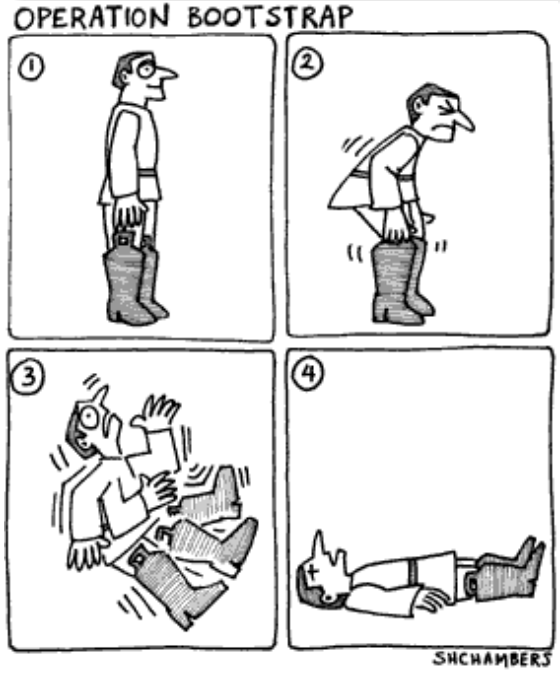
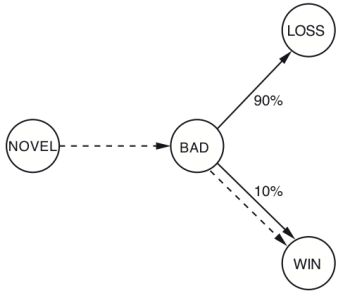
Reinforcement Learning

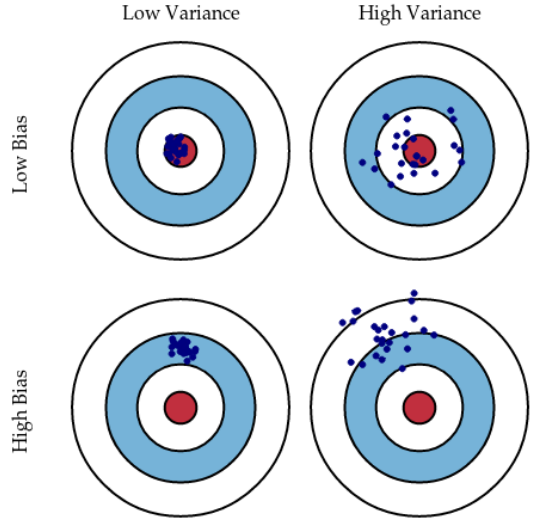
RL in finite MDPs



Marcello Restelli

February, 2024





RL techniques

Marcello Restelli

- Model–free vs Model–based
- On–policy vs Off–policy
- Online vs Offline
- Tabular vs Function Approximation
- Value–based vs Policy–based vs Actor–Critic

RL problems

Marcello Restelli

- **Model–free Prediction**: Estimate the value function of an **unknown** MRP (MDP + policy)
- **Model-free Control**: Optimize the value function of an **unknown** MDP

Monte–Carlo Reinforcement Learning

Marcello Restelli

- MC methods learn **directly** from episodes of **experience**
- MC is **model–free**: no knowledge of MDP transitions/rewards
- MC learns from **complete** episodes: no bootstrapping
- MC uses the simplest possible idea: **value = mean return**
- Caveat: can only apply MC to **episodic** MDPs
  - All episodes **must terminate**

Monte Carlo for Prediction and Control

Marcello Restelli

- MC can be used for **prediction**:
  - **Input**: Episodes of experience $\{s_1, a_1, r_2, \ldots, s_T\}$ generated by following policy π in given MDP
  - or: Episodes of experience $\{s_1, a_1, r_2, \ldots, s_T\}$ generated by MRP
  - **Output**: Value function V^π
- Or for **control**:
  - **Input**: Episodes of experience $\{s_1, a_1, r_2, \ldots, s_T\}$ in given MDP
  - **Output**: Optimal value function V^*
  - **Output**: Optimal policy π^*

Estimation of Mean: Monte Carlo

Marcello Restelli

- Let X be a random variable with mean $\mu = \mathbb{E}[x]$ and variance $\sigma^2 = Var[X]$. Let $x_i \sim X$, $i = 1, \ldots, n$ be n i.i.d. realizations of X.
- **Empirical mean** of X:

$$\hat{\mu}_n = \frac{1}{n} \sum_{i=1}^{n} x_i$$

- We have $\mathbb{E}[\hat{\mu}_n] = \mu$, $\text{Var}[\hat{\mu}_n] = \frac{\text{Var}[x]}{n}$
  - **Weak law of large numbers**: $\hat{\mu}_n \xrightarrow{P} \mu$ $(\lim_{n\to\infty} \mathbb{P}(|\hat{\mu}_n - \mu| > \epsilon) = 0)$
  - **Strong law of large numbers**: $\hat{\mu}_n \xrightarrow{a.s.} \mu$ $(\mathbb{P}(\lim_{n\to\infty} \hat{\mu}_n = \mu) = 1)$
  - **Central limit theorem**: $\sqrt{n}(\hat{\mu}_n - \mu) \xrightarrow{D} \mathcal{N}(0, \text{Var}[x])$

Monte–Carlo Policy Evaluation

Marcello Restelli

- **Goal**: learn V^π from experience under policy π

$$s_1, a_1, r_2, \ldots, s_T \sim \pi$$

- Recall that the **return** is the total discounted reward:

$$v_t = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{T-1} r_{t+T}$$

- Recall that the **value function** is the expected return:

$$V^\pi(s) = \mathbb{E}[v_t | s_t = s]$$

- Monte Carlo policy evaluation uses **empirical mean** return instead of expected return
  - **first visit**: average returns only for the first time s is visited (**unbiased** estimator)
  - **every visit**: average returns for every time s is visited (**biased** but **consistent** estimator)

First–Visit Monte–Carlo Policy Evaluation

Initialize:
$\pi \leftarrow$ policy to be evaluated
$V \leftarrow$ an arbitrary state–value function
$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

loop
  Generate an episode using π
  for each state s in the episode **do**
    $R \leftarrow$ return following the first occurrence of s
    Append R to $Returns(s)$
    $V(s) \leftarrow$ average$(Returns(s))$
  end for
end loop

Every–Visit Monte–Carlo Policy Evaluation

Marcello Restelli

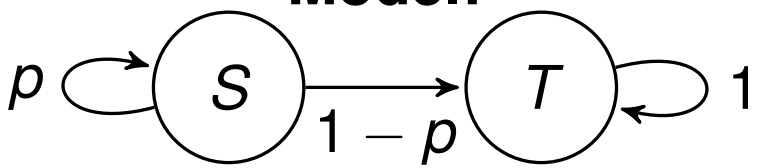
Initialize:
$\pi \leftarrow$ policy to be evaluated
$V \leftarrow$ an arbitrary state–value function
$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

loop
  Generate an episode using π
  for each state s in the episode **do**
    for each occurrence of state s in the episode **do**
      $R \leftarrow$ return following this occurrence of s
      Append R to $Returns(s)$
      $V(s) \leftarrow$ average($Returns(s)$)
    end for
  end for
end loop

First–Visit vs Every–Visit

Marcello Restelli

Model:

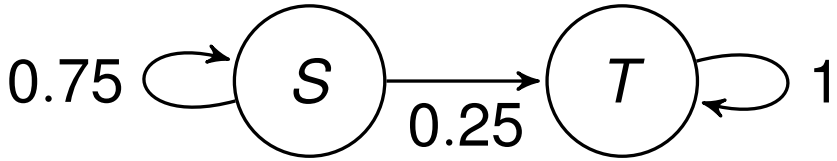


Reward is $+1$ on every step

Sample Path: $S \to S \to S \to S \to T$

$$V^{FV}(S) = 4 \quad V^{EV}(S) = 2.5$$

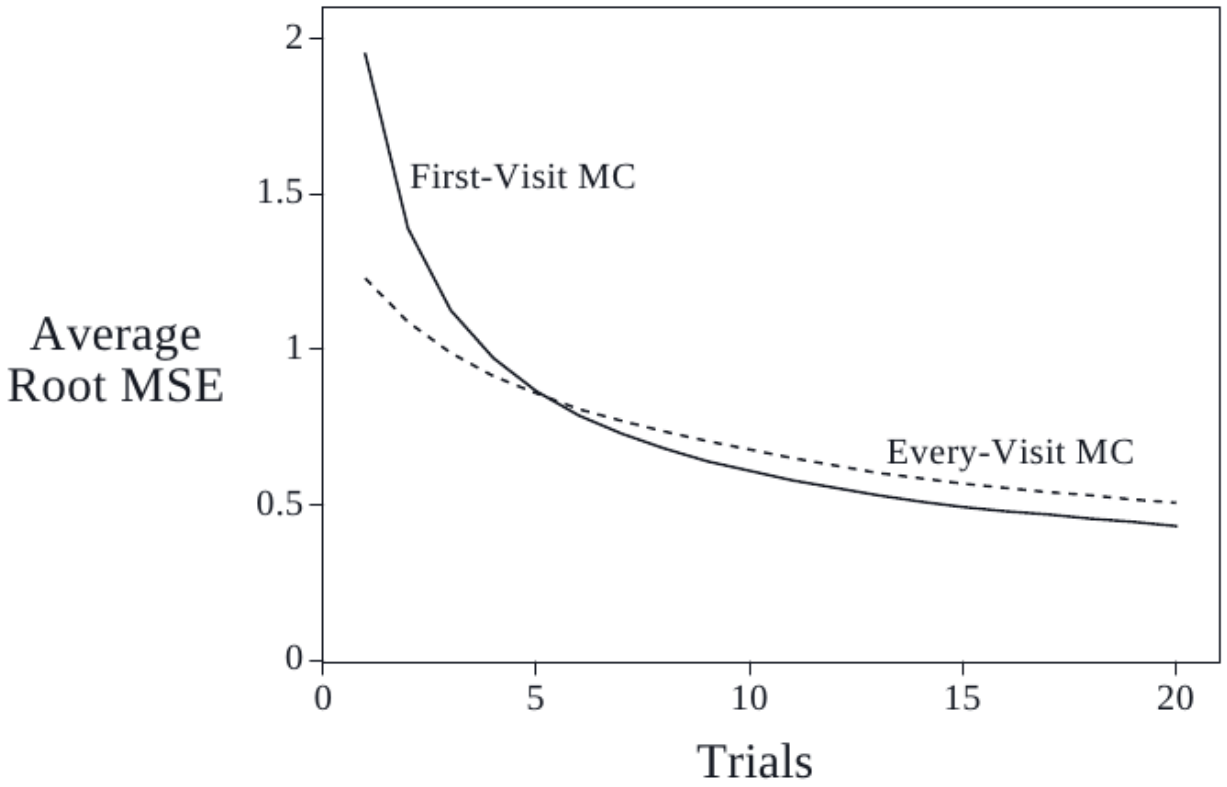
Max Likelihood Model:



