Part 4: Monte Carlo methods and TD learning

EL 2805 – Reinforcement Learning

Alexandre Proutiere

KTH, The Royal Institute of Technology

Objectives of this part

Model-free policy evaluation:
How can we evaluate the value function of a policy π by observing trajectories or episodes generated under π?

- Monte Carlo methods: Evaluating policies through sampling
- Robbins-Monro's stochastic approximation algorithm
- TD (Time Difference) learning: Evaluating policies through sampling and bootstrapping

Part 4: Outline

1. Monte Carlo methods
2. Stochastic approximation algorithms
3. TD learning methods

References

- Chapters 5-6-7 of Sutton-Barto's book.
- A Stochastic Approximation Method, H. Robbins, S. Monro. The Annals of Mathematical Statistics, Vol. 22, No. 3. (Sep., 1951), pp. 400-407.
- Some studies in machine learning using the game of checkers, S. Samuel. IBM Journal on Research and Development, 3(3), (1959), pp. 210-229.
- Stochastic Approximation: A dynamical systems viewpoint, V. Borkar (2008).

1. Monte Carlo methods

Discounted MDP with terminal state

- MDP: $(\lambda, S, A_s, p(\cdot|s,a), r(s,a), a \in A_s, s \in S)$.
- **Terminal state:** There is a state s_{end} after which no reward is collected.
- **Episode:** It starts at time $t = 1$ in state s_1 and finishes after a random time when s_{end} is reached.
- Assumption: under any policy, episodes finish in finite time almost surely.

On-policy evaluation using Monte Carlo methods

Let π be a deterministic stationary policy.

If one knows the MDP, evaluating the value function $V^\pi(s) = \mathbb{E}_\pi[\sum_{t=1}^{\infty} \lambda^t r_t(s_t, a_t)]$ of π is done by solving: $V^\pi(s_{end}) = 0$ and

$$\forall s \neq s_{end},\ V^\pi(s) = r(s, \pi(s)) + \lambda \sum_{j \in S} p(j|s, \pi(s)) V^\pi(j).$$

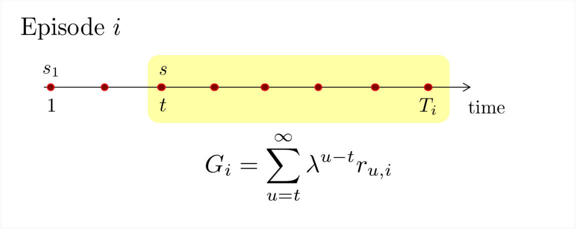
What if we do not know the MDP?

Evaluating π through sampling

The solution consists in simulating π (Monte Carlo):

- Objective: for any state s, evaluate $V^\pi(s)$
- Simulate π for n episodes:

  Episode i: $\tau_i = (s_{1,i}, a_{1,i}, r_{1,i}, \ldots, s_{T_i,i}, a_{T_i,i}, r_{T_i,i})$
  - $r_{t,i}$ is the reward received in step t in episode i
  - T_i is the length of the episode, i.e., $s_{T_i} = s_{end}$
  - If s is visited at time t in episode i, compute the return $G_i = \sum_{u \geq t} \lambda^{u-t} r_{u,i}$



Evaluating π through sampling

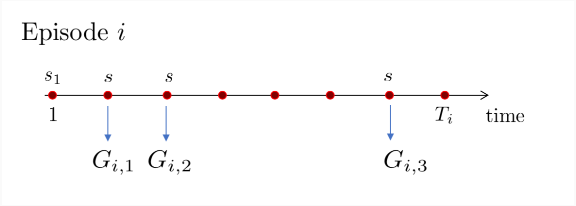
The law of large numbers yields: $\frac{1}{n}\sum_{i=1}^{n} G_i \to V^\pi(s), \;\;$ as $n \to \infty$.

