Temporal-difference learning

Corrado Possieri

Machine and Reinforcement Learning in Control Applications

Introduction

- Learn directly from experience without a model of the environment.
- As DP, use learned estimates to update the prediction
  - learns from incomplete episodes;
  - bootstrap.
- Updates a guess towards a guess.
- Can be used both for prediction and control.

Monte Carlo vs temporal-difference prediction

- Given a policy π, the goal is to estimate v_π.
- MC methods wait until the return following the visit is known
  - use the return G_t as a target for $V(S_t)$

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

- The simplest TD method (TD(0)) uses the immediate reward
  - use the reward R_{t+1}
  - and the expected return $\gamma V(S_{t+1})$ as a target for $V(S_t)$

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

One-step temporal-difference

Tabular TD(0)

Input: π
Output: v_π

Initialization

$V(s) \leftarrow$ arbitrary, $\forall s \in \mathcal{S}$
$V(\text{terminal}) \leftarrow 0$

Loop

initialize S
for each step of the episode **do**
$A \leftarrow \pi(S)$
take action A and observe R, S'
$V(S) \leftarrow V(S) + \alpha(R + \gamma V(S') - V(S))$
$S \leftarrow S'$
if S is terminal **then**
reinitialize the episode

Comparison of DP, MC, and TD

- DP estimates $v_\pi(s)$ by bootstrapping

$$V(s) \leftarrow \mathbb{E}_\pi[R_{t+1} + \gamma \underbrace{V(S_{t+1})}_{\text{estimate}} | S_t = s].$$

- MC estimates $v_\pi(s)$ by sample mean

$$V(s) \leftarrow \underbrace{\mathbb{E}_\pi}_{\text{sample}}[G_t | S_t = s].$$

- TD estimates $v_\pi(s)$ by estimating both

$$V(s) \leftarrow \underbrace{\mathbb{E}_\pi}_{\text{sample}}[R_{t+1} + \gamma \underbrace{V(S_{t+1})}_{\text{estimate}} | S_t = s].$$

