Reinforcement Learning

Lecture 4: First RL algorithms

Alexandre Proutiere, Sadegh Talebi, Jungseul Ok

KTH, The Royal Institute of Technology

Objectives of this lecture

Introduce basic algorithms of reinforcement learning. Solving MDPs without the knowledge of the reward function and the transition probabilities.

- Preliminaries: Stochastic approximation
- Off-policy algorithm: Q-learning and its variants
- On-policy algorithm: SARSA

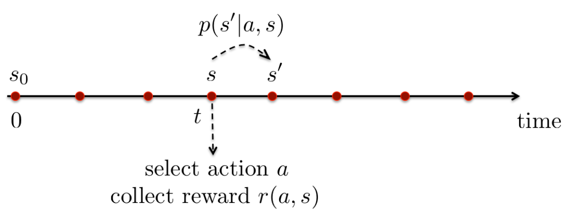
Lecture 4: outline

1. Q-learning and stochastic approximation
2. SARSA (State-Action-Reward-State-Action) algorithm
3. Improving Q-learning convergence rate

Lecture 4: outline

1. **Q-learning and stochastic approximation**
2. SARSA (State-Action-Reward-State-Action) algorithm
3. Improving Q-learning convergence rate

Infinite-horizon discounted MDP



- Stationary transition probabilities: $p(s'|s,a)$
- Stationary reward: $r(s,a)$, uniformly bounded
- Objective: for a given discount factor $\lambda \in [0,1)$, find a policy $\pi \in MD$ maximising (over all possible policies)

$$\lim_{T\to\infty} \mathbb{E}\left[\sum_{u=0}^{T} \lambda^u r(s_u^\pi, a_u^\pi)\right]$$

Discounted RL

- Learn $\pi^\star$ (the optimal policy) from the data
- Off-policy design problems. Data = (a given trajectory $(s_t, a_t, r(s_t, a_t))_{t=0}^{T}$), output at time T = a policy π_T
- On-policy design problems. In each step t:
  - Observe the transition s_{t-1}, a_{t-1}, s_t and the reward $r(s_{t-1}, a_{t-1})$
  - Devise a policy π_t and select the next action $a_t = \pi_t(s_t)$

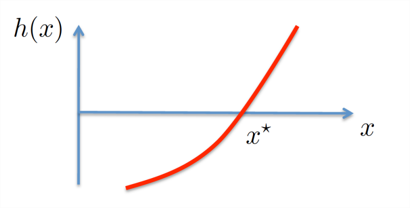
How can we solve Bellman's fixed point equation w/o knowing p and r?

$$V^\star(s) = \max_{a \in A_s}(r(s,a) + \lambda \sum_j p(j|s,a) V^\star(j)$$

$$\pi^\star(s) \in \arg\max_{a \in A_s}(r(s,a) + \lambda \sum_j p(j|s,a) V^\star(j)$$

Stochastic Approximation

Find the root of an increasing function from noisy measurements



Assume that at the n-th iteration, you select x_n

You get a noisy measurement $y_n = h(x_n) + M_n$ with $\mathbb{E}[M_n] = 0$

Stochastic Approximation: Example

$(M_n)_{n\geq 0}$ is i.i.d. Gaussian with mean 0 and variance 1.
Algorithm: $x_{n+1} = x_n - \frac{1}{n}(h(x_n) + M_n)$

Then

$$x_{n+p} = x_n - \sum_{t=n}^{n+p} \frac{h(x_t)}{t} - \sum_{t=n}^{n+p} \frac{M_t}{t}$$

The noise is averaged out after a while: for white Gaussian noise, $Var(\sum_{t\geq n} \frac{M_t}{t})$ tends to 0 as $n \to \infty$. "Hence", $x_n \to x_\infty$ as $n \to \infty$ almost surely where $h(x_\infty) = 0$