Based on this observation, we can devise an algorithm estimating V^π in all states:

Monte Carlo prediction algorithm:

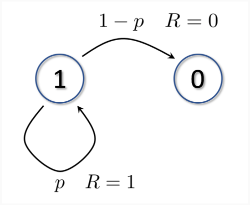
```
1. Initialization: $\forall s$, $V^{(0)}(s) = 0$
2. Iterations: for episode $i = 1, \dots, n$
   generate $\tau_i = (s_1, a_{1,i}, r_{1,i}, \dots, s_{T_i,i}, a_{T_i,i}, r_{T_i,i})$ under $\pi$
   $G = 0$
   for $t = T_i, T_i - 1, \dots, 1$:
      a. $G = r_{t,i} + \lambda G$
      b. Unless $s_{t,i}$ appears in $\{s_{1,i}, \dots, s_{t-1,i}\}$
           $V^{(i)}(s_{t,i}) = V^{(i-1)}(s_{t,i}) + \frac{1}{i}(G - V^\pi(s_{t,i}))$
```

Alternative methods?



- First-visit MC method (see previous algorithm): $\hat{V}^{\pi}(s) = \frac{1}{n}\sum_{i=1}^{n} G_{i,1}$
- Last-visit MC method: $\hat{V}^{\pi}(s) = \frac{1}{n}\sum_{i=1}^{n} G_{i,n_i(s)}$ where $n_i(s)$ is the number of visits of s in episode i
- Every-visit MC method: $\hat{V}^{\pi}(s) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n_i(s)} G_j$ where $n(s)$ is the number of visits of s, and G_j is the return of the j-th return

Alternative methods?



Two states: 0 is terminal, rewards associated to transitions, no action.
Value function: $V(0) = 0$, $V(1) = p/(1-p)$.

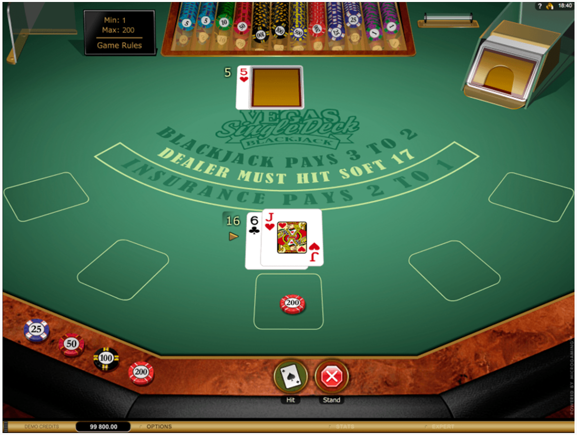
First-visit MC: $\mathbb{E}[\hat{V}(1)] = p/(1-p)$, $\text{Var}[\hat{V}(1)] = \frac{1}{n}\frac{p}{(1-p)^2}$
Last-visit MC: $\mathbb{E}[\hat{V}(1)] = 0$
Every-visit MC: $\mathbb{E}[\hat{V}(1)] = \frac{n}{n+1} \cdot \frac{p}{(1-p)}$ (there is bias)
First-visit MC is provably better than Every-visit MC for n large enough in terms of MSE=Bias^2+Var

Alternative methods?

Beware of every-visit MC estimates:

- "Samples" of $V(s)$ from the same episode are not i.i.d. (they share the same rewards and transitions)
- More complex scenarios with non-geometric episode durations: The bus paradox (every-visit MC can lead (even asymptotically) to the wrong answer.

Example: the game of blackjack[1]



[1]See Sutton-Barto's book

Example: the game of blackjack[2]

Rules: (check wikipedia)

- You against the dealer
- Card values: aces 1 or 11, face cards 10
- Initial state: you see your two initial cards (randomly chosen) and one card of the dealer
- You play first a sequence of actions "hit" or "stick". "hit" means you request a new card, "stick" means you stop
- Objective: the sum of your cards should be closer to 21 than that of the dealer's cards; when you exceed 21 "bust" you loose
- After you finish playing (stick), the dealer plays, and sticks on any sum of or higher than 17