Temporal-difference error

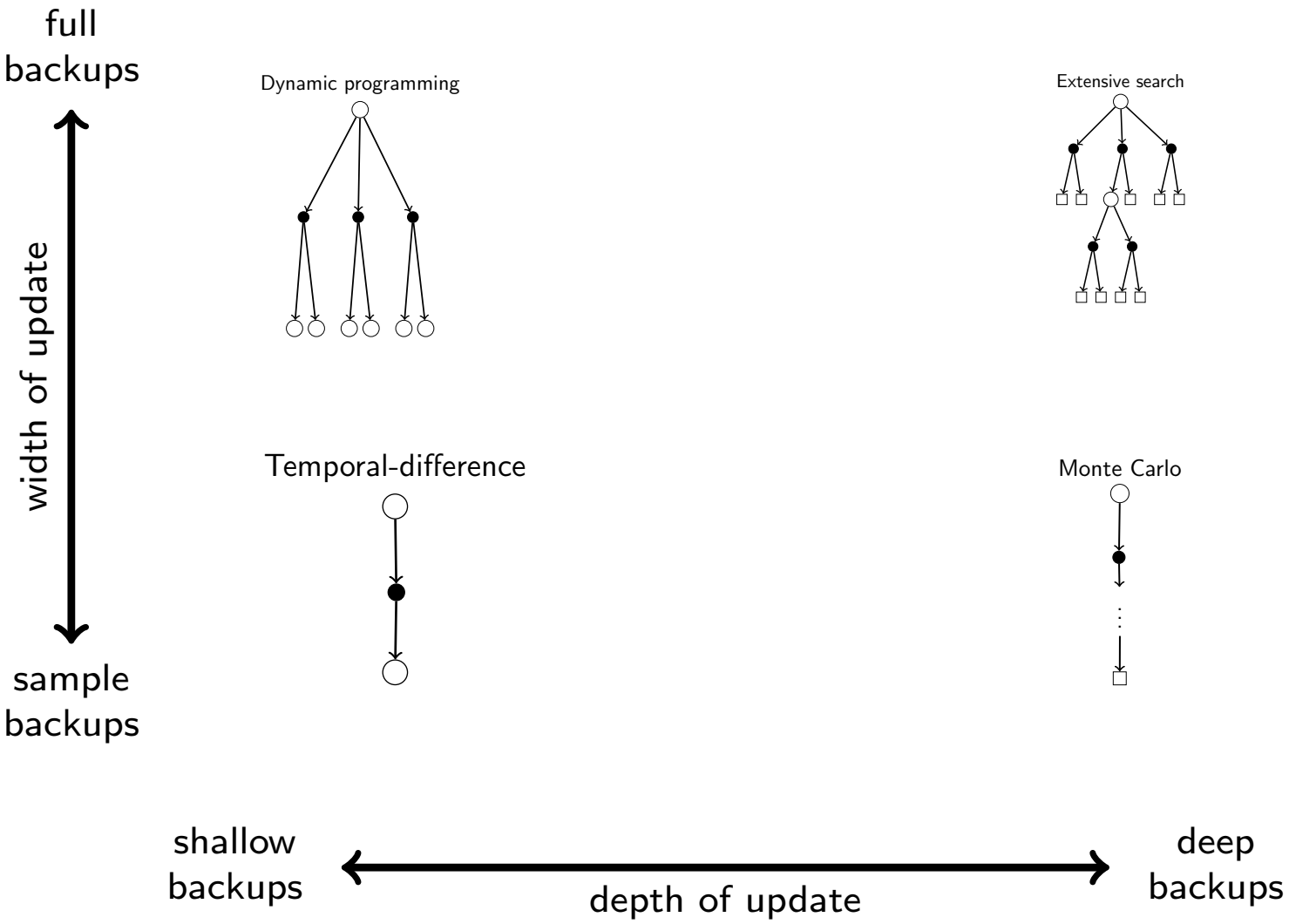
- The *TD error* is

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t).$$

- This error is used to update $V(S_t)$ towards a better estimate.
- It is not available until $t+1$.
- If V does not change, the MC error satisfies

$$\begin{aligned} G_t - V(S_t) &= R_{t+1} + \gamma G_{t+1} - V(S_t) \\ &= \delta_t + \gamma(G_{t+1} - V(S_{t+1})) \\ &= \delta_t + \gamma\delta_{t+1} + \gamma^2(G_{t+1} - V(S_{t+1})) \\ &= \sum_{k=t}^{T-1} \gamma^{k-t}\delta_k. \end{aligned}$$