First–Visit vs Every–Visit

Crossover

Marcello Restelli



Blackjack Example

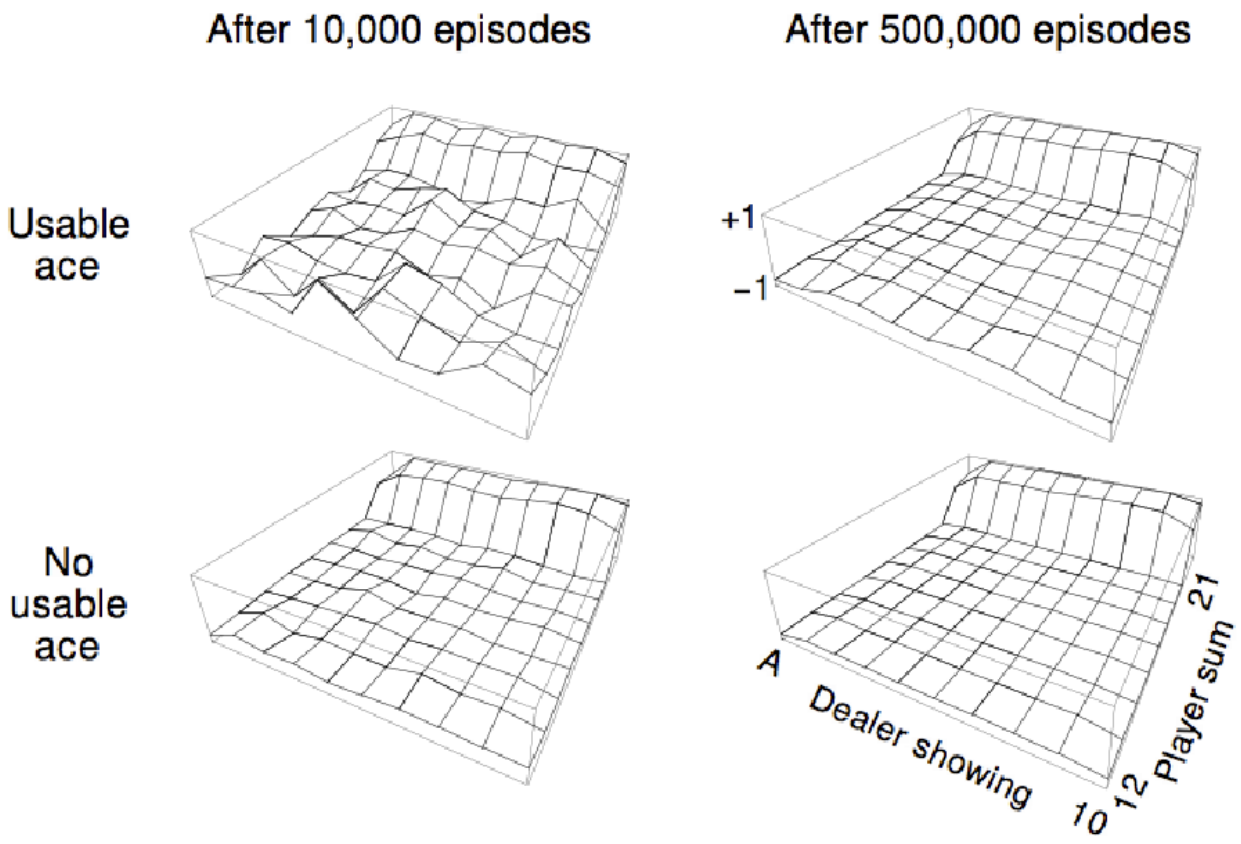
Marcello Restelli

- **Goal**: Have your card sum be greater than the dealers without exceeding 21
- **States** (200 of them):
  - current sum (12–21)
  - dealer's showing card (ace–10)
  - do I have a usable ace?
- **Reward**: +1 for winning, 0 for a draw, -1 for losing
- **Actions**: stand (stop receiving cards), hit (receive another card)
- **Policy**: Stand if my sum is 20 or 21, else hit

Blackjack Example

After Monte–Carlo Learning

Marcello Restelli



Incremental Mean

Marcello Restelli

The mean $\hat{\mu}_1, \hat{\mu}_2, \ldots$ of a sequence $x_1, x_2, \ldots$ can be computed incrementally

$$\begin{aligned}
\hat{\mu}_k &= \frac{1}{k} \sum_{j=1}^{k} x_j \\
&= \frac{1}{k} \left(x_k + \sum_{j=1}^{k-1} x_j \right) \\
&= \frac{1}{k} \left(x_k + (k-1)\hat{\mu}_{k-1} \right) \\
&= \hat{\mu}_{k-1} + \frac{1}{k} \left(x_k - \hat{\mu}_{k-1} \right)
\end{aligned}$$

Incremental Monte–Carlo Updates

Marcello Restelli

- Update $V(s)$ **incrementally** after episode $s_1, a_1, r_2, \ldots, s_T$
- For each state s_t with return v_t

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

$$V(s_t) \leftarrow V(s_t) + \frac{1}{N(s_t)}(v_t - V(s_t))$$

- In **non–stationary** problems, it is useful to track a running mean, i.e., **forget** old episodes

$$V(s_t) \leftarrow V(s_t) + \alpha(v_t - V(s_t))$$

Marcello Restelli

Let X be a random variable in $[0, 1]$ with mean $\mu = \mathbb{E}[X]$. Let $x_i \sim X, i = 1, \ldots, n$ be n i.i.d. realizations of X.
Consider the estimator (**exponential average**)

$$\mu_i = (1 - \alpha_i)\mu_{i-1} + \alpha_i x_i,$$

with $\mu_1 = x_1$ and α_i's are **step–size parameters** or **learning rates**

Proposition

If $\sum_{i \geq 0} \alpha_i = \infty$ and $\sum_{i \geq 0} \alpha_i^2 < \infty$, then $\hat{\mu}_n \xrightarrow{a.s.} \mu$, i.e., the estimator $\hat{\mu}_n$ is **consistent**

Note: The step sizes $\alpha_i = \frac{1}{i}$ satisfy the above conditions. In this case, the exponential average gives us the empirical mean $\hat{\mu}_n = \frac{1}{n} \sum_{i=1}^{n} x_i$, which is **consistent** according to the **strong law of large numbers**

Monte–Carlo Backups

Marcello Restelli

- **Entire episode** included
- Only **one choice** at each state (unlike DP)
- MC **does not bootstrap**
- Time required to estimate one state **does not depend** on the total number of states

Temporal Difference Learning

Marcello Restelli

- TD methods **learn directly** from episodes of experience
- TD is **model–free**: no knowledge of MDP transitions/rewards
- TD learns from **incomplete** episodes: **bootstrapping**
- TD updates a **guess** towards a **guess**

TD Prediction

- **Goal**: learn V^π online from experience under policy π
- **Recall**: incremental every–visit Monte Carlo

$$V(s_t) \leftarrow V(s_t) + \alpha(v_t - V(s_t))$$

- **Simplest** temporal–difference learning algorithm: TD(0)
  - Update value $V(s_t)$ towards **estimated return** $r_{t+1} + \gamma V(s_{t+1})$

$$V(s_t) \leftarrow V(s_t) + \alpha(r_{t+1} + \gamma V(s_{t+1}) - V(s_t))$$

  - $r_{t+1} + \gamma V(s_{t+1})$ is called the **TD target**
  - $\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$ is called the **TD error**

Bias–Variance Trade–Off

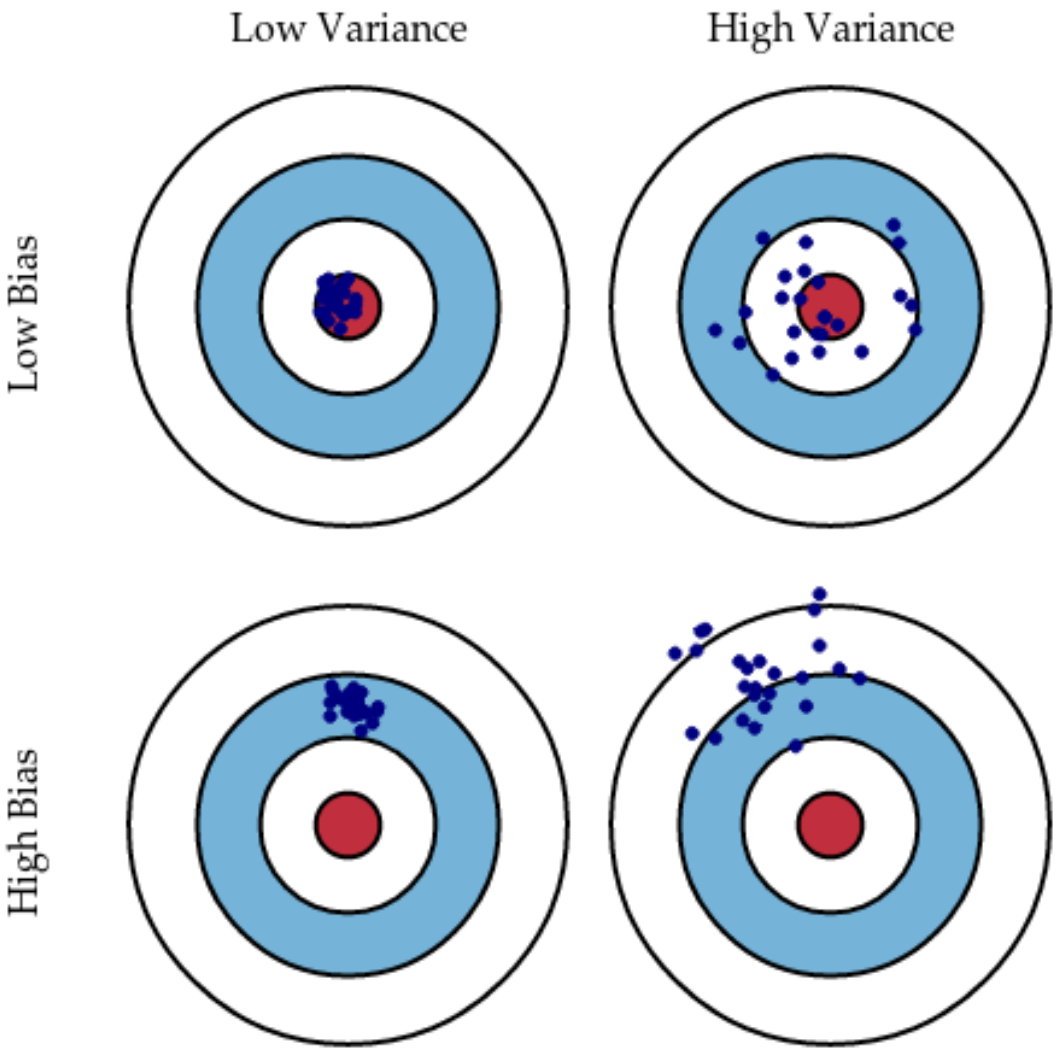
Conceptual Definition

- **Error due to bias**: is the difference between the expected prediction of our model and the actual value we want to predict
- **Error due to variance**: is the variability of a model prediction for a given data point

Bias–Variance Trade–Off

Graphical Definition

Marcello Restelli



Bias–Variance Trade–Off

Mathematical Definition

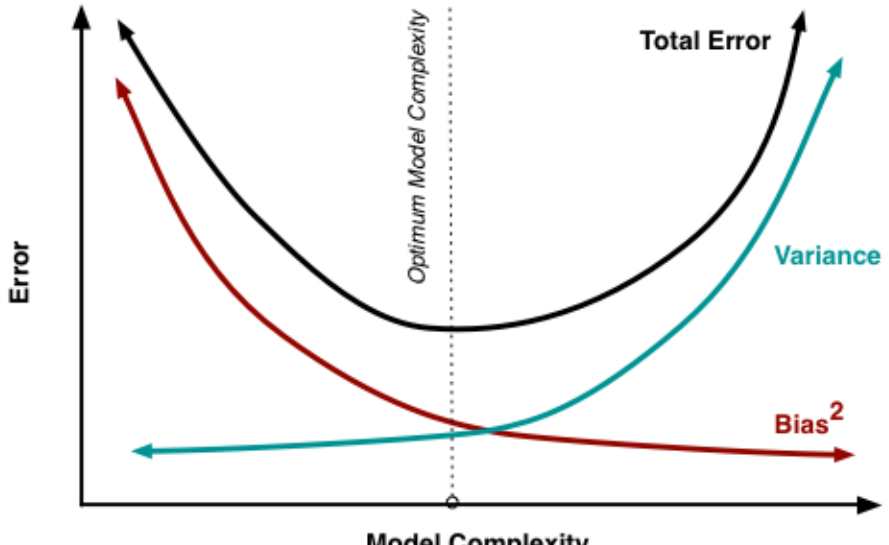
- We want to **predict** $Y = f(X) + \epsilon$, where the error term ϵ is normally distributed $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon)$
- We estimate a model $\hat{f}(X)$, with the following **prediction error** at point x:

$$Err(x) = \mathbb{E}[(Y - \hat{f}(x))^2]$$

- This error can be decomposed into **bias** and **variance**:

$$Err(x) = \left(f(x) - \mathbb{E}[\hat{f}(x)]\right)^2 + \mathbb{E}\left[\mathbb{E}[\hat{f}(x)] - \hat{f}(x)\right]^2 + \sigma_\epsilon^2$$

$$Err(x) = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$



Bias–Variance Trade–Off

MC vs TD

Marcello Restelli

- Return $v_t = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{T-1} r_{t+T}$ is an **unbiased** esitmate of $V^\pi(s_t)$
- TD target $r_{t+1} + \gamma V(s_{t+1})$ is a **biased** estimate of $V^\pi(s_t)$
  - Unless $V(s_{t+1}) = V^\pi(s_{t+1})$
- But the TD target is much **lower variance**:
  - Return depends on **many** random actions, transitions, rewards
  - TD target depends on **one** random action, transition, reward