[2]See Sutton-Barto's book

Example: the game of blackjack

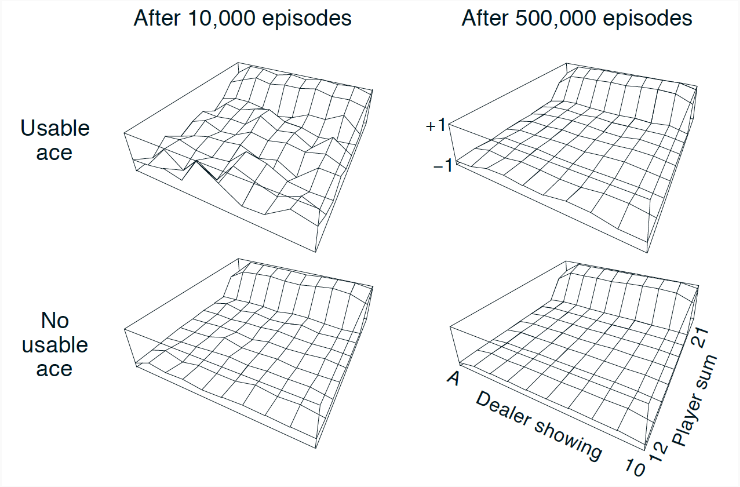
States: Your cards and dealer's card
Usable ace: if you can count it as 11 without being bust, then count it as 11
States bis: (i) usable ace or not, (ii) the sum of your cards, (iii) dealer's card

We know the MDP but transitions are complex to write down ... Hence we use RL.

Example of policy: stick only when your sum is 20 or 21.

Example: the game of blackjack

MC prediction algorithm to estimate the value of the policy sticking at 20 and 21.

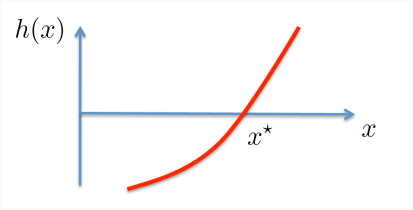


2. Robbins-Monro algorithm

Finding the root of a function:
Let $h : \mathbb{R}^d \to \mathbb{R}^d$ be a continuous function with a unique root $x^\star$ ($h(x^\star) = 0$) and with the following *monotonicity* property:

$$\exists \beta > 0 \ : \ \forall x \in \mathbb{R}^d, \quad h(x) \cdot (x - x^\star) \geq \beta \|x - x^\star\|_2^2.$$

Objective: find $x^\star$ from noisy observations of the function h.



Robbins-Monro Algorithm

If x is selected by the algorithm, one observes $Y(x)$ a r.v. bounded in magnitude by G and such that $\mathbb{E}[Y(x)] = h(x)$.

Robbins-Monro Algorithm:

1. **Initialization:** $x^{(0)}$
2. **Iterations:** for $k \geq 0$,

$$x^{(k+1)} = x^{(k)} - \alpha_k Y(x^{(k)}).$$

Convergence of the RM algorithm

Let $e_{\min}^{(k)} = \min_{i=0,\dots,k} \mathbb{E}[(x^{(i)} - x^\star)^2]$.

Theorem. *We have for all $k \geq 1$:*

$$e_{\min}^{(k)} \leq \frac{\|x^{(0)} - x^\star\|_2^2 + G^2 \sum_{i=0}^{k} \alpha_i^2}{2\beta \sum_{i=0}^{k} \alpha_i}$$

- Constant step sizes: $\alpha_i = \alpha$. The error $e_{\min}^{(k)}$ converges to a value smaller than $G^2\alpha/2\beta$ as $k \to \infty$.
- Square summable but not summable step sizes: $\sum_{i=0}^{\infty} \alpha_i = \infty$, $\sum_{i=0}^{\infty} \alpha_i^2 < \infty$. $e_{\min}^{(k)}$ converges to 0 as $k \to \infty$.

Proof

$$\begin{aligned}\|x^{(k+1)} - x^\star\|_2^2 &= \|x^{(k)} - \alpha_k Y(x^{(k)}) - x^\star\|_2^2 \\ &= \|x^{(k)} - x^\star\|_2^2 - 2\alpha_k Y(x^{(k)}) \cdot (x^{(k)} - x^\star) + \alpha_k^2 \|Y(x^{(k)})\|_2^2\end{aligned}$$