Unified view

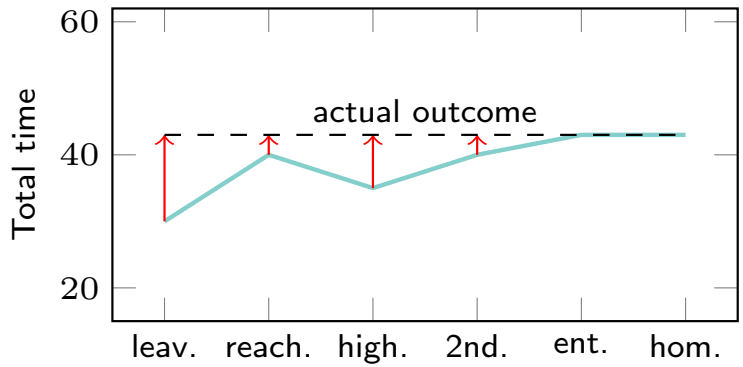


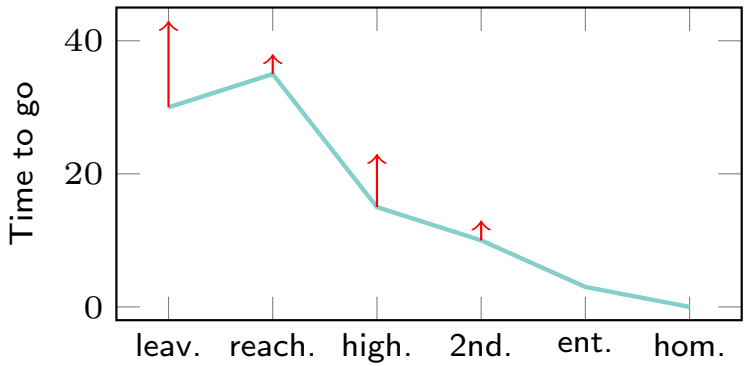
Driving home example

State	Elapsed time	Predicted time to go	Predicted total time
Leaving university	0	30	30
Reaching car	5	35	40
Exiting highway	20	15	35
Secondary road	30	10	40
Entering home	40	3	43
Home	43	0	43

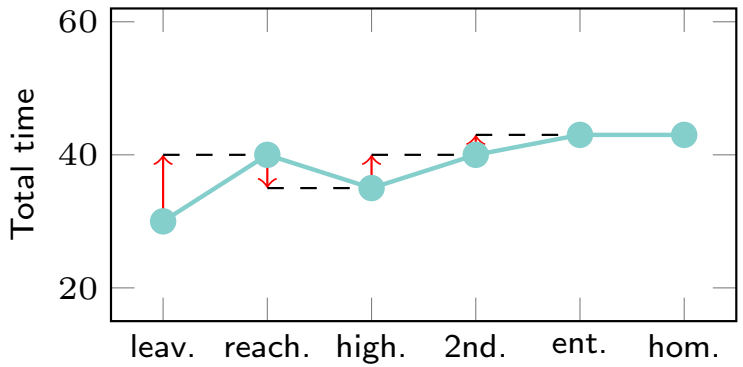
MC vs TD updates

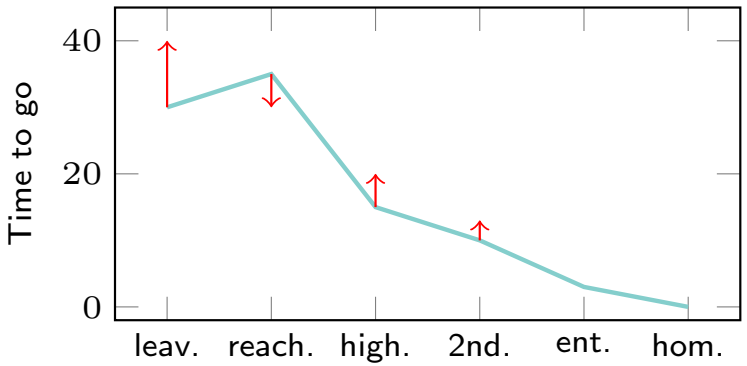
MC updates





TD updates





Advantages of TD

- TD can be implemented online.
- MC waits until the end of the episode.
- TD require just a single step.
- MC requires complete sequences.
- TD works also for incomplete sequences.
- MC can be applied just for episodic tasks.
- TD can be used for continuous and episodic tasks.

Bias and variance of estimators

- MC target

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

  is an unbiased estimate of $v_\pi(S_t)$.

- TD true target

$$R_{t+1} + \gamma v_\pi(S_{t+1})$$

  is an unbiased estimate of $v_\pi(S_t)$.

- TD target

$$R_{t+1} + \gamma V(S_{t+1})$$

  is a biased estimate of $v_\pi(S_t)$.

- TD target has lower variance than MC target
  - return depends on *many* random actions, transitions, rewards;
  - TD target depends on *one* random action, transition, reward.

Advantages and disadvantages of MC and TD

- MC has high variance and zero bias
  - good convergence properties;
  - not very sensitive to initial value;
  - simple to understand and use;
  - more efficient in non-Markov environments.