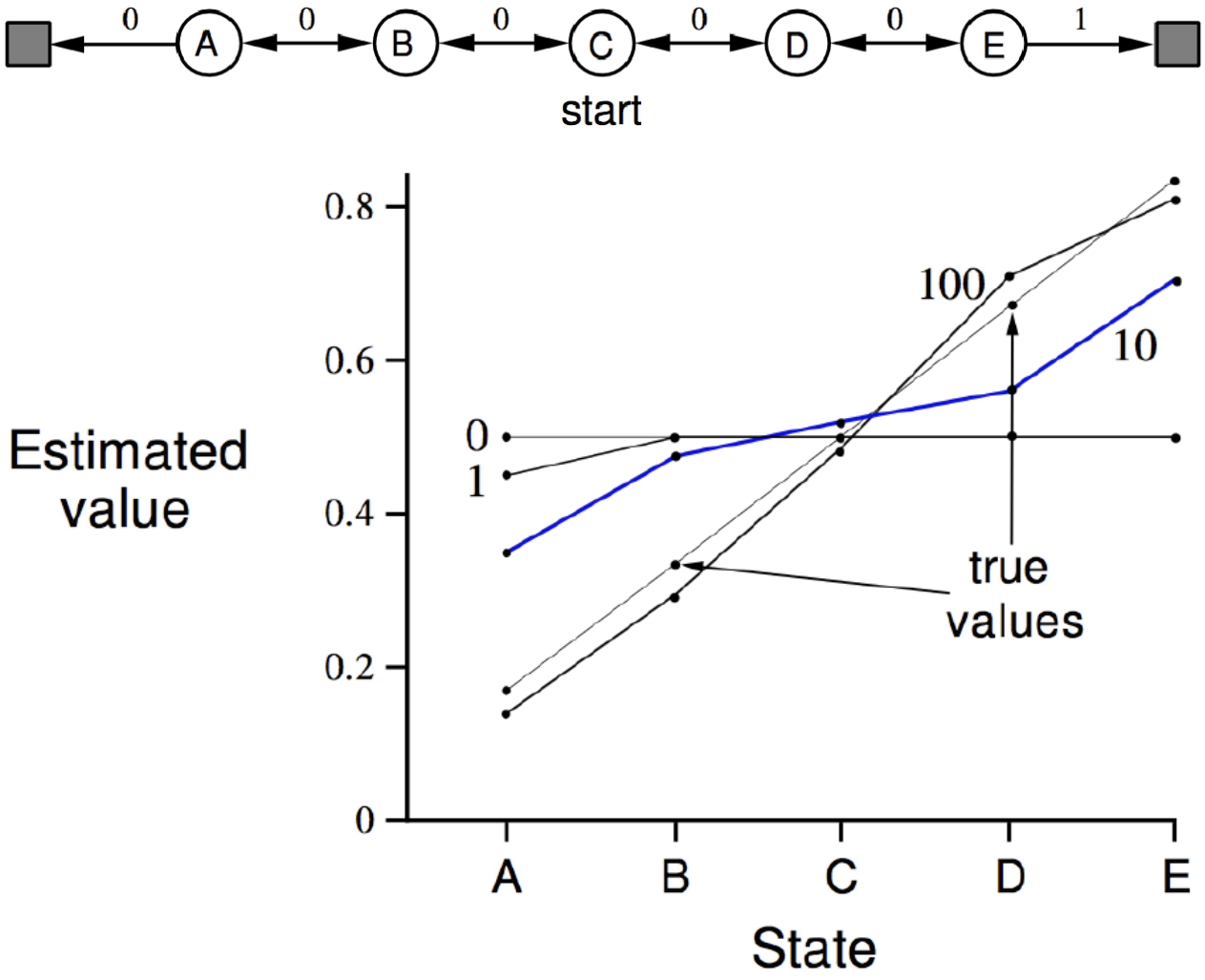
Bias–Variance comparison between MC and TD

Marcello Restelli

- MC has high variance, zero bias
  - **Good convergence** properties
  - Works well with **function approximation**
  - **Not** very sensitive to **initial value**
  - Very **simple** to understand and use
- TD has low variance, some bias
  - Usually **more efficient** than MC
  - TD(0) converges to $V^{\pi}(s)$
  - **Problem** with function approximation
  - More **sensitive** to initial values

Random Walk Example

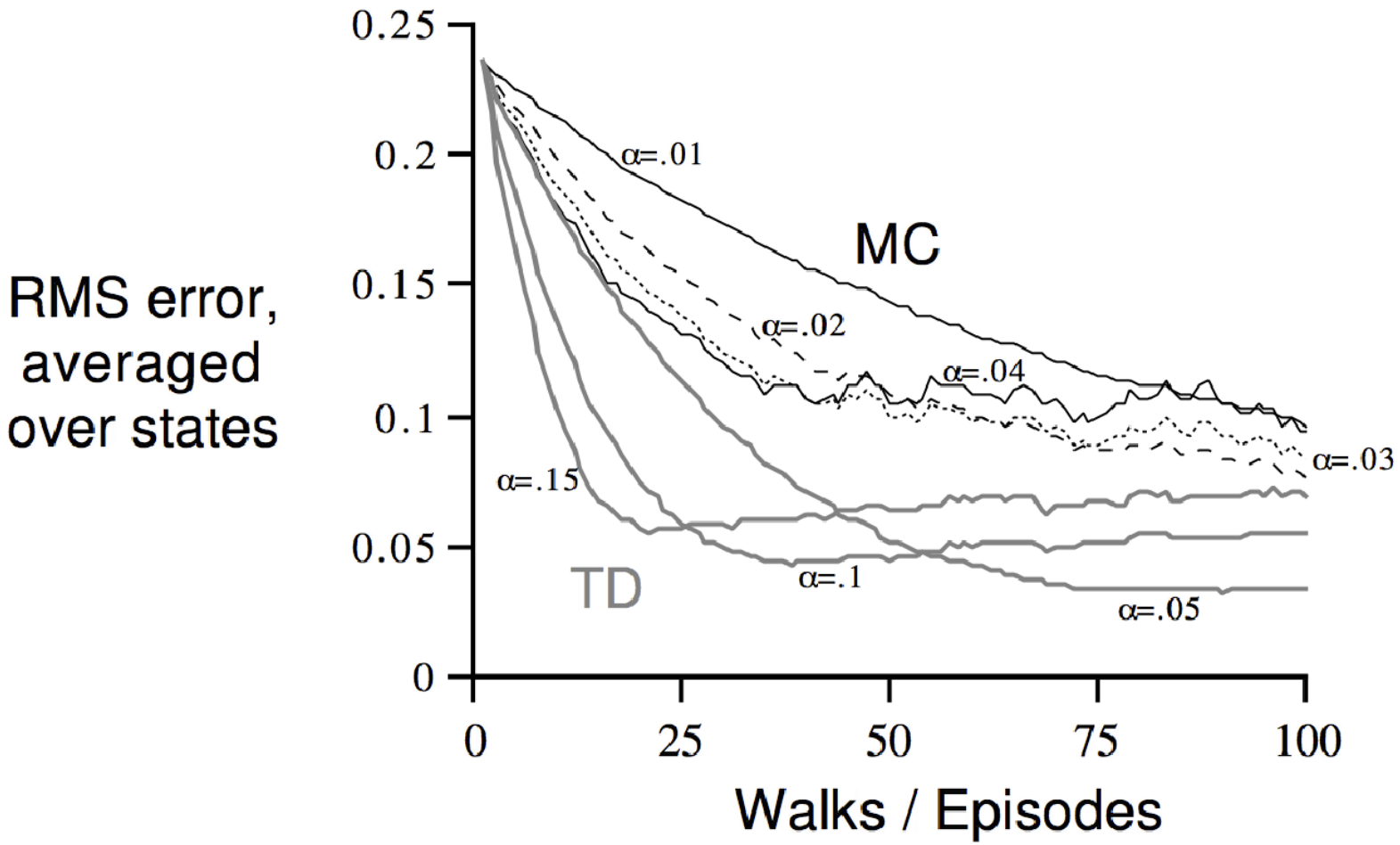
Marcello Restelli



Random Walk

MC vs TD

Marcello Restelli



Batch MC and TD

Marcello Restelli

- MC and TD **converge**: $V(s) \rightarrow V^{\pi}(s)$ as experience $\rightarrow \infty$
- But what about **batch** solution for finite experience?

$$s_1^1, a_1^1, r_2^1, \ldots, s_{T_1}^1$$
$$\vdots$$
$$s_1^K, a_1^K, r_2^K, \ldots, s_{T_1}^K$$

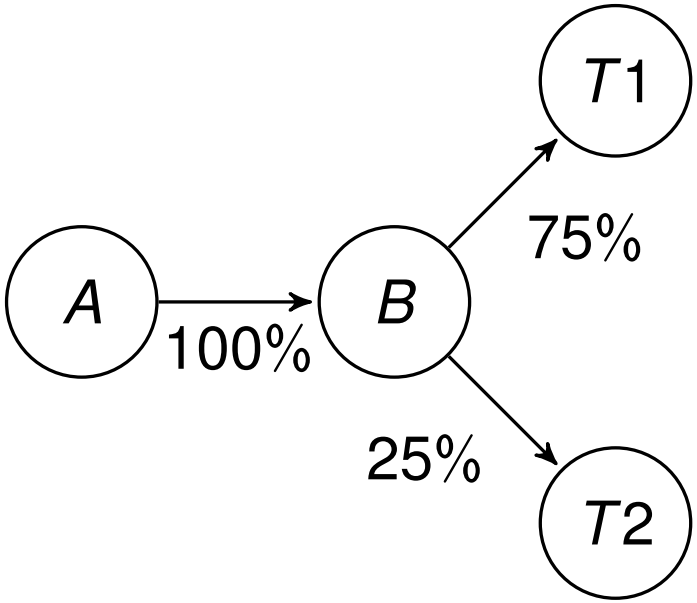
  - e.g., **repeatedly** sample episode $k \in [1, K]$
  - Apply MC or TD(0) to episode k

AB Example

Marcello Restelli

- Two states A, B
- Undiscounted
- 8 episodes of experience

$A, 0, B, 0$
$B, 1$
$B, 1$
$B, 1$
$B, 1$
$B, 1$
$B, 1$
$B, 0$



- What is $V(A)$, $V(B)$?