Robbins-Monro Algorithm (1951): $x_{n+1} = x_n - \alpha_n(h(x_n) + M_n)$

A generic SA algorithm

Let $X_n = (X_n(1), ..., X_n(d))^\top \in \mathbb{R}^d$ satisfying:

$$X_{n+1} = X_n + \alpha_n[h(X_n) + M_{n+1} + N_{n+1}] ,$$

Assumptions:

(A1) $h : \mathbb{R}^d \to \mathbb{R}^d$ is lipschitz

(A2) (Diminishing step sizes) $\sum_{n=1}^{\infty} \alpha_n = \infty$ and $\sum_{n=1}^{\infty} \alpha_n^2 < \infty$.

(A3) (Martingale difference) $\forall n$, $\mathbb{E}[M_{n+1}|\mathcal{F}_n] = 0$ where $\mathcal{F}_n = \sigma(X_0, M_1, N_1, \ldots, M_n, N_n, X_n)$ and $\forall n$, $\mathbb{E}[\|M_{n+1}\|^2 \mid \mathcal{F}_n] \leq c_0(1 + \|X_n\|^2)$.

(A4) (Additional noise) $\forall n$, .$\|N_n\|^2 \leq c_n(1 + \|X_n\|^2)$ a.s., where $\lim_{n\to\infty} c_n = 0$ a.s..

(A5) (Stability) $\dot{x} = h(x)$ has a unique globally stable equilibrium $x^\star$. $\forall x$, $h_\infty(x) = \lim_{c\to\infty} \frac{h(cx)}{c}$ exists and 0 is the only globally stable point of $\dot{x} = h_\infty(x)$.

Convergence

Let $X_n = (X_n(1), ..., X_n(d))^\top \in \mathbb{R}^d$ satisfying for all n:

$$X_{n+1} = X_n + \alpha_n[h(X_n) + M_{n+1} + N_{n+1}] \ ,$$

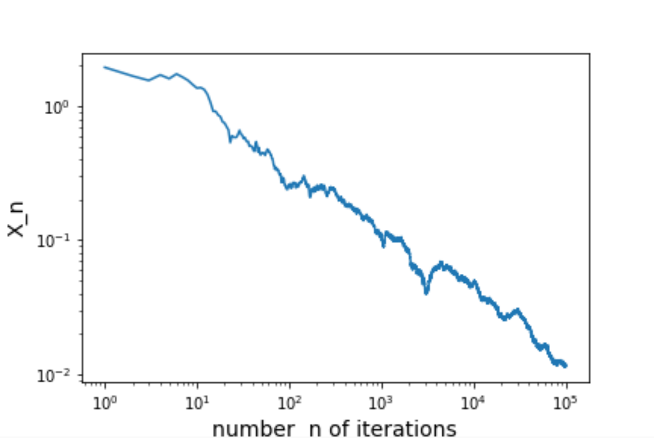
Theorem. If (A1)-(A5) hold, for any initial condition X_0,

$$\lim_{n\to\infty} X_n = x^\star, \quad \text{almost surely,}$$

where $x^\star$ is the only globally stable point of $\dot{x} = h(x)$.

Example

$h(x) = \arctan(x)$, $\alpha_n = 1/n$, $M_n \sim \mathcal{N}(0,1)$, $N_n = 0$



Asynchornous SA algorithm

Let $X_n = (X_n(1), ..., X_n(d))^\top \in \mathbb{R}^d$. At each iteration n, only a random set of coordinates $I_n \subset \{1, \dots, d\}$ of X_n are updated: for $1 \le i \le d$,

$$X_{n+1}(i) = \begin{cases} X_n(i) + \alpha_{\mathcal{I}_n(i)}[h(X_n; i) + M_{n+1}(i) + N_{n+1}(i)] & \text{if } i \in I_n \\ X_n(i) & \text{otherwise} \end{cases}$$

where $\mathcal{I}_n(i)$ is the number of updates of the i-th coordinate up to time n, i.e., $\mathcal{I}_n(i) := \sum_{m=0}^{n} 1[i \in I_m]$ and $h(x; i)$ is the i-th entry of $h(x)$.