- TD has low variance and nonzero bias
  - usually more efficient than MC;
  - TD(0) proved to converge to v_π
    - in the mean for constant (small) α;
    - almost surely if $\sum_k \alpha_k = \infty$ and $\sum_k \alpha_k^2 < \infty$;
  - sensitive to initial value;
  - more efficient in Markov environments.

Batch updating

- Suppose to have a finite amount of experience.
- We can present the experience repeatedly until convergence.
- Given V, update it only once for each batch
  - compute the updates $\alpha(\text{target}_t - V(S_t))$ at each time step;
  - change the value function once with the sum of all increments.
- Constant α TD converges deterministically.
- Constant α MC converges deterministically.
- The two results may be different.

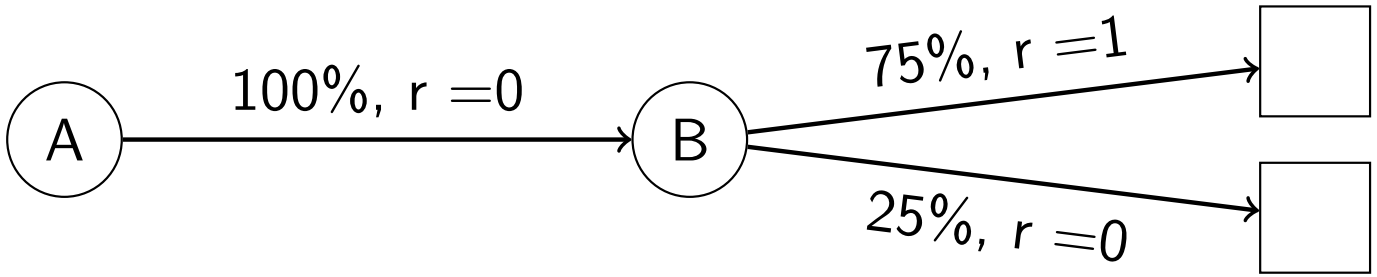
You are the predictor (1)

- Suppose you observe the following eight episodes
  1. A, 0, B, 0;
  2. B, 1;
  3. B, 1;
  4. B, 1;
  5. B, 1;
  6. B, 1;
  7. B, 1;
  8. B, 0.

- We want to estimate $V(A)$ and $V(B)$.

You are the predictor (2)

- The optimal value for B is $V(B) = \frac{6}{8} = 0.75$.
- Modeling experience via an MP



  - $V(A) = 0.75$;
  - same answer given by TD(0).

- We observed the return from A once and it was 0
  - $V(A) = 0$;
  - same answer given by MC;
  - minimum squared error on training data.

Certainty equivalence

- Batch MC converges to solution with minimum MS error
  - best fit to the observed returns

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

- Batch TD(0) converges to solution of max likelihood MDP
  - solution to the MDP that best fits the data

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

$$\hat{R}_{s,a} = \frac{1}{N(s,a)} \sum_{k=1}^{K} \sum_{t=1}^{T_k} r_t^k \mathbf{1}(s_t^k = s, a_t^k = a);$$

  - equivalent to assuming that the process estimate was known.

Off-policy TD prediction

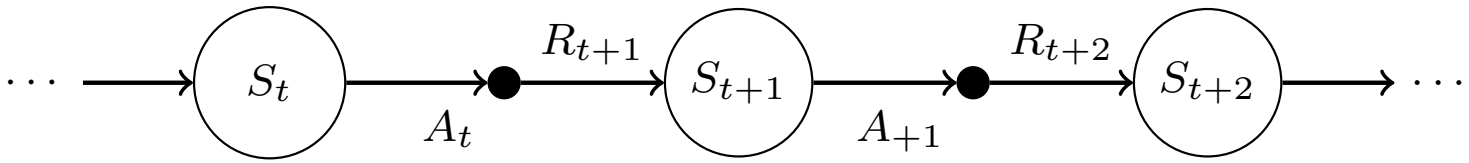
- Use TD targets generated from b to evaluate π.
- Weight TD target by importance sampling.
- Only need a single importance sampling correction

$$V(S_t) \leftarrow V(S_t) + \alpha \left(\frac{\pi(A_t|S_t)}{b(A_t|S_t)} \underbrace{(R_{t+1} + \gamma V(S_{t+1}))}_{\text{TD target}} - V(S_t) \right).$$

- Lower variance than Monte-Carlo importance sampling.