Certainty Equivalence

Marcello Restelli

- MC converges to solution with **minimum mean–squared error**
  - Best fit to the **observed returns**

$$\sum_{k=1}^{K}\sum_{t=1}^{T}(v_t^k - V(s_t^k))^2$$

  - In the AB example, $V(A) = 0$
- TD(0) converges to solution of **max likelihood Markov model**
  - Solution to the MDP $\langle \mathcal{S}, \mathcal{A}, \hat{P}, \hat{R}, \gamma, \mu \rangle$ that **best fits the data**

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

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

  - In the AB example, $V(A) = 0.75$

Comparison between MC and TD

Markov Property

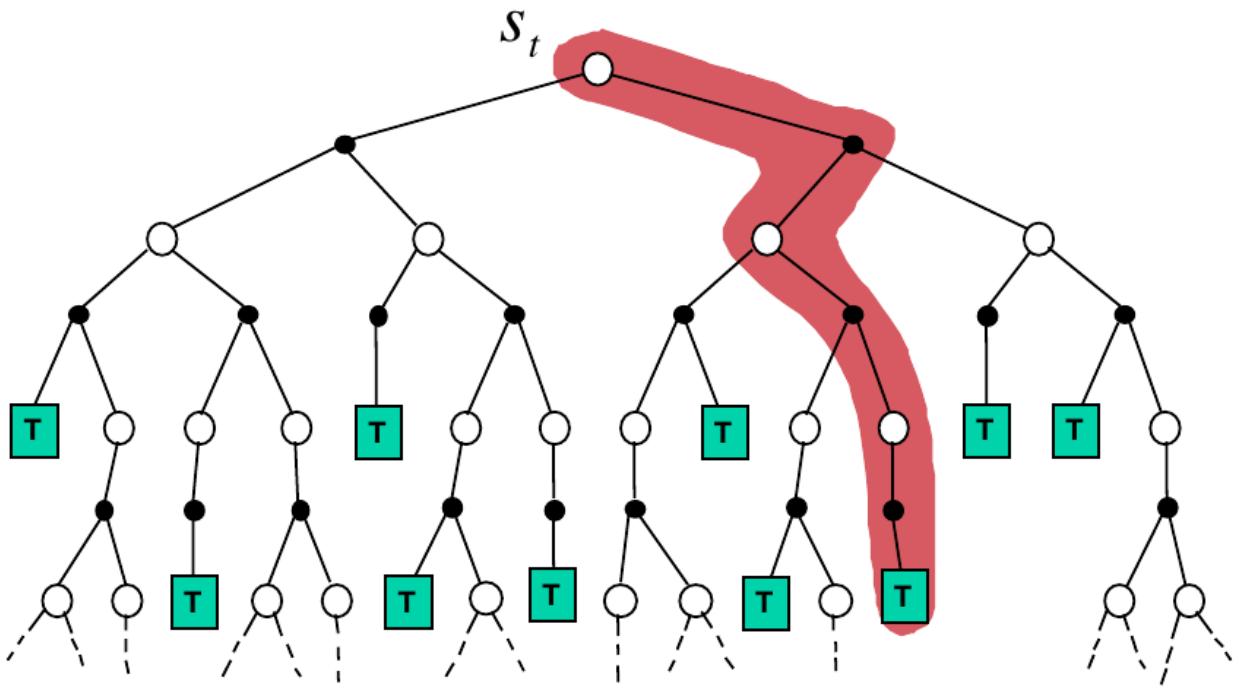
- TD **exploits** Markov property
  - Usually more efficient in **Markov environments**
- MC **does not exploit** Markov property
  - Usually more efficient in **non–Markov environments**