Asynchornous SA algorithm

Assumptions:

(B1) (Linearly growing $\mathcal{I}_n(i)$) There exists a deterministic $\Delta > 0$ such that for all $1 \le i \le d$, $\liminf_{n\to\infty} \mathcal{I}_n(i)/n \ge \Delta$ a.s. Furthermore, for $c > 0$ and all $1 \le i, j \le d$, the limit of $\big(\sum_{m=\mathcal{I}_n(i)}^{\bar{\mathcal{I}}_n(c,i)} \alpha_m\big)/\big(\sum_{m=\mathcal{I}_n(j)}^{\bar{\mathcal{I}}_n(c,j)} \alpha_m\big)$ as $n \to \infty$ exists a.s. where $\bar{\mathcal{I}}_n(c,i) := \mathcal{I}_{N_n(c)}(i)$ with $N_n(c) := \min\big\{N > n : \sum_{m=n+1}^{N} \alpha_m > c\big\}$.

(B2) (Slowly decreasing α_n) The sequence $\{\alpha_n\}$ satisfies that $\alpha_{n+1} \le \alpha_n$ eventually and that for $c \in (0,1)$, $\sup_n \alpha_{\lfloor cn \rfloor}/\alpha_n < \infty$ and $\big(\sum_{m=0}^{\lfloor cn \rfloor} \alpha_m\big)/\big(\sum_{m=0}^{n} \alpha_m\big) \to 1$, where $\lfloor cn \rfloor$ is the integer part of cn.

Convergence

Let $X_n = (X_n(1), ..., X_n(d))^\top \in \mathbb{R}^d$ satisfying for all n and for $1 \leq i \leq d$,

$$X_{n+1}(i) = \begin{cases} X_n(i) + \alpha_{\mathcal{I}_n(i)}[h(X_n; i) + M_{n+1}(i) + N_{n+1}(i)] & \text{if } i \in I_n \\ X_n(i) & \text{otherwise} \end{cases}$$

Theorem. If (A1)-(A5) and (B1)-(B2) hold, for any initial condition X_0,

$$\lim_{n \to \infty} X_n = x^\star, \quad \text{almost surely,}$$

where $x^\star$ is the only globally stable point of $\dot{x} = h(x)$.

Application to discounted RL problems

Can we apply the R-M algorithm to evaluate the value function in an online manner?

- Remember that $V^\star$ is a fixed point of L, i.e., $L(V^\star) - V^\star = 0$

$$\forall s,\ V^\star(s) = \max_{a \in A_s}(r(s,a) + \lambda \sum_j p(j|s,a)V^\star(j))$$

- At time n, assume we have an estimate V_n of $V^\star$. Assume that we can try all actions a and observe the new state $S_{n+1}(s_n, a)$ a r.v. such that $\mathbb{P}[S_{n+1} = j | s_n, a] = p(j|s_n, a)$. We can compute

$$y_n = \max_{a \in A_{s_n}} \left(r(s_n, a) + \lambda V_n(S_{n+1}(s_n, a)) \right) - V_n(s_n)$$

Application to discounted RL problems

To apply the R-M algorithm to evaluate $V^\star$, we would need that $\mathbb{E}[y_n] = L(V_n)(s_n) - V_n(s_n)$. However:

$$\begin{aligned} L(V_n)(s_n) - V_n(s_n) &= \max_{a \in A_{s_n}} (r(s_n, a) + \lambda \sum_j p(j|s_n, a) V_n(j)) - V_n(s_n) \\ &= \max_{a \in A_{s_n}} (r(s_n, a) + \lambda \mathbb{E}(V_n(S_{n+1}(s_n, a)))) - V_n(s_n) \end{aligned}$$

whereas

$$\mathbb{E}[y_n] = \mathbb{E}[\max_{a \in A_{s_n}} (r(s_n, a) + \lambda V_n(S_{n+1}(s_n, a)))] - V_n(s_n)$$

and $\mathbb{E}[\max(X, Y)] \neq \max(\mathbb{E}[X], \mathbb{E}[Y])$

Q-function

We apply the R-M algorithm to the estimate the Q-function instead.