On-policy TD control

- We follow the path of GPI. Generalized policy iteration
- Use TD for the evaluation part.
- Estimate q_π rather than v_π.
- Transitions from a state–action pair to a state–action pair. funzione qualità



- This is still an MDP.

SARSA

- Apply TD(0) to action values

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \underbrace{\left(\underbrace{R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})}_{\text{TD target}} - Q(S_t, A_t)\right)}_{\text{TD error } \delta_t}.$$

- If S_{t+1} is terminal

$$Q(S_{t+1}, A_{t+1}) = 0, \quad \forall A_t.$$

- The update is carried out on the basis of the quintuple

$$(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}).$$

Greedy in the Limit with Infinite Exploration

GLIE policies

greedy limit exploration

A policy π is GLIE if

1. All state-action pairs are explored infinitely many times

necessaria per convergenza

$$N_k(s,a) \to \infty, \quad \forall a \in \mathcal{A}(s), \forall s \in \mathcal{S}.$$

2. The policy converges on a greedy policy

$$\pi(a|s) = \mathbf{1}(a = \arg\max_a Q(s,a)).$$

- For instance, ε-greedy policies are GLIE if $\varepsilon_k = \frac{1}{k}$.

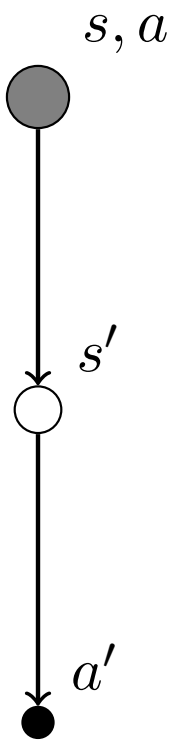
al limite

Notes on SARSA

- If Q does not change, the MC error satisfies

errore, ma aspettando la fine dell'episodio

$$\begin{aligned} G_t - Q(S_t, A_t) &= R_{t+1} + \gamma G_{t+1} - Q(S_t, A_t) \\ &= \delta_t + \gamma(G_{t+1} - Q(S_{t+1}, A_{t+1})) \\ &= \delta_t + \gamma\delta_{t+1} + \gamma^2(G_{t+2} - Q(S_{t+2}, A_{t+2})) \\ &= \sum_{k=t}^{T-1} \gamma^{k-t}\delta_k. \end{aligned}$$



- If the policy is GLIE and

$$\sum_k \alpha_k = \infty, \quad \sum_k \alpha_k^2 < \infty$$

  - SARSA converges with probability 1.

On-policy TD control algorithm

SARSA algorithm

on policy perché si sfrutta la conoscenza della q(s,a) per scegliere l'azione
non si richiede che il task sia episodico, converge sempre se il sistema è stazionario
in caso contrario conviene fissare delle costanti

Input: $\alpha > 0$, $\varepsilon > 0$
Output: q_*, π_*

Initialization

$Q(s, a) \leftarrow$ arbitrary, $\forall a \in \mathcal{A}(s), \forall s \in \mathcal{S}$
$Q(\text{terminal}, \cdot) \leftarrow 0$

Loop

initialize S
$A \leftarrow$action derived by $Q(S, \cdot)$ (e.g., ε-greedy)
for each step of the episode **do**
  take action A and observe R, S'
  $A' \leftarrow$action derived by $Q(S', \cdot)$ (e.g., ε-greedy)
  $Q(S, A) \leftarrow Q(S, A) + \alpha(R + \gamma Q(S', A') - Q(S, A))$
  $S \leftarrow S'$
  $A \leftarrow A'$
  if S is terminal **then**
    reinitialize the episode

Q-learning

- Independent of the policy being followed.
- Directly approximate q_*.
- Update Q as

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_a Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)).$$

- No importance sampling is required.