MC vs TD vs DP

Marcello Restelli

$$V(s_t) \leftarrow V(s_t) + \alpha[v_t - V(s_t)]$$

where R_t is the actual return following state s_t

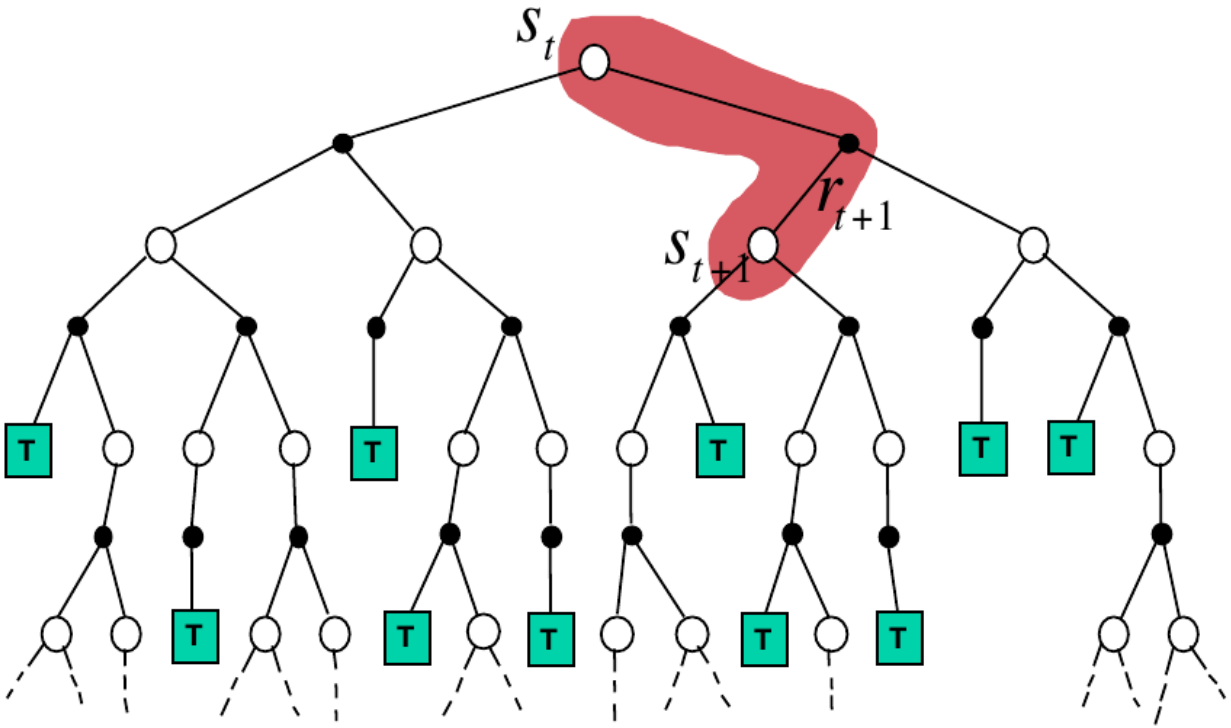


MC vs TD vs DP

Marcello Restelli

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$$

where R_t is the actual return following state s_t

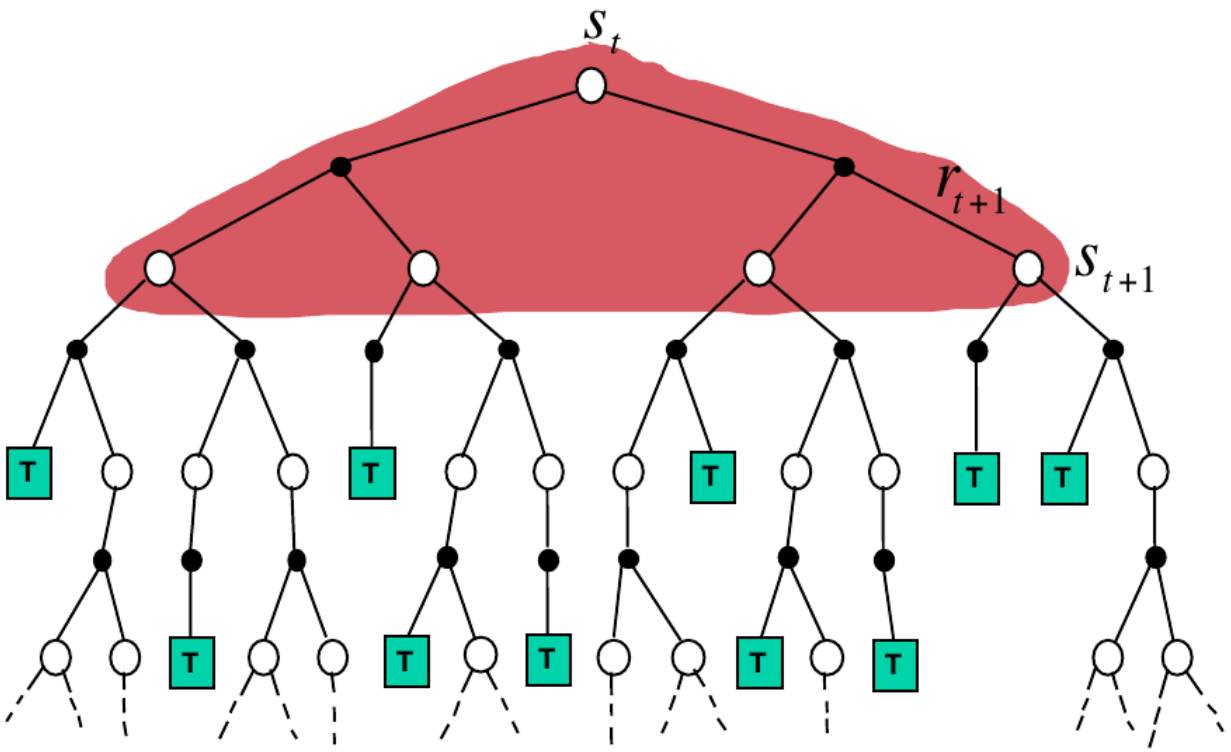


MC vs TD vs DP

Marcello Restelli

$$V(s_t) \leftarrow \mathbb{E}_\pi[r_{t+1} + \gamma V(s_{t+1})]$$

where R_t is the actual return following state s_t



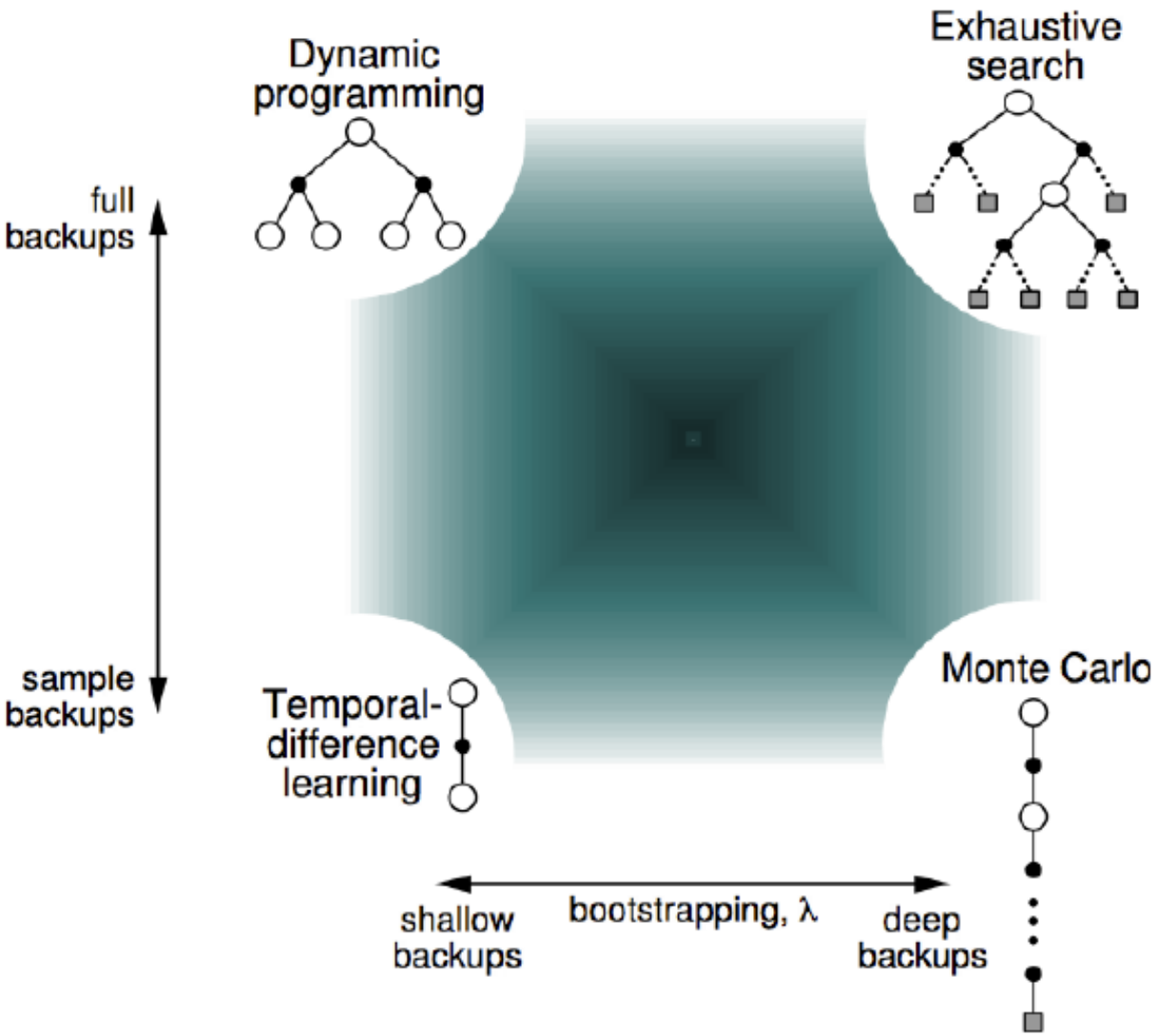
MC vs TD vs DP

Marcello Restelli

	Bootstrapping	Sampling
MC	No	Yes
TD	Yes	Yes
DP	Yes	No

MC vs TD vs DP

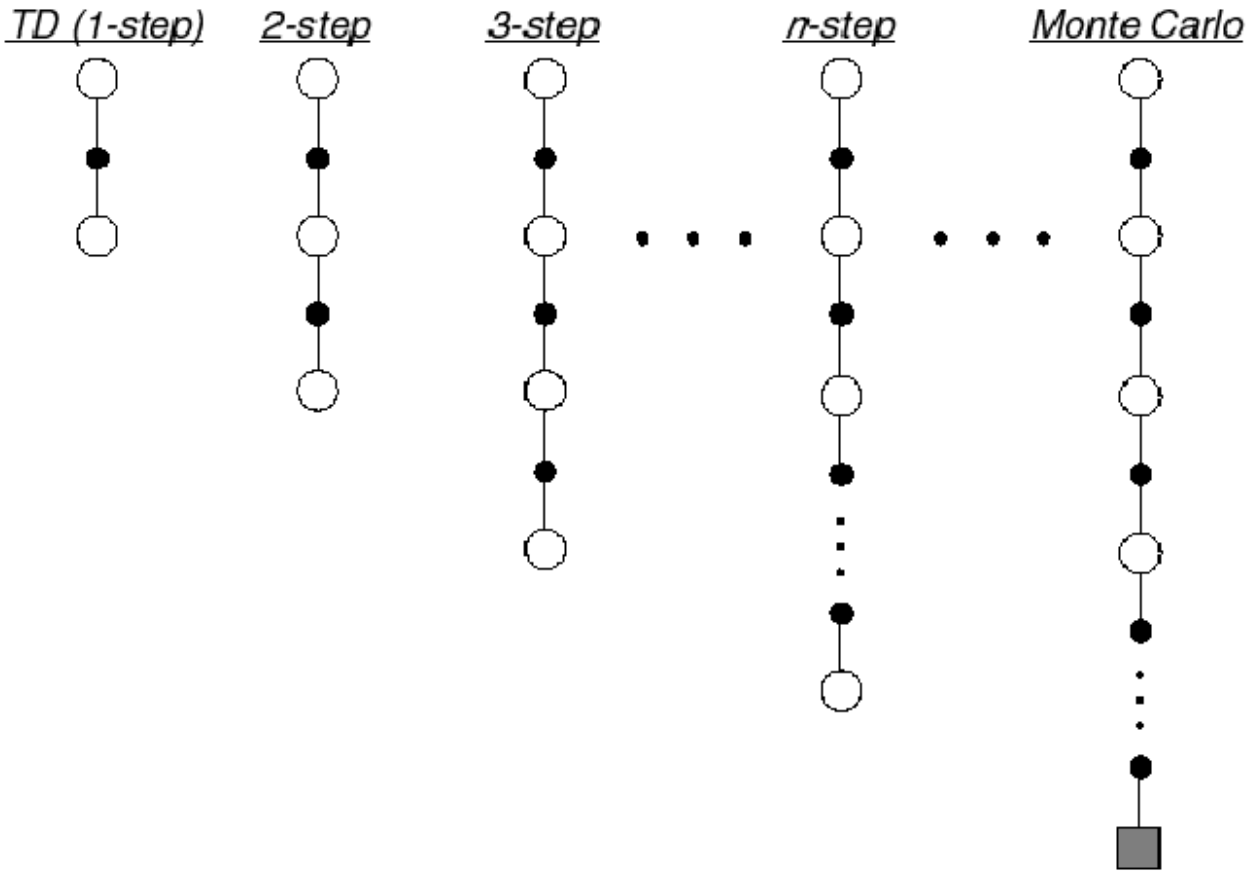
Marcello Restelli



n–Step Prediction

Marcello Restelli

Let TD target look *n* steps into the future



n–Step Return

Marcello Restelli

- Consider the following n–step returns for $n = 1, 2, \ldots, \infty$:

$$\begin{array}{lll} n = 1 & (TD) & v_t^{(1)} = r_{t+1} + \gamma V(s_{t+1}) \\ n = 2 & & v_t^{(2)} = r_{t+1} + \gamma r_{t+2} + \gamma^2 V(s_{t+2}) \\ \vdots & & \vdots \\ n = \infty & (MC) & v_t^{(\infty)} = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{T-1} r_T \end{array}$$

- Define the n–step return

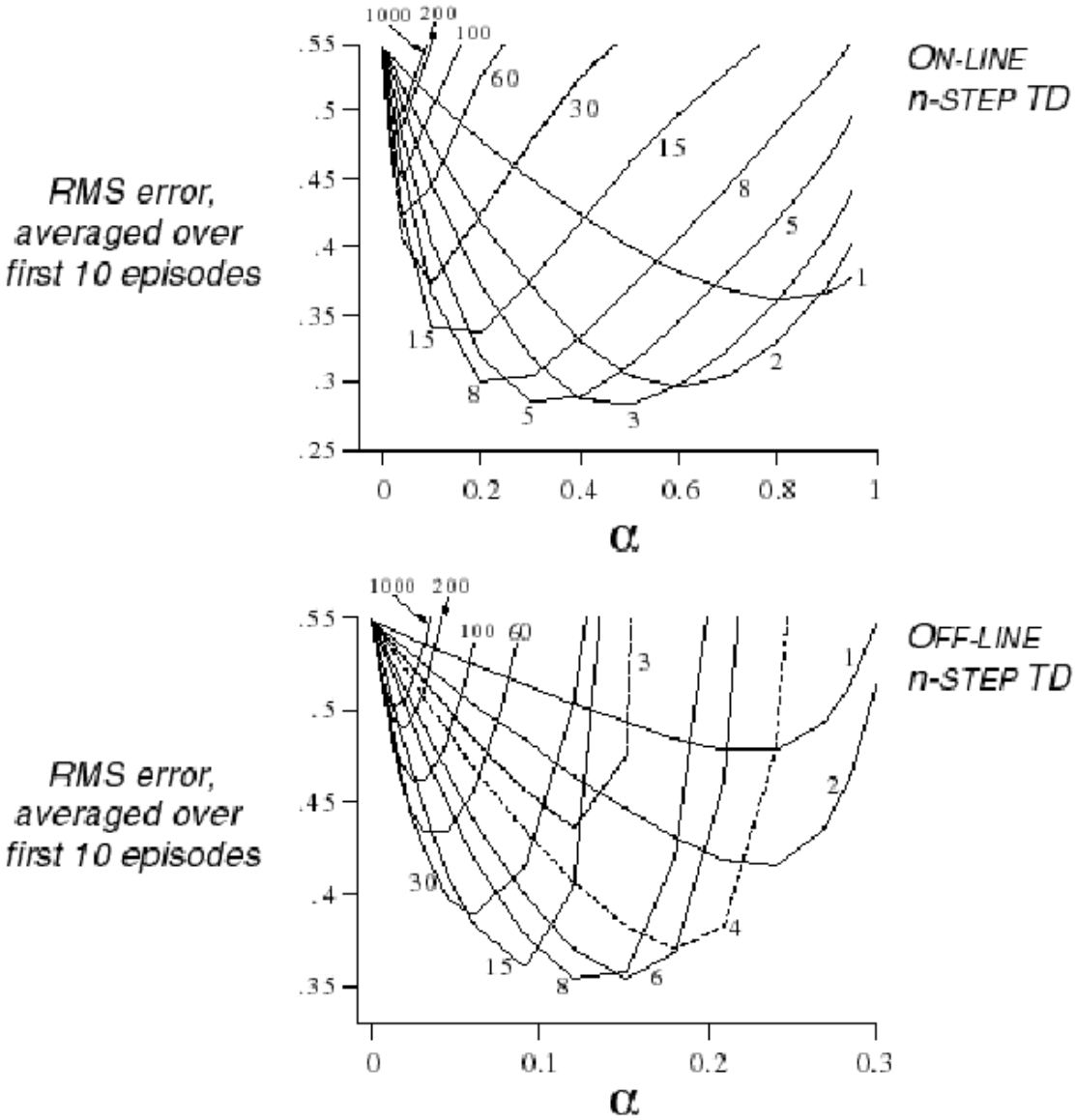
$$v_t^{(n)} = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{n-1} r_{t+n} + \gamma^n V(s_{t+n})$$

- n–step temporal–difference learning

$$V(s_t) \leftarrow V(s_t) + \alpha(v_t^{(n)} - V(s_t))$$

Large Random Walk Example

Marcello Restelli



Averaging n–step Returns

Marcello Restelli

- We can **average** n–step returns over **different** n
- e.g., average the 2–step and 4–step returns

$$\frac{1}{2} v^{(2)} + \frac{1}{2} v^{(4)}$$

- **Combines information** from two different time–steps
- Can we **efficiently** combine information from all time–steps?

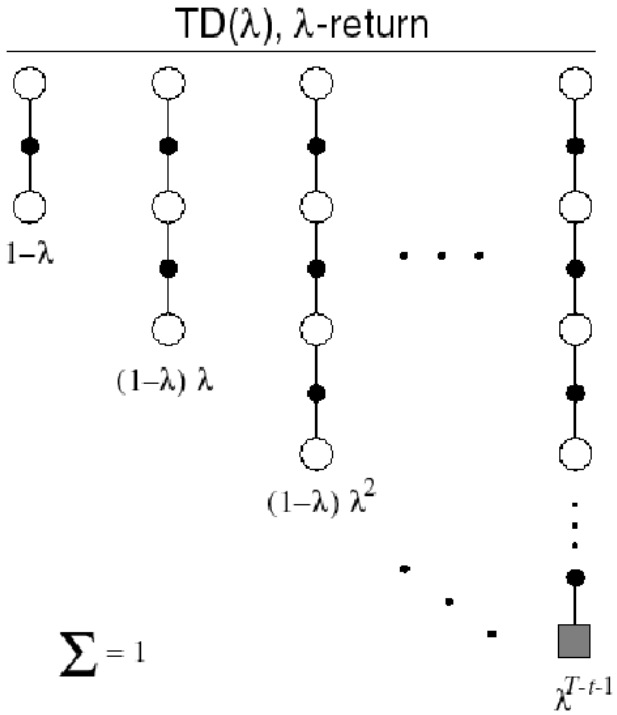
λ–return

- The λ–return v_t^λ combines **all** n–step returns $v_t^{(n)}$
- Using **weight** $(1-\lambda)\lambda^{n-1}$

$$v_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}v_t^{(n)}$$

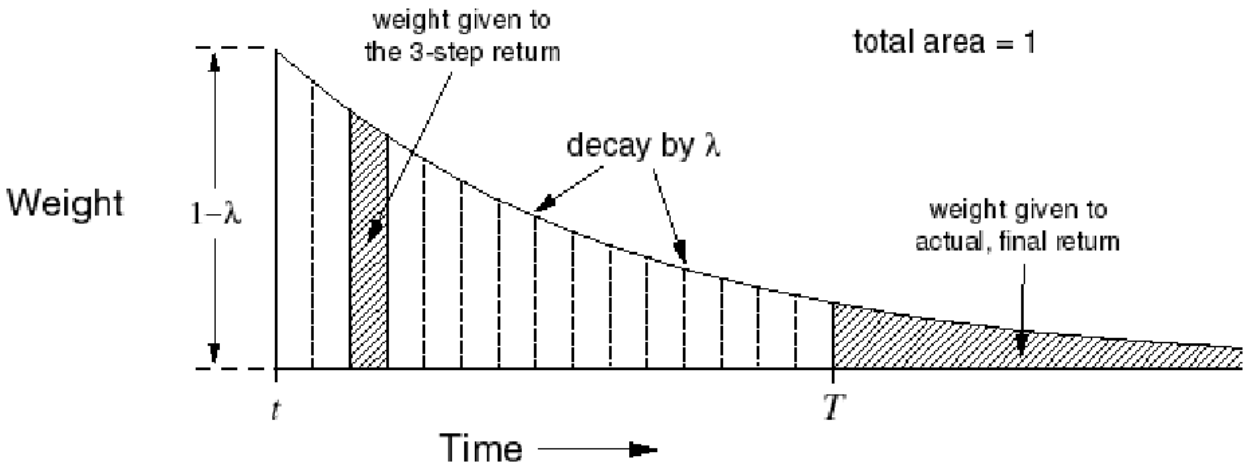
- **Forward–view** TD(λ)