$Q(s, a)$ is the maximum expected reward starting from state s and taking action a:

$$Q(s, a) = r(s, a) + \lambda \sum_j p(j|s, a) V^\star(j)$$

Note that $V^\star(s) = \max_{a \in A_s} Q(s, a)$, and hence

$$Q(s, a) = r(s, a) + \lambda \sum_j p(j|s, a) \max_b Q(j, b)$$

Q **is the fixed point of an operator** H (defined on $\mathbb{R}^{S \times A}$)

$$(HQ)(s, a) = r(s, a) + \lambda \sum_j p(j|s, a) \max_b Q(j, b)$$

Q-function and Stochastic Approximation

- At time n, assume we have an estimate Q_n of Q. Assume that we can try all actions a and observe the new state $S_{n+1}(s_n, a)$ a r.v. such that $\mathbb{P}[S_{n+1} = j | s_n, a] = p(j|s_n, a)$. We can compute for all a

$$y_n = r(s_n, a) + \lambda \max_b Q_n(S_{n+1}(s_n, a), b)) - Q_n(s_n, a)$$

- y_n can be used in the R-M algorithm converging to Q, because

$$\begin{aligned}
\mathbb{E}[y_n] &= r(s_n, a) + \lambda \mathbb{E}[\max_b Q_n(S_{n+1}(s_n, a), b))] - Q_n(s_n, a) \\
&= r(s_n, a) + \lambda \sum_j p(j|s_n, a) \max_b Q_n(j, b) - Q_n(s_n, a) \\
&= H(Q_n)(s_n, a) - Q_n(s_n, a)
\end{aligned}$$

Q-learning

We observe the trajectory of the system under some behaviour policy π_b: $(s_n, a_n, r_n)_{n\geq 0}$, where r_n is the reward collected in the n-th step.

Parameter. Step sizes (α_n)

1. Initialization. Select a Q-function $Q_0 \in \mathbb{R}^{S\times A}$

2. Q-function improvement. For $n \geq 0$. Update the Q-function as follows: $\forall s, a$

$$Q_{n+1}(s_n, a_n) = Q_n(s_n, a_n)$$

$$+ 1_{(s,a)=(s_n,a_n)} \alpha_{\nu_n(s_n,a_n)} \left[r_n + \gamma \max_{b\in\mathcal{A}} Q_n(s_{n+1}, b) - Q_n(s_n, a_n) \right]$$

where $\nu_n(s,a) := \sum_{m=0}^{n} 1[(s,a) = (s_m, a_m)]$.

Q-learning convergence

Theorem. Assume that the step sizes (α_n) satisfy (A2), and that the sets (I_n) defined through the behaviour policy π_b satisfy (B1)-(B2). For any discount factor $\lambda \in (0,1)$:

$$\lim_{n\to\infty} Q_n = Q, \quad \text{almost surely.}$$

The conditions required in the above theorem are met if $\alpha_n = \frac{1}{n+1}$ and if the behaviour policy yields an irreducible Markov chain (e.g. unichain model). They are also met for the ϵ-greedy policy: it selects an action uniformly at random w.p. ϵ and $a_n \in \arg\max_{a\in A_{s_n}} Q_n(s_n, a)$ w.p. $1-\epsilon$.