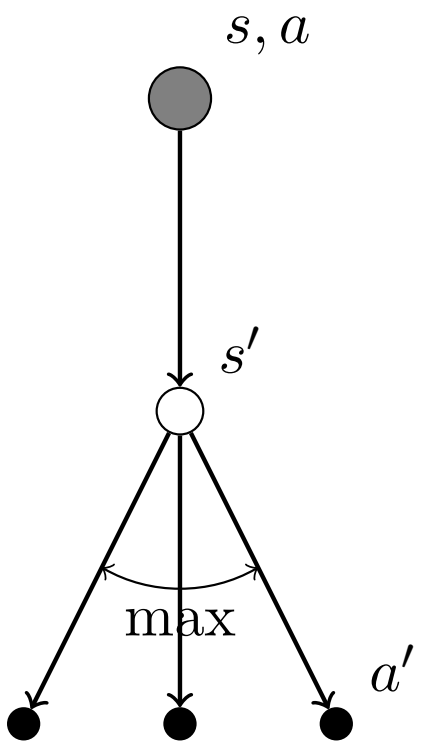
Notes on Q-learning

off-policy poiché la policy non viene scelta senza alcun criterio, converge se esploro tutte le azioni

- Next action is chosen using behavior policy, e.g., ε-greedy.
- We consider alternative successor action following the greedy target policy π.
- Both behavior and target policies improve.
- If all pairs continue to be updated and

$$\sum_k \alpha_k(x_k, a_k) = \infty, \quad \sum_k \alpha_k^2(x_k, a_k) < \infty$$

  - Q-learning converges with probability 1.



Off-policy TD control algorithm

Q-learning algorithm

on policy

Input: $\alpha > 0$, $\varepsilon > 0$
Output: q_*, π_*

Initialization

$Q(s, a) \leftarrow$ arbitrary, $\forall a \in \mathcal{A}(s), \forall s \in \mathcal{S}$
$Q(\text{terminal}, \cdot) \leftarrow 0$

secondo qualsiasi legge comportamentale purché si mantenga l'esplorazione

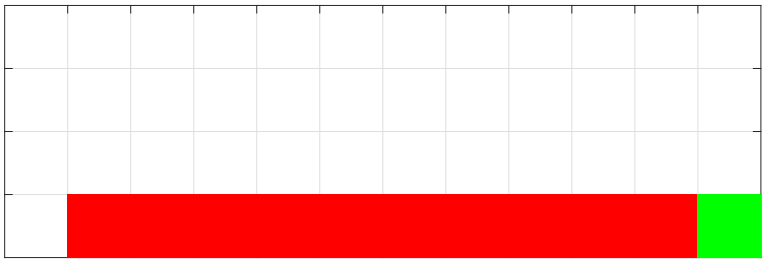
Loop

initialize S
for each step of the episode **do**
  $A \leftarrow$action derived by $Q(S, \cdot)$ (e.g., ε-greedy)
  take action A and observe R, S'
  $Q(S, A) \leftarrow Q(S, A) + \alpha(R + \gamma \max_a \gamma Q(S', a) - Q(S, A))$
  $S \leftarrow S'$
  if S is terminal **then**
    reinitialize the episode