Now observe that $\mathbb{E}[Y(x^{(k)}) \cdot (x^{(k)} - x^\star) | x^{(k)}] = h(x^{(k)}) \cdot (x^{(k)} - x^\star)$. Hence, using the monotonicity of h:

$$\begin{aligned}\mathbb{E}\|x^{(k+1)} - x^\star\|_2^2 &\leq \mathbb{E}\|x^{(k)} - x^\star\|_2^2 - 2\alpha_k \mathbb{E}[h(x^{(k)}) \cdot (x^{(k)} - x^\star)] + \alpha_k^2 G^2 \\ &\leq \mathbb{E}\|x^{(k)} - x^\star\|_2^2 - 2\alpha_k \beta \mathbb{E}\|x^{(k)} - x^\star\|_2^2 + \alpha_k^2 G^2\end{aligned}$$

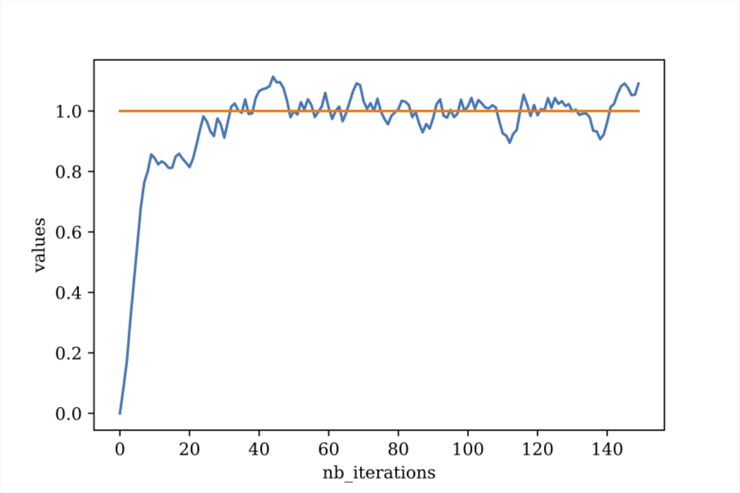
Iterating the above computation:

$$0 \leq \mathbb{E}\|x^{(k+1)} - x^\star\|_2^2 \leq \mathbb{E}\|x^{(0)} - x^\star\|_2^2 - 2\beta \sum_{i=0}^k \alpha_i \mathbb{E}\|x^{(i)} - x^\star\|_2^2 + \sum_{i=0}^k \alpha_i^2 G^2.$$

Hence: $\min_{i=0,\ldots,k} \mathbb{E}\|x^{(i)} - x^\star\|_2^2 \leq \frac{\|x^{(0)} - x^\star\|_2^2 + G^2 \sum_{i=0}^k \alpha_i^2}{2 \sum_{i=0}^k \alpha_i}$