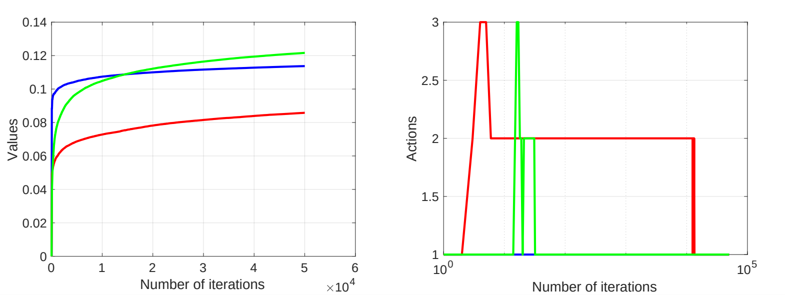
Q-learning: demo

The crawling robot ...

https://www.youtube.com/watch?v=2iNrJx6IDEo

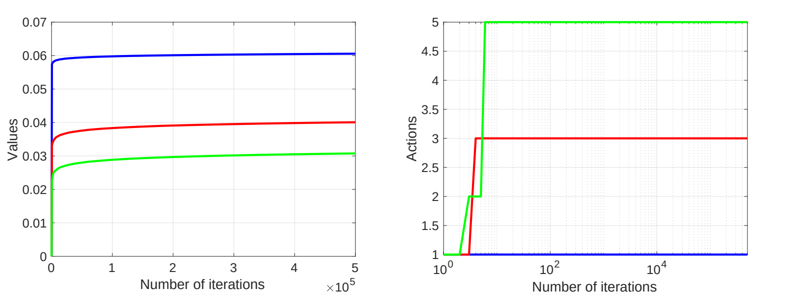
Q-learning: example

Randomly selected MDPs with 3 states and 3 actions.



Q-learning: example

Randomly selected MDPs with 3 states and 20 actions.



Lecture 4: outline

1. Q-learning and stochastic approximation
2. **SARSA (State-Action-Reward-State-Action) algorithm**
3. Improving Q-learning convergence rate

SARSA

State-Action-Reward-State-Action

- **On-policy** algorithm: we select actions according to a policy defined through Q_n
- ϵ-greedy policy:

$$\text{w.p. } 1-\epsilon, \text{select } a \in \arg\max_b Q_n(s_n, b)$$
$$\text{w.p. } \epsilon, \text{select } a \text{ uniformly at random}$$

SARSA

Parameter. Step sizes (α_n)

1. **Initialization.** Select a Q-function $Q_0 \in \mathbb{R}^{S \times A}$
2. Q-**function improvement.** For $n \geq 0$, select an action a_n according to a policy $\pi_n(Q_n)$, observe $r(s_n, a_n)$ and the next state s_{n+1}, select a_{n+1} according to $\pi_n(Q_n)$. Update the Q-function as follows: $\forall s, a$,

$$
\begin{aligned}
&Q_{n+1}(s,a) = Q_n(s,a) \\
&+ 1_{(s,a)=(s_n,a_n)} \alpha_n \left(r(s,a) + \lambda Q_n(s_{n+1}, a_{n+1}) - Q_n(s,a) \right)
\end{aligned}
$$

SARSA convergence

Theorem. Assume that the step sizes (α_n) satisfy (A2), and that SARSA is based on the ϵ-greedy policy. For any discount factor $\lambda \in (0, 1)$:

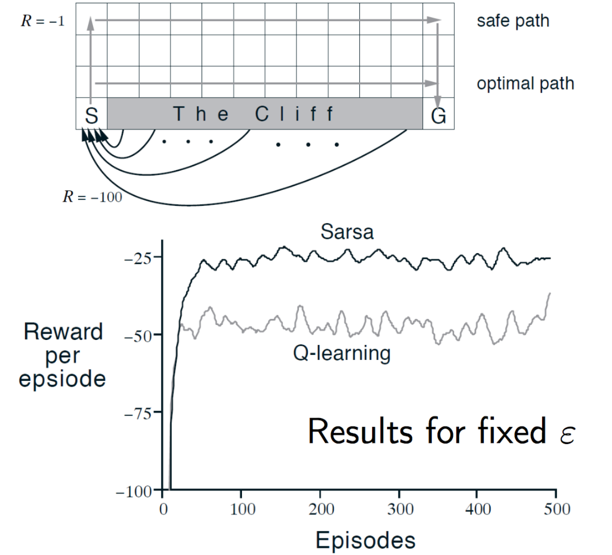
$$\lim_{n\to\infty} Q_n = Q^\epsilon, \quad \text{almost surely}$$