aggiornamento funzione qualità ottima

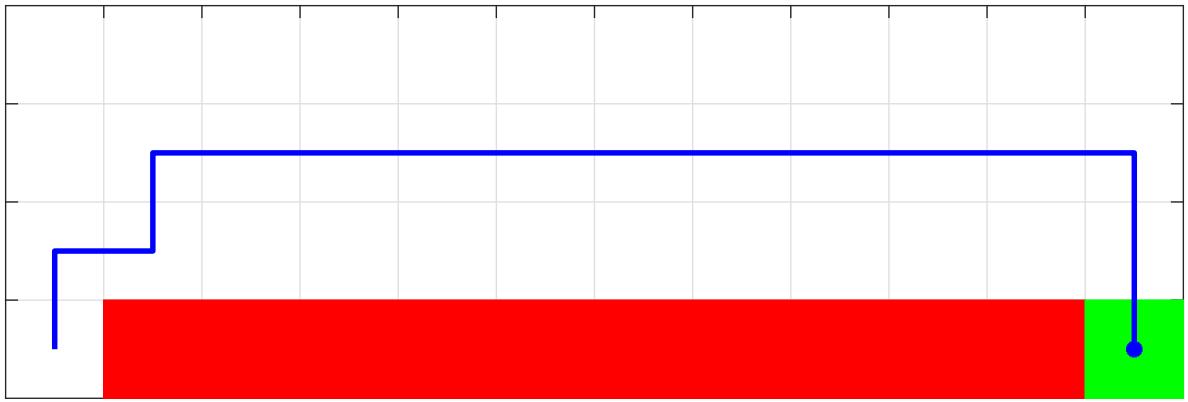
Cliff walking

- Consider the gridworld on the right.
- Terminal states are those in red and green.
- Reward is -1 on all transitions except those into the red area.
- Stepping into this region incurs a reward of -100.

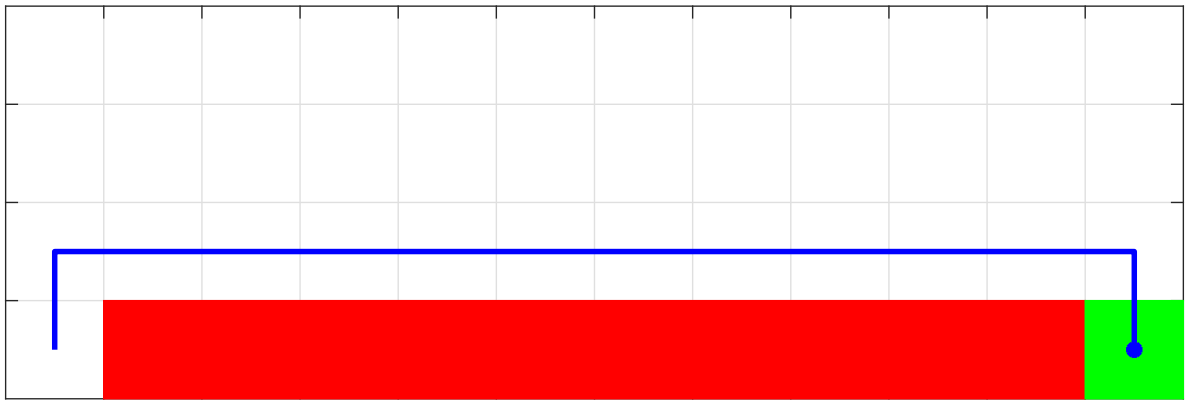


Comparison of SARSA and Q-learning on cliff walking

SARSA - episode 1000



Q-learning - episode 1000



Expected SARSA

vogliamo migliorare SARSA prendendo il valore atteso invece che il massimo
invece di scegliere l'azion successiva al'istante corrente, togliamo quel passo e
prendiamo il valore atteso delle azioni che potrei prendere.
Quindi si riduce la varianza poiché vedo la probabilità di scegliere tutte le azioni.

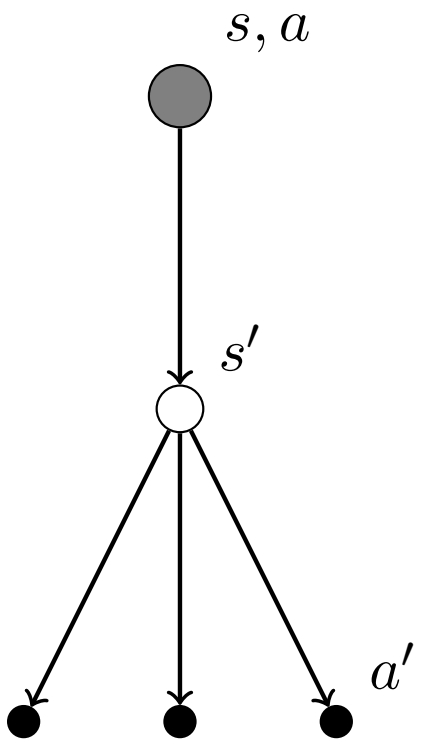
- Consider the basic update of Q-learning.
- Rather than maximizing, take expectation