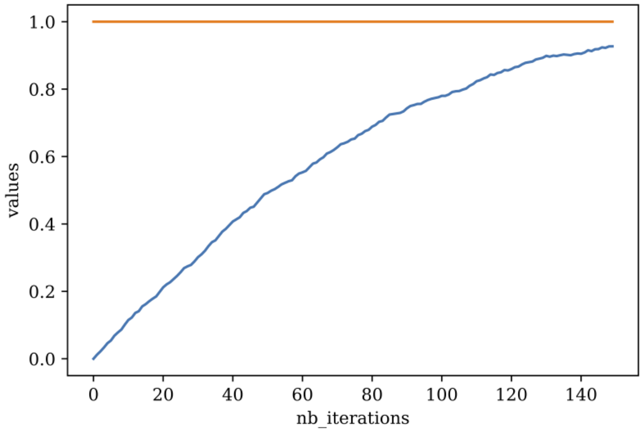
Example $h(x) = x^2 - 1$

Fixed step sizes: $\alpha_k = 0.1$



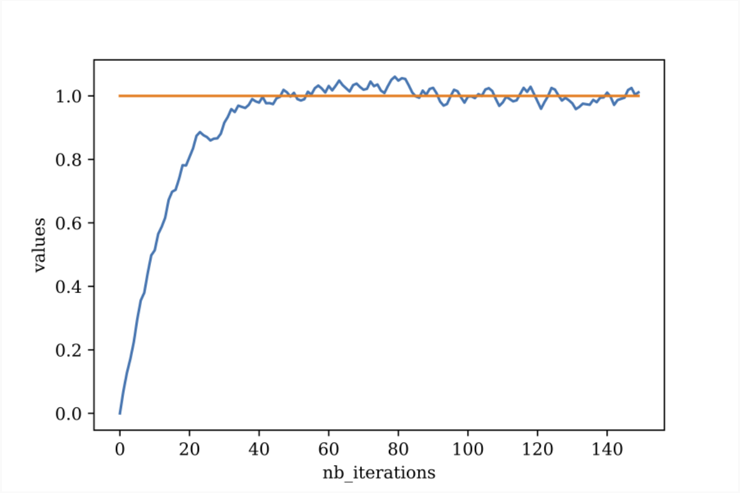
Example $h(x) = x^2 - 1$

Fixed step sizes: $\alpha_k = 0.01$



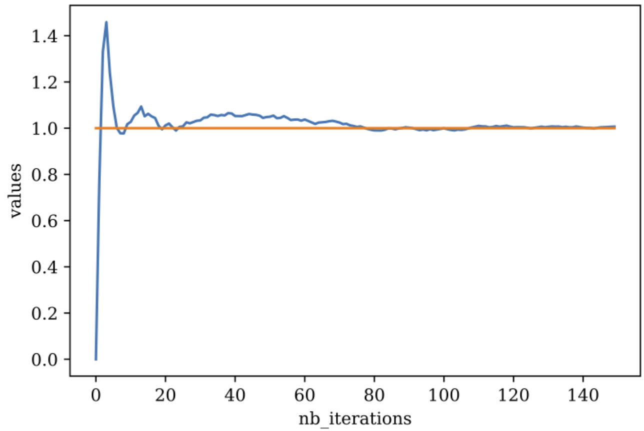
Example $h(x) = x^2 - 1$

Fixed step sizes: $\alpha_k = 0.05$



Example $h(x) = x^2 - 1$

Decreasing step sizes: $\alpha_k = 1/k$



The ODE method

At the limit, the dynamics under RM algorithm resemble that dictated by the ODE $\dot{x} = h(x)$.

Time $t_k = \sum_{i=1}^{k-1} \alpha_i$; $x(t_k) := x^{(k)}$ for $k \geq 2$. Then under RM algorithm: $x^{(k+1)} = x^{(k)} + \alpha_k(h(x^{(k)}) + M_{k+1})$ becomes

$$\underbrace{\frac{x(t_{k+1}) - x(t_k)}{t_{k+1} - t_k}}_{\to \dot{x}(t_k) \ \text{ as } k \to \infty} = h(x(t_k)) + M_{k+1}$$

A generic SA algorithm

Generic SA algorithm:

1. **Initialization:** $x^{(0)} \in \mathbb{R}^d$
2. **Iterations:** for $k \geq 0$,

$$x^{(k+1)} = x^{(k)} + \alpha_k[h(x^{(k)}) + M_{k+1}]$$

Assumptions:

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

(A2) (Diminishing step sizes) $\sum_{k=0}^{\infty} \alpha_k = \infty$ and $\sum_{k=0}^{\infty} \alpha_k^2 < \infty$.

(A3) (Martingale difference) $\forall k$, $\mathbb{E}[M_{k+1}|\mathcal{F}_k] = 0$ where $\mathcal{F}_k = \sigma(x^{(1)}, M_1, \ldots, x^{(k)}, M_k)$ and $\forall k$, $\mathbb{E}[\|M_{k+1}\|_2^2 \mid \mathcal{F}_k] \leq c_0(1 + \|x^{(k)}\|^2)$.

(A4) (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

Theorem. If (A1)-(A4) hold, for any initial condition $x^{(0)}$,

$$\lim_{k\to\infty} x^{(k)} = x^\star, \quad \text{almost surely,}$$

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

Proof. Leverage the similarities of the SA algorithm and the ODE $\dot{x} = h(x)$.

Asynchornous SA algorithm

Let $x^{(k)} = (x^{(k)}(1), ..., x^{(k)}(d))^\top \in \mathbb{R}^d$.