where $Q^\epsilon(s,a) = r(s,a) + \lambda \sum_j p(j|s,a) V^{\star\epsilon}(j)$ and $V^{\star\epsilon}$ is the value function of the policy selecting an optimal action w.p. $1-\epsilon$ and a (uniform) random action w.p. ϵ.

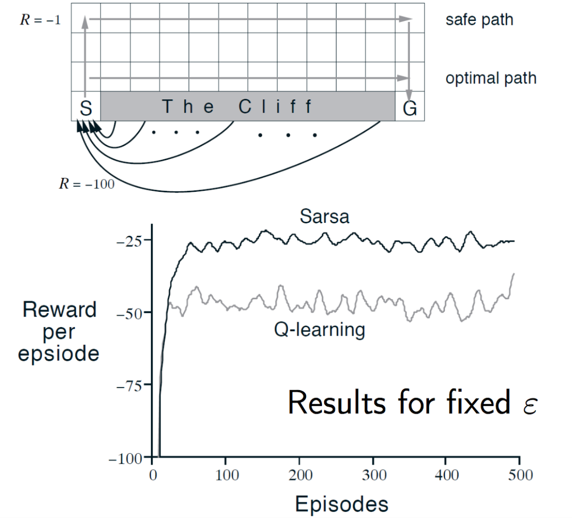
On-policy algorithms are "safer", they do not explore (state, action) pairs yielding very negative rewards (for fixed exploration rate $\epsilon > 0$, SARSA does not converge to the optimal policy)

On vs Off-policy Algorithms

R. Sutton, NIPS 2015



On vs Off-policy Algorithms



https://studywolf.wordpress.com/2013/07/01/reinforcement-learning-sarsa-vs-q-learning/

Lecture 4: outline

1. Q-learning and stochastic approximation
2. SARSA (State-Action-Reward-State-Action) algorithm
3. **Improving Q-learning convergence rate**

Slow Convergence of Q-learning

A synchronous version of Q-learning with $\alpha_k = \frac{1}{k+1}$, where every $(s, a) \in S \times A$ is observed in each round k, and

$$
\begin{aligned}
Q_{k+1} &= (1 - \alpha_k) Q_k + \alpha_k (H Q_k - \varepsilon_k) \\
&= \frac{1}{k+1} \sum_{j=0}^{k} (H Q_j - \varepsilon_j)
\end{aligned}
$$

Note that the estimation error ε_k is asymptotically averaged out, i.e., $\frac{1}{k+1} \sum_{j=0}^{k} \varepsilon_j \to 0$ a.s. as $k \to \infty$.

The convergence of recursion $Q_{k+1} = \frac{1}{k+1} \sum_{j=0}^{k} H Q_j$ is slower than $Q_{k+1} = H Q_k$ since the former is dragged down by immature $\{Q_j\}_{j<k}$.

Speedy Q-learning

Synchronous version of speedy Q-learning [Azar et al., NIPS 2011] is

$$\begin{aligned} Q_{k+1} &= \alpha_k Q_k + (1-\alpha_k)[kHQ_k - (k-1)HQ_{k-1} - \varepsilon_k] \\ &= HQ_k + \frac{1}{k+1}(HQ_{-1} - HQ_k) - \frac{1}{k+1}\sum_{j=0}^{k} \varepsilon_j \end{aligned}$$

which is asymptotically $Q_{k+1} = HQ_k$.