$$V(s_t) \leftarrow V(s_t) + \alpha\left(v_t^\lambda - V(s_t)\right)$$



TD(λ) Weighting Function

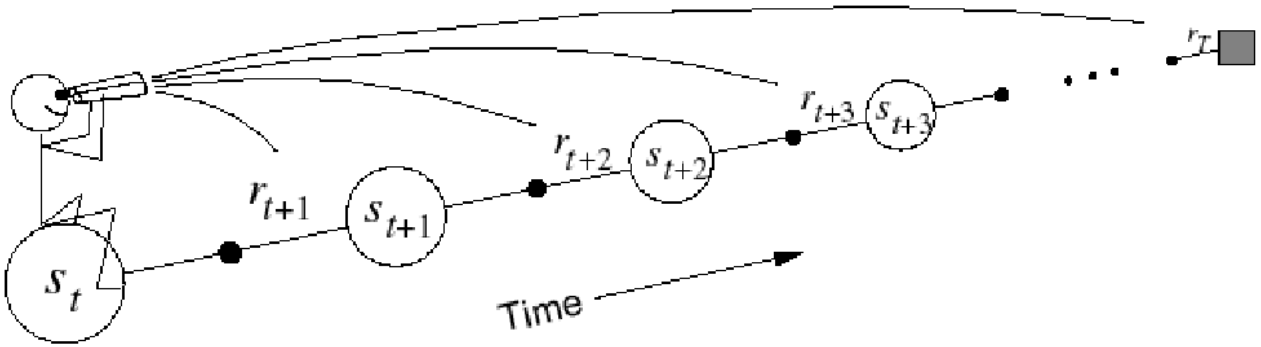
Marcello Restelli



$$v_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}v_t^{(n)}$$

Forward–view TD(λ)

Marcello Restelli

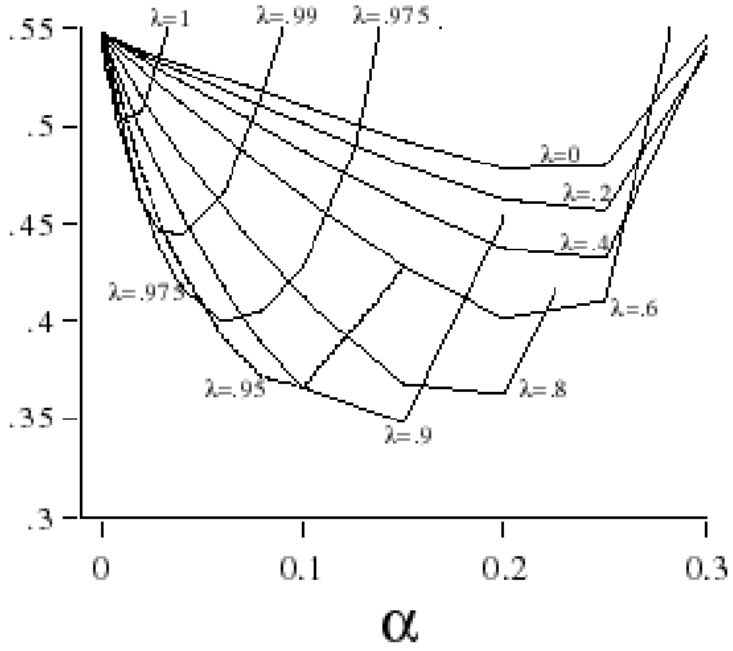


- Update value function towards the λ**–return**
- Forward-view looks into the **future** to compute v_t^λ
- Like MC, can only be computed from **complete episodes**

Forward–view TD(λ) on Large Random Walk

Marcello Restelli

RMS error, averaged over first 10 episodes



OFF-LINE
λ-RETURN

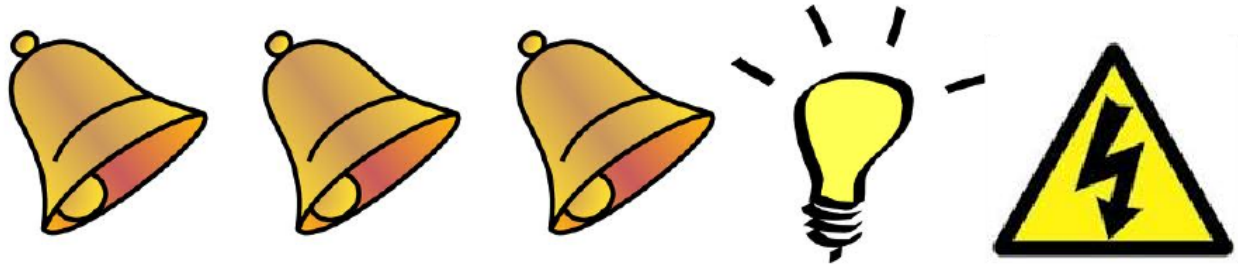
Backward–view TD(λ)

Marcello Restelli

- Forward view provides **theory**
- Backward view provides **mechanism**
- Update **online**, every step, from **incomplete sequences**

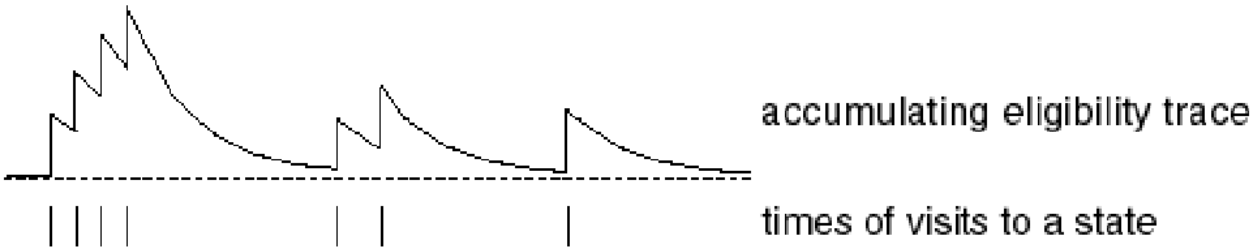
Eligibility Traces

Marcello Restelli



- **Credit assignment problem**: did bell or light cause shock?
- **Frequency heuristic**: assign credit to the most frequent states
- **Recency heuristics**: assign credit to the most recent states
- **Eligibility traces** combine both heuristics

$$e_{t+1}(s) = \gamma\lambda e_t(s) + \mathbf{1}(s = s_t)$$

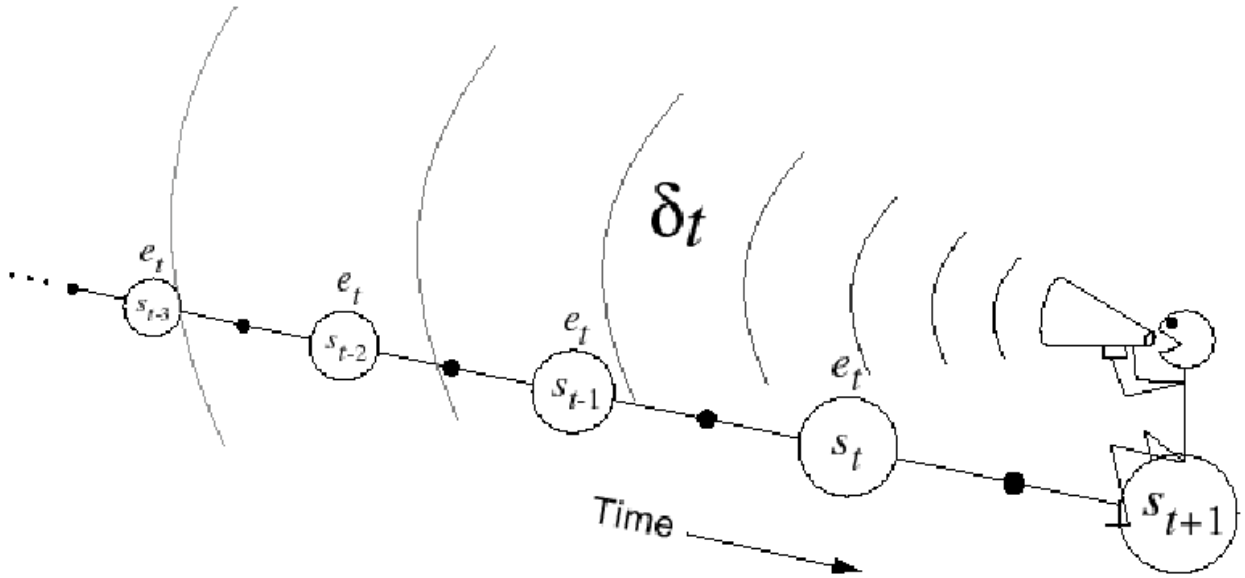


Backward–view TD(λ)

Marcello Restelli

- **Update value** $V(s)$ for every state s
- In proportion to TD–error δ_t and **eligibility trace** $e_t(s)$

$$
\begin{aligned}
e_0(s) &= 0 \\
e_t(s) &= \gamma\lambda e_{t-1}(s) + \mathbf{1}(s = s_t) \\
V(s) &\leftarrow V(s) + \alpha\delta_t e_t(s)
\end{aligned}
$$



Backward–view TD(λ) Algorithm

Marcello Restelli

Initialize $V(s)$ arbitrarily
for all episodes **do**
  $e(s) = 0, \quad \forall s \in \mathcal{S}$
  Initialize s
  repeat
    $a \leftarrow$ action given by π for s
    Take action a, observe reward r, and next state s'
    $\delta \leftarrow r + \gamma V(s') - V(s)$
    $e(s) \leftarrow e(s) + 1$
    for all $s \in \mathcal{S}$ **do**
      $V(s) \leftarrow V(s) + \alpha \delta e(s)$
      $e(s) \leftarrow \gamma \lambda e(s)$
    end for
    $s \leftarrow s'$
  until s is terminal
end for

TD(λ) and TD(0)

Marcello Restelli

- When $\lambda = 0$, only current state is updated

$$\begin{aligned} e_t(s) &= \mathbf{1}(s = s_t) \\ V(s) &\leftarrow V(s) + \alpha \delta_t e_t(s) \end{aligned}$$

- This is exactly equivalent to TD(0) update

$$V(s_t) \leftarrow V(s_t) + \alpha \delta_t$$

Telescoping in TD(1)

When $\lambda = 1$, sum of TD errors telescopes into MC error

$$
\begin{aligned}
& \delta_t + \gamma\delta_{t+1} + \gamma^2\delta_{t+2} + \cdots + \gamma^{T-t}\delta_{T-1} \\
= \;& r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \\
+ \;& \gamma r_{t+2} + \gamma^2 V(s_{t+2}) - \gamma V(s_{t+1}) \\
+ \;& \gamma^2 r_{t+3} + \gamma^3 V(s_{t+3}) - \gamma^2 V(s_{t+2}) \\
& \vdots \\
+ \;& \gamma^{T-1} r_{t+T} + \gamma^T V(s_{t+T}) - \gamma^{T-1} V(s_{t+T-1}) \\
= \;& r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-1} r_{t+T} - V(s_t) \\
= \;& v_t - V(s_t)
\end{aligned}
$$

TD(λ) and TD(1)

Marcello Restelli

- TD(1) is **roughly equivalent** to every–visit Monte–Carlo
- Error is accumulated online, **step–by–step**
- If value function is only **updated offline** at end of episode, then the total update is **exactly** the same as MC
- If value function is **updated online** after every step, then TD(1) may have **different** total update to MC

Forwards and Backwards TD(λ)