Asynchronous SA algorithm:

1. **Initialization:** $x^{(0)} \in \mathbb{R}^d$
2. **Iterations:** for $k \geq 0$, only a random set of coordinates $I_k \subset \{1, \ldots, d\}$ of $x^{(k)}$ are updated: for $1 \leq i \leq d$,

$$x^{(k+1)}(i) = \begin{cases} x^{(k)}(i) + \alpha_{\mathcal{I}_k(i)}[h(x^{(k)}; i) + M_{k+1}(i)] & \text{if } i \in I_k \\ x^{(k)}(i) & \text{otherwise} \end{cases}$$

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

Asynchornous SA algorithm

Assumptions:

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

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

Convergence

Asynchronous SA algorithm with decreasing step-size:

Theorem. If (A1)-(A4) and (B1)-(B2) hold, for any initial condition $x^{(0)}$,

$$\lim_{k\to\infty} x^{(k)} = x^\star, \quad \text{almost surely,}$$

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

Asynchronous SA algorithm with constant step-sizes: Convergence towards a near-optimal point as the regular SA algorithm.

3. TD learning

Limitations of the Monte Carlo method.
(i) Episodes must have finite lengths (we wait for the end of an episode to update the estimated value function)
(ii) It does not exploit the Markovian structure (it is valid for most stochastic processes)

TD (Temporal-Difference). Combines Monte Carlo and DP techniques! It exploits the Markovian structure, and is applicable in a fully on-line manner (in a single episode)

Objective

- Consider a discounted MDP: $(\lambda, S, A_s, p(\cdot|s,a), r(s,a), a \in A_s, s \in S)$
- Let π be a stationary deterministic policy
- Objective: Estimate $V^\pi = (V^\pi(s), s \in S)$ for all s from an observed trajectory generated under π:

$$(s_1, a_1, r_1, \ldots, s_t, a_t, r_t, \ldots),$$

where $r_t = r(s_t, \pi(s_t))$.

Preliminary: Incremental Value Function Iteration

If the model is known, we can proceed as follows:

Incremental Value Iteration.

1. **Initialization.** Select a value function $V^{(0)}$
2. **Value update.** For all $k \geq 1$,

$$V^{(k+1)} = V^{(k)} + \alpha_k \left(L^{\pi}(V^{(k)}) - V^{(k)} \right)$$

where the function/operator L^{π} is defined by:

$$\forall s \in S,\ L^{\pi}(V)(s) = r(s, \pi(s)) + \lambda \sum_{j} p(j|s, \pi(s)) V(j))$$

Preliminary: Incremental Value Function Iteration

Why does it work?

Preliminary: Incremental Value Function Iteration

Why does it work?

It is a SA algorithm without any noise with h defined as the linear function:

$$h(V) = L^\pi(V) - V := AV$$

h is Lipschitz and $\dot{x} = h(x)$ as unique fixed point equal to V^π. V^π is a stable fixed point because for any eigenvalue r of A, $Re(r) < 0$ ($\lambda < 1$).

Preliminary: Incremental Value Function Iteration

Why does it work?

It is a SA algorithm without any noise with h defined as the linear function:

$$h(V) = L^\pi(V) - V := AV$$

h is Lipschitz and $\dot{x} = h(x)$ as unique fixed point equal to V^π. V^π is a stable fixed point because for any eigenvalue r of A, $Re(r) < 0$ ($\lambda < 1$).

Can we make a "sampling" version of the algorithm?

TD(0) Algorithm

TD(0) algorithm

1. **Initialization.**
   Select a value function $V^{(1)}$
   Initial state s_1
   Number of visits: $\forall s$, $n_s^{(1)} = 1_{(s=s_1)}$
2. **Value function updates.** For all $t \geq 1$, select action $\pi(s_t)$ and observe the new state s_{t+1} and reward r_t.
   Update the value function estimate: for all s,

$$V^{(t+1)}(s) = V^{(t)}(s) + 1_{(s_t=s)} \alpha_{n_s^{(t)}} \left(r_t + \lambda V^{(t)}(s_{t+1}) - V^{(t)}(s) \right)$$

TD(0) Algorithm

TD(0) is an asynchronous SA algorithm:

$$\mathbb{E}[r_t + \lambda V^{(t)}(s_{t+1}) - V^{(t)}(s_t) | s_1, \ldots, s_t] = L^\pi(V^{(t)})(s_t) - V^{(t)}(s_t).$$

In theory: The almost sure convergence $V^{(t)} \to V^\pi$ as $t \to \infty$ is guaranteed if (A1)-(A4) and (B1)-(B2) holds.
Example: $\alpha_k = 1/k$, π induces an irreducible and ergodic Markov chains on S (all states are visited an infinite number of times).