Asynchronous Speedy Q-learning

We observe the trajectory under behaviour policy π_b: $(s_n, a_n, r_n)_{n\geq 0}$.

Parameter. Initial Q-function $Q_0 \in \mathbb{R}^{S\times A}$

Initialization. Set $Q_{-1} = Q_0$, $k = 0$

Main loop. Repeat 1 & 2 along $(s_n, a_n, r_n)_{n\geq 0}$.

1. **Speedy Q update.** Let $\alpha_k = \frac{1}{k+1}$ and define $T_k Q(s,a) = \frac{1}{\nu_{k,n}(s,a)} \sum_{i=1}^{\nu_{k,n}(s,a)} [r_{k,i}(s,a) + \lambda \max_{b\in A} Q(s_{k,i}(s,a), b)]$ where $r_{k,i}$ and $s_{k,i}$ are the reward and next state at i-th visit of (s,a) in round k, and $\nu_{k,n}(s,a)$ is the number of visits at (s,a) up to time n in round k.

$$
\begin{aligned}
Q_{k+1}(s_n, a_n) &= (1-\alpha_k) Q_k(s_n, a_n) \\
&+ \alpha_k \left(k T_k Q_k(s_n, a_n) - (k-1) T_k Q_{k-1}(s_n, a_n)\right) .
\end{aligned}
$$

2. **Move to next round.** If all possible state-action pairs have been visited, increase k by 1.

Zap Q-learning [Devraj & Meyn NIPS 2017]

Achieving the fastest convergence among Q-learning like algorithms
Designing the optimal gain matrix G_n so that the recursion $X_{n+1} = X_n + G_n f(X_n)$ converges the fastest

Asymptotic covariance: a CTL states that $\sqrt{n}(X_n - X^\star) \sim \mathcal{N}(0, \Sigma)$.

Newton-Raphson method: How can we design G_n so as to minimise the positive semidefinite asymptotic covariance matrix Σ.

The asymptotic covariance is related to the sample complexity in some sense...

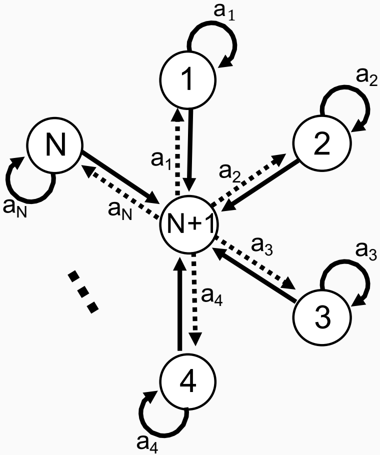
Zap Q-learning [Devraj & Meyn NIPS 2017]

Parameter. Initial Q-function $Q_0 \in \mathbb{R}^{S \times A}$ and step size α_n, β_n

1. **Initialization.** Let Q_0 be a column vector in $\mathbb{R}^{S \times A}$
2. **Zap Q update.** Let $\hat{A}_{n+1} = \hat{A}_n + \beta_{n+1}[A_{n+1} - \hat{A}_n]$ and $A_{n+1} = 1_{s_n,a_n}[\lambda 1_{s_{n+1},a^*_{n+1}} - 1_{s_n,a_n}]^\top$ where $a^*_{n+1} = \arg\max_{a \in A} Q_n(s_{n+1}, a)$ and $1_{s_n,a_n}$ is a column vector whose entry is 1 at (s_n, a_n) and 0 elsewhere. Find $\hat{A}_n^\dagger$ which is pseudo inverse of $\hat{A}_n$. Then,

