
  - Reward at any non-terminal state: -12 or +10 (random)
  - Under an optimal policy, an episode ends after 5 actions

		G
S		



Left: average reward per step. Right: maximal value in start state S.

ϵ-greedy exploration, $\epsilon = 1/\sqrt{n}$, $\alpha = 1/n$, $\gamma = 0.95$

Double Q-learning may lead to "under-estimation" (with finite samples)

van Hasselt et al., Double Q Learning, NIPS 2010

Appendix: Proof of ε-Greedy Policy Improvement Theorem

Recall from Lecture 3: Proof of Monotonic Policy Improvement by "Peeling off"

Recall from Lecture 3: Proof of Monotonic Policy Improvement by "Peeling off"

$$V^{\pi_k}(s) \leq \max_a Q^{\pi_k}(s,a)$$

← Step1: peel-off for one step

$$
\begin{aligned}
&= \max_a R(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V^{\pi_k}(s') \\
&= R(s, \pi_{k+1}(s)) + \gamma \sum_{s'} P(s' \mid s, \pi_{k+1}(s)) V^{\pi_k}(s') \\
&\leq R(s, \pi_{k+1}(s)) + \gamma \sum_{s'} P(s' \mid s, \pi_{k+1}(s)) \max_{a'} Q^{\pi_k}(s',a') \\
&= R(s, \pi_{k+1}(s)) + \gamma \sum_{s'} P(s' \mid s, \pi_{k+1}(s)) \\
&\qquad \times \left(R(s', \pi_{k+1}(s)) + \gamma \sum_{s''} P(s'' \mid s', \pi_{k+1}(s')) V^{\pi_k}(s'') \right) \\
&\quad \ldots \\
&= V^{\pi_{k+1}}(s)
\end{aligned}
$$

↑ Step 2: peel-off for all the remaining steps

Proof of ε-Greedy Policy Improvement: One-Step Comparison

Step 1: Show $V^{\pi}(s) \leq \sum_{a} \pi'(a \mid s) Q^{\pi}(s,a) =: Q^{\pi}(s, \pi'(s))$

$$V^{\pi}(s) = \sum_{a} \pi(a \mid s) Q^{\pi}(s,a)$$

This term ≥ 0 if π is ε-greedy

$$= \frac{\varepsilon}{|\mathcal{A}|} \sum_{a} Q^{\pi}(s,a) + (1-\varepsilon) \sum_{a} \boxed{\frac{\pi(a \mid s) - \frac{\varepsilon}{|\mathcal{A}|}}{1-\varepsilon}} Q^{\pi}(s,a)$$

$$\leq \frac{\varepsilon}{|\mathcal{A}|} \sum_{a} Q^{\pi}(s,a) + (1-\varepsilon) \cdot \max_{a} Q^{\pi}(s,a)$$

$$= \sum_{a} \pi'(a \mid s) Q^{\pi}(s,a) \equiv Q^{\pi}(s, \pi'(s))$$

Hence, we have $V^{\pi}(s) \leq Q^{\pi}(s, \pi'(s))$

Proof of ε-Greedy Policy Improvement: "Peeling off"

Step 2: Peel-off for all the remaining steps

$$
\begin{aligned}
V^{\pi}(s) &\leq Q^{\pi}(s, \pi'(s)) \\
&= R(s, \pi'(s)) + \gamma \sum_{s'} P(s' \mid s, \pi'(s)) V^{\pi}(s') \\
&\leq R(s, \pi'(s)) + \gamma \sum_{s'} P(s' \mid s, \pi'(s)) Q^{\pi}(s', \pi'(s')) \\
&= R(s, \pi'(s)) + \gamma \sum_{s'} P(s' \mid s, \pi'(s)) \\
&\qquad \times \left(R(s', \pi'(s')) + \gamma \sum_{s''} P(s'' \mid s', \pi'(s')) V^{\pi}(s'') \right) \\
&\quad \cdots \\
&= V^{\pi'}(s)
\end{aligned}
$$