Marcello Restelli

- Consider an episode where s is **visited once** at time–step k
- TD(λ) eligibility trace **discounts time** since visit

$$e_t(s) = \gamma\lambda e_{t-1}(s) + \mathbf{1}(s_t = s)$$

$$= \begin{cases} 0 & \text{if } t < k \\ (\gamma\lambda)^{t-k} & \text{if } t \geq k \end{cases}$$

- Backward TD(λ) updates **accumulate error online**

$$\sum_{t=1}^{T} \alpha\delta_t e_t(s) = \alpha \sum_{t=k}^{T} (\gamma\lambda)^{t-k}\delta_t = \alpha(v_k - V(s_k))$$

- By end of episode it accumulates **total error** for λ–return
- For **multiple** visits to s, $e_t(s)$ accumulates **many errors**

Equivalence of Forward and Backward TD

Marcello Restelli

Theorem

The sum of offline updates is identical for forward–view and backward–view TD(λ)

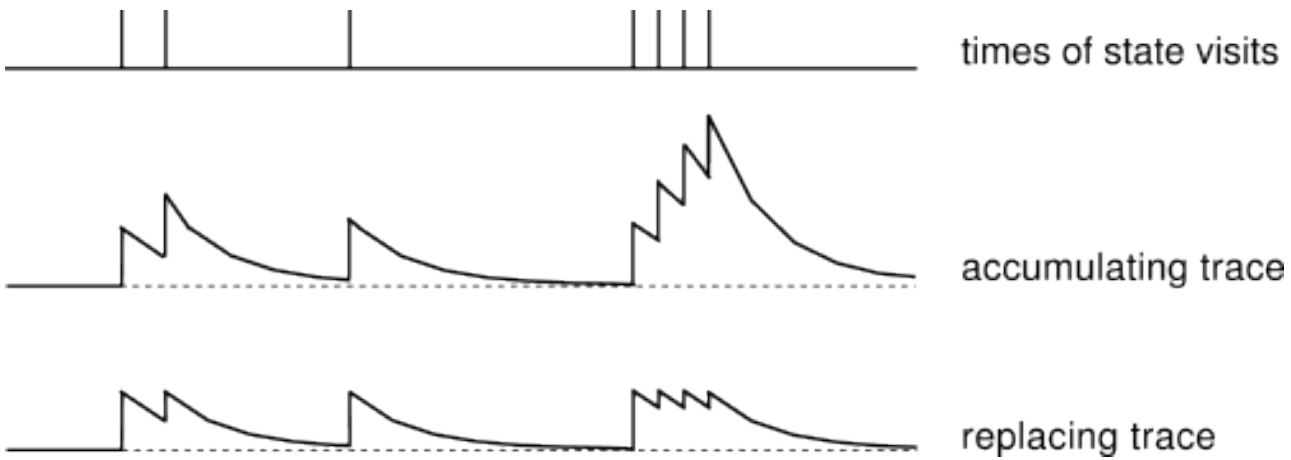
$$\sum_{t=1}^{T} \alpha \delta_t e_t(s) = \sum_{t=1}^{T} \alpha (v_t^\lambda - V(s_t)) \mathbf{1}(s_t = s)$$

- In **practice**, value function is updated **online** by TD(λ)
- But if α is **small** then equivalence is almost exact

Replacing Traces

- Using **accumulating** traces, **frequently** visited states can have eligibilities greater than 1
  - This can be a **problem for convergence**
- **Replacing traces**: Instead of adding 1 when you visit a state, set that trace to 1

$$e_t(s) = \begin{cases} \gamma\lambda e_{t-1}(s) & \text{if } s \neq s_t \\ 1 & \text{if } s = s_t \end{cases}$$



Use of Model–Free Control

Marcello Restelli

Some example problems that can be modeled as MDPs:

- Elevator
- Parallel Parking
- Ship Steering
- Bioreactor
- Helicopter
- Airplane Logistics
- Robocup Soccer
- Quake
- Portfolio management
- Protein folding
- Robot walking
- Game of Go

For most of these problems, either:

- MDP model is **unknown**, but experience can be **sampled**
- MDP model is **known**, but is **too big** to use, except by samples

Model–free control can solve these problems

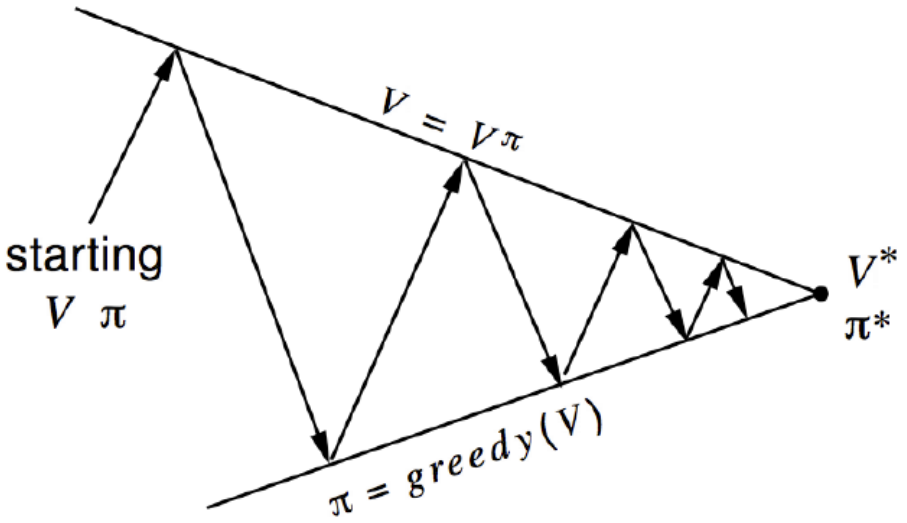
On and Off–Policy Learning

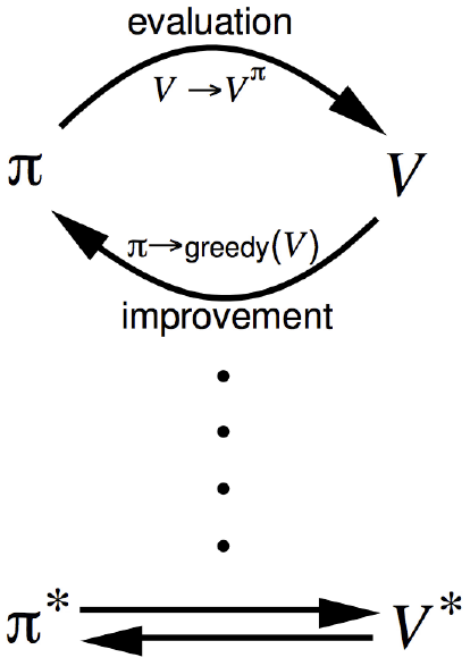
Marcello Restelli

- **On–policy** learning
  - "Learn on the job"
  - Learn about policy π from experience sampled from π
- **Off–policy** learning
  - "Learn over someone's shoulder"
  - Learn about policy π from experience sampled from $\overline{\pi}$

Generalized Policy Iteration (Refresher)

Marcello Restelli

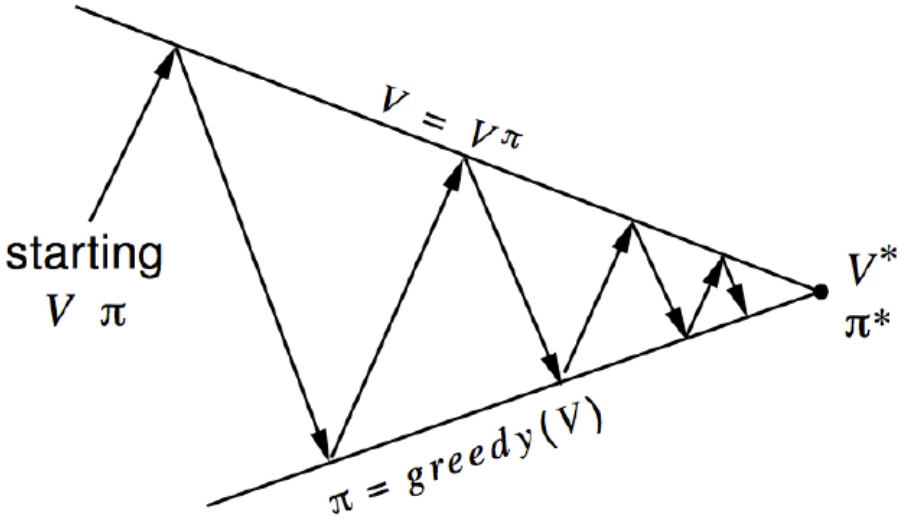




- **Policy evaluation**: Estimate V^π
  - e.g., Iterative policy evaluation
- **Policy improvement**: Generate $\pi' \geq \pi$
  - e.g., Greedy policy improvement

Generalized Policy Iteration with Monte–Carlo Evaluation

Marcello Restelli



- **Policy Evaluation**: Monte–Carlo policy evaluation, $V = V^\pi$?
- **Policy Improvement**: Greedy policy improvement?

Model–Free Policy Iteration Using Action–Value Function

Marcello Restelli

- Greedy policy improvement over $V(s)$ **requires model** of MDP

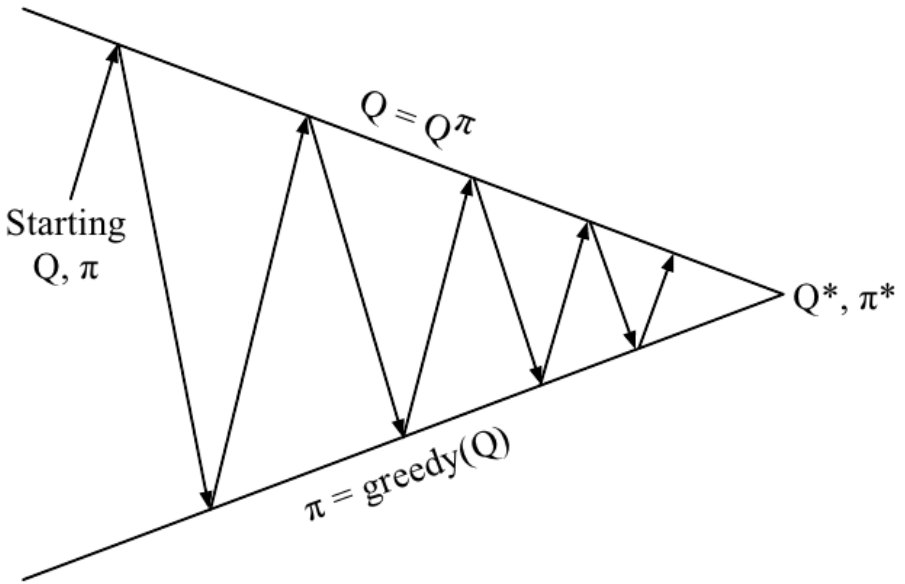
$$\pi'(s) = arg \max_{a \in \mathcal{A}} \left\{ R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V(s') \right\}$$

- Greedy policy improvement over $Q(s,a)$ is **model–free**

$$\pi'(s) = arg \max_{a \in \mathcal{A}} Q(s,a)$$

Generalized Policy Iteration with Monte–Carlo Evaluation

Marcello Restelli



- **Policy Evaluation**: Monte–Carlo policy evaluation, $Q = Q^\pi$
- **Policy Improvement**: Greedy policy improvement?

On–Policy Exploration

Marcello Restelli

"Behind one door is tenure - behind the other is flipping burgers at McDonald's."

- There are two doors in front of you
- You open the left door and get reward 0, $V(left) = 0$
- You open the right door and get reward $+1$, $V(right) = +1$
- You open the right door and get reward $+3$, $V(right) = +2$
- You open the right door and get reward $+2$, $V(right) = +2$

$\vdots$

- Are you sure you've chosen the **best** door?

ϵ–Greedy Exploration

Marcello Restelli

- **Simplest** idea for ensuring **continual exploration**
- **All** m actions are tried with **non–zero** probability
- With probability $1 - \epsilon$ choose the **greedy action**
- With probability ϵ choose an action **at random**

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

ϵ–Greedy Policy Improvement

Marcello Restelli

Theorem

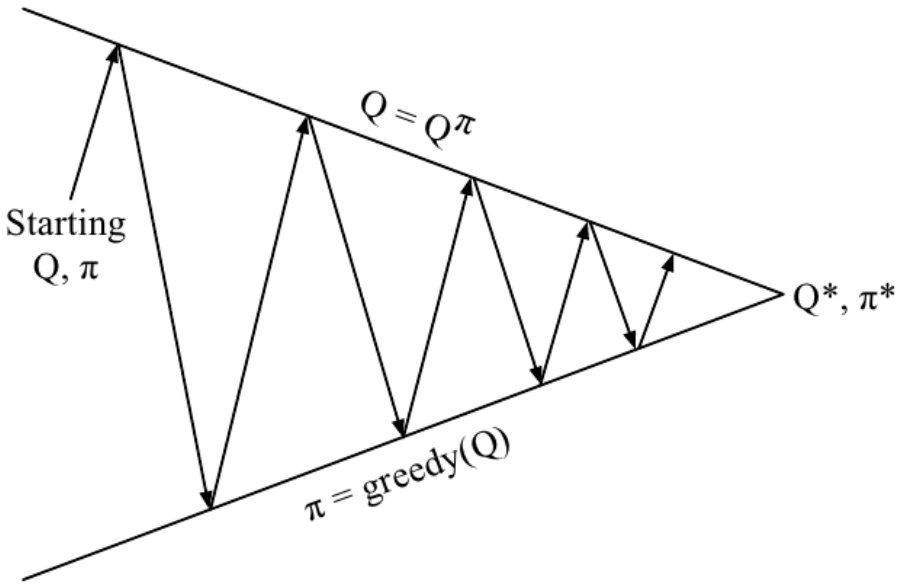
For any ϵ–greedy policy π, the ϵ–greedy policy π' with respect to Q^π is an improvement

$$\begin{aligned} Q^\pi(s,\pi'(s)) &= \sum_{a\in\mathcal{A}} \pi'(a|s) Q^\pi(s,a) \\ &= \frac{\epsilon}{m} \sum_{a\in\mathcal{A}} Q^\pi(s,a) + (1-\epsilon) \max_{a\in\mathcal{A}} Q^\pi(s,a) \\ &\geq \frac{\epsilon}{m} \sum_{a\in\mathcal{A}} Q^\pi(s,a) + (1-\epsilon) \sum_{a\in\mathcal{A}} \frac{\pi(a|s) - \frac{\epsilon}{m}}{1-\epsilon} Q^\pi(s,a) \\ &= \sum_{a\in\mathcal{A}} \pi(a|s) Q^\pi(s,a) = V^\pi(s) \end{aligned}$$

Therefore from policy improvement theorem, $V^{\pi'(s)} \geq V^\pi(s)$

Monte–Carlo Policy Iteration

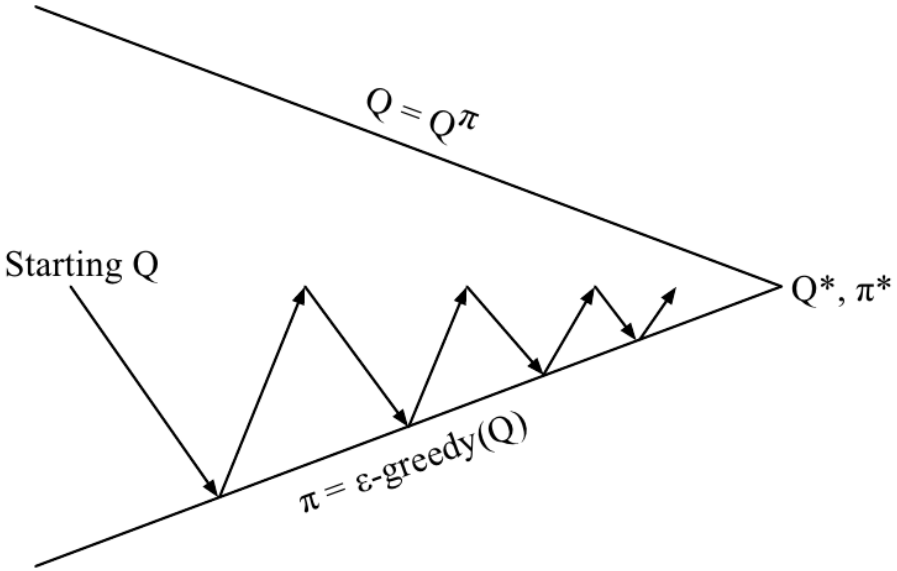
Marcello Restelli



- **Policy Evaluation**: Monte–Carlo policy evaluation, $Q = Q^\pi$
- **Policy Improvement**: ϵ–greedy policy improvement

Monte–Carlo Control

Marcello Restelli



Every episode:

- **Policy Evaluation**: Monte–Carlo policy evaluation, $Q \approx Q^\pi$
- **Policy Improvement**: ϵ–greedy policy improvement

GLIE

Marcello Restelli

Definition

Greedy in the Limit of Infinite Exploration (GLIE)

- **All** state–action pairs are explored **infinitely** many times

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

- The policy **converges** on a **greedy** policy

$$\lim_{t \to \infty} \pi_k(a|s) = \mathbf{1}(a = arg \max_{a' \in \mathcal{A}} Q_k(s', a'))$$

GLIE Monte–Carlo Control

Marcello Restelli

- Sample k–th episode using π: $\{s_1, a_1, r_2, \ldots, s_T\} \sim \pi$
- For each state s_t and action a_t in the episode,

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

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

- **Improve policy** based on new action–value function

$$\epsilon \leftarrow \frac{1}{k}$$

$$\pi \leftarrow \epsilon\text{–greedy}(Q)$$

Theorem

GLIE Monte–Carlo control ***converges*** *to the* ***optimal*** *action–value function,* $Q(s, a) \rightarrow Q^*(s, a)$

Relevant Time Scales

- There are three main time scales
  1. **Behavioral** time scale $\frac{1}{1-\gamma}$ (**discount factor**)
  2. **Sampling** in the estimation of the Q–function α (**learning rate**)
  3. **Exploration** ϵ (e.g., for ϵ–greedy strategy)
- $1-\gamma \gg \alpha \gg \epsilon$
- **Initially** $1-\gamma \approx \alpha \approx \epsilon$ is possible
- Then decrease ϵ **faster** than α
- **Practically**, you can choose number of trials $M < \infty$ and set $\alpha \sim 1 - \frac{m}{M}$ and $\epsilon \sim \left(1 - \frac{m}{M}\right)^2, \quad m = 1, \ldots, M$
- In some cases, γ should be **initialized to low values** and then gradually moved towards its correct value

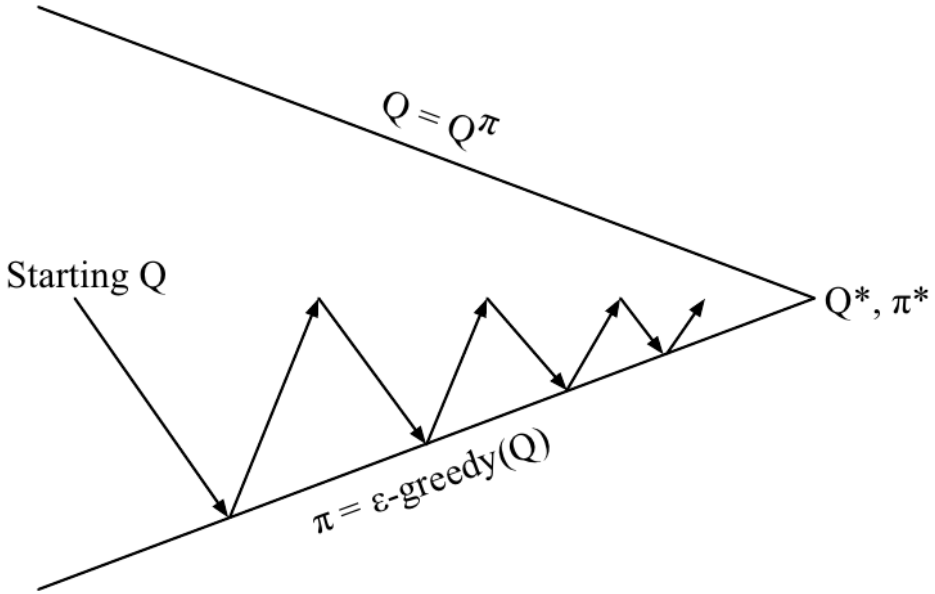
MC vs TD Control

- Temporal–Difference (TD) learning has several **advantages** over Monte–Carlo (MC)
  - Lower Variance
  - Online
  - Incomplete sequences
- Natural idea: use **TD** instead of MC in our **control loop**
  - Apply TD to $Q(s, a)$
  - Use ϵ–greedy policy improvement
  - Update every time–step

On–Policy Control with SARSA

Marcello Restelli

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



Every time–step:

- **Policy Evaluation**: SARSA, $Q \approx Q^{\pi}$
- **Policy Improvement**: ϵ–greedy policy improvement

SARSA Algorithm for On–Policy Control

Marcello Restelli

Initialize $Q(s, a)$ arbitrarily
loop
  Initialize s
  Choose a from s using policy derived from Q (e.g., ϵ–greedy)
  repeat
    Take action a, observe r, s'
    Choose a' from s' using policy derived from Q (e.g., ϵ–greedy)
    $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$
    $s \leftarrow s'; a \leftarrow a';$
  until s is terminal
end loop

Convergence of SARSA

Marcello Restelli

Theorem

SARSA converges to the optimal action–value function, $Q(s,a) \rightarrow Q^*(s,a)$, under the following conditions:

- *GLIE sequence of policies $\pi_t(s,a)$*
- *Robbins–Monro sequence of step–sizes α_t*

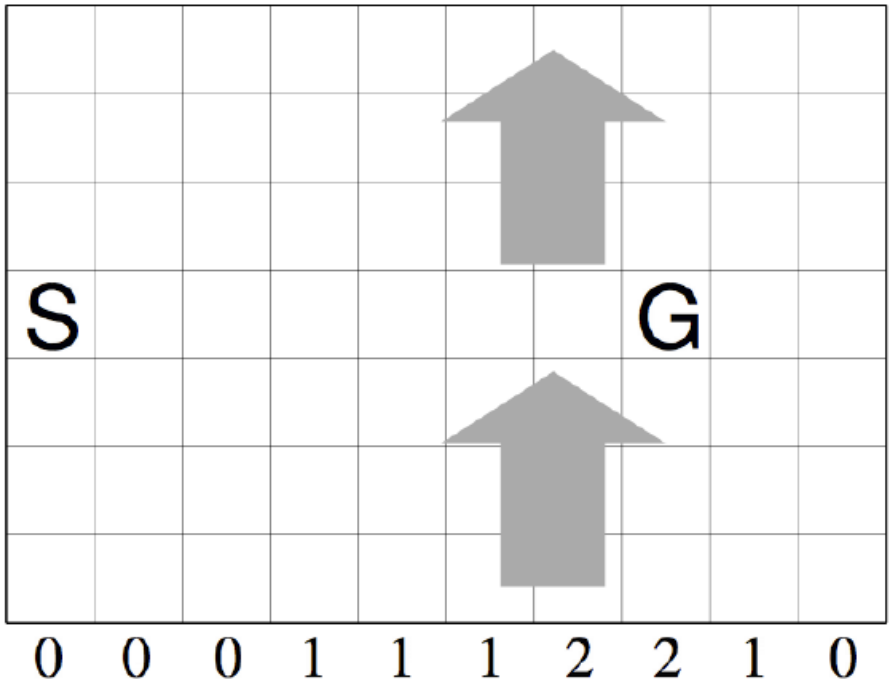
$$\sum_{t=1}^{\infty} \alpha_t = \infty$$

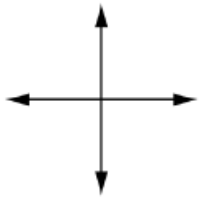
$$\sum_{t=1}^{\infty} \alpha_t^2 < \infty$$

SARSA Example

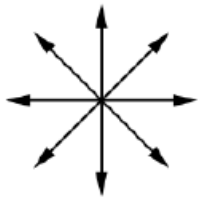
Windy Gridworld

Marcello Restelli





standard moves