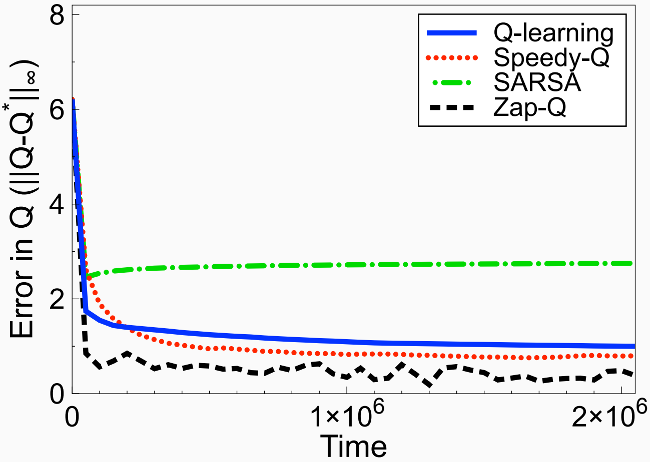
$$Q_{n+1} = Q_n - \alpha_{n+1} \hat{A}_{n+1}^\dagger \left(r_n + \lambda \max_{b \in A} Q_n(s_{n+1}, b) - Q_n(s_n, a_n) \right) 1_{s_n,a_n}$$

Numerical experiments



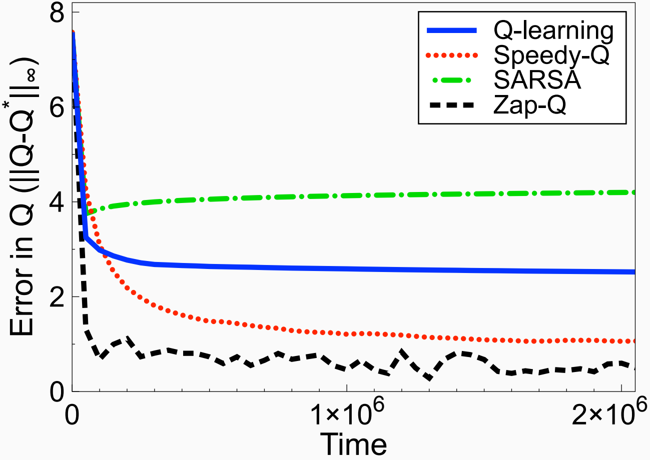
Numerical experiments

$\lambda = 0.8$



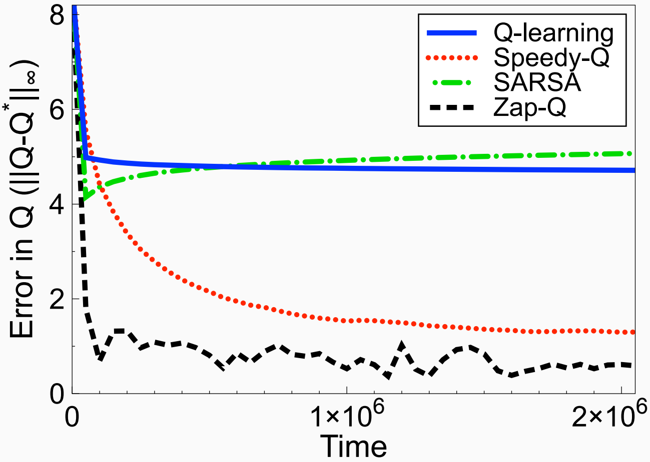
Numerical experiments

$\lambda = 0.9$



Numerical experiments

$\lambda = 0.95$



References

Stochastic approximation

- A Stochastic Approximation method. H. Robbins, S. Monro. 1951, Annals of Math. Stat., 22.
- Random Iterative Models. M. Duflo. 1997, Springer.
- Stochastic Approximation and Recursive Algorithms. H. Kushner, H. Yin. 2003, Springer.
- Stochastic Approximation: A Dynamical Systems viewpoint. V. Borkar. 2008, Cambridge Univ.

Improved Q-learning

- Speedy Q-learning. M. Azar et al., proc. of NIPS 2011.
- Zap Q-learning. A. Devraj, S. Meyn, proc. of NIPS 2017.