In practice: A constant step size works.

TD(0) Algorithm

TD(0) is a **bootstrapping** method: The targeted value in each iteration depends on the current estimate of V^π.

TD error:

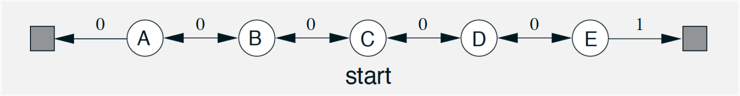
$$\varepsilon = \underbrace{r_t + \lambda V^{(t)}(s_{t+1})}_{\text{target}} - V^{(t)}(s)$$

MC error:

$$\varepsilon = \underbrace{G_t}_{\text{target}} - V^{(t)}(s)$$

Example: MC vs. TD(0)

The random walk example[3].
Episode starts in the state "start" and ends in states represented by squares.



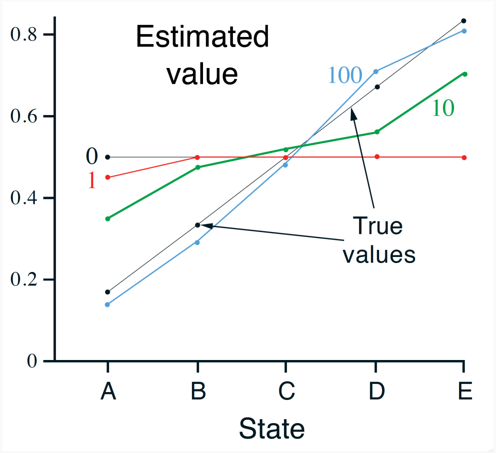
Policy π: move right and left with equal probability.
Performance metric: RMS (Root Mean-Squared)

$$RMS(t) = \sqrt{\sum_s (V^{(t)}(s) - V^{\pi}(s))^2}$$

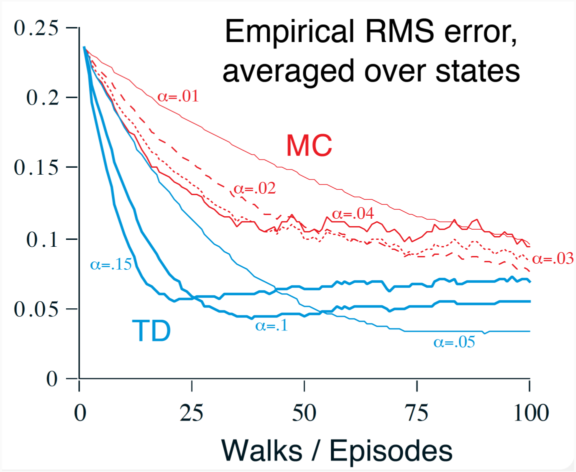
[3]See Sutton-Barto's book

Example: MC vs. TD(0)

TD(0) algorithm:



Example: MC vs. TD(0)



Summary

Model-free policy evaluation can be made using:

- Monte Carlo methods in episodic RL problems. After each episode k, if s appears in the episode:

  $$V^{(k)}(s) = V^{(k-1)}(s) + \frac{1}{k}(G_k(s) - V^{(k-1)}(s))$$

  where $G_k(s)$ is the return observed from state s in episode k.
  Convergence almost sure towards V^π.
  High variance, slow convergence.
- TD learning in episodic and discounted RL problems. After each step t, the estimated value of state s_t only is updated:

  $$V^{(t+1)}(s_t) = V^{(t)}(s_t) + \alpha_{n_{s_t}^{(t)}} \left(r_t + \lambda V^{(t)}(s_{t+1}) - V^{(t)}(s_t)\right)$$

  Convergence almost sure to V^π for decreasing step sizes or to a neighborhood of V^π in expectation for fixed step size.
  Generally better than MC methods.