$$\begin{aligned} Q(S_t, A_t) &\leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \mathbb{E}_\pi[Q(S_{t+1}, A_{t+1})|S_t] - Q(S_t, A_t)) \\ &= Q(S_t, A_t) + \alpha \left(R_{t+1} + \gamma \left(\sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a) \right) - Q(S_t, A_t) \right). \end{aligned}$$

- This moves deterministically as SARSA moves in expectation.
- It eliminates the variance due to the random selection of A_{t+1}.

Notes on expected SARSA

- Retains advantages of SARSA. è ancora un algoritmo on-policy
- Consistent empirical advantage of expected SARSA over SARSA.
- Can select large values of α ($\simeq 1$) when the environment is non-stochastic.
- Can be used off-policy poiché
  - π is the greedy policy
  - b is the behavior policy
    - expected SARSA is Q-learning.



Maximization bias

- All control algorithms discussed so far involve maximization
  - in Q-learning the target policy is the greedy policy on Q;
  - in SARSA in Sarsa the policy is often ε-greedy.
- A maximization over estimates may lead to a (positive) bias.
- Consider, e.g., the case
  - $q(s, a) = 0$ for all $a \in \mathcal{A}(s)$;
  - $Q(s, a)$ are uncertain and thus distributed some above and some below zero.
- The problem is due to the fact that the same data are used to estimate both optimal actions and their values.

Double learning

- Divide data in two sets and used them to learn two independent estimates: Q_1 and Q_2.
- Q_1 can be used to determine the maximizing action

$$A_t^* = \arg\max_a Q(S_t, a).$$

- Q_2 can be used to estimate its value

$$Q_2(S_t, A_t^*) = Q_2(S_t, \arg\max_a Q(S_t, a)).$$

- This is equivalent to

$$Q_2(s,a) \leftarrow Q_2(s,a) + \alpha(R + \gamma Q_1(s', \arg\max_{a'} Q_2(s', a')) - Q_2(S, A)).$$

- The estimate will be unbiased

$$\mathbb{E}[Q_2(S_t, A_t^*)] = q(S_t, A_t^*).$$

Double Q-learning

Double Q-learning algorithm

Input: $\alpha > 0$, $\varepsilon > 0$
Output: q_*, π_*

Initialization

$Q_1(s,a) \leftarrow$ arbitrary, $\forall a \in \mathcal{A}(s), \forall s \in \mathcal{S}$, $Q_1(\text{terminal}, \cdot) \leftarrow 0$
$Q_2(s,a) \leftarrow$ arbitrary, $\forall a \in \mathcal{A}(s), \forall s \in \mathcal{S}$, $Q_2(\text{terminal}, \cdot) \leftarrow 0$

Loop

initialize S
for each step of the episode **do**
$A \leftarrow$action derived by $Q_1(S,\cdot) + Q_2(S,\cdot)$ (e.g., ε-greedy)
take action A and observe R, S'
with probability 0.5
$Q_1(S,A) \leftarrow Q_1(S,A) + \alpha(R + \gamma Q_2(S', \arg\max_a Q_1(S',a)) - Q_1(S,A))$
else
$Q_2(S,A) \leftarrow Q_2(S,A) + \alpha(R + \gamma Q_1(S', \arg\max_a Q_2(S',a)) - Q_2(S,A))$
$S \leftarrow S'$
if S is terminal **then**
reinitialize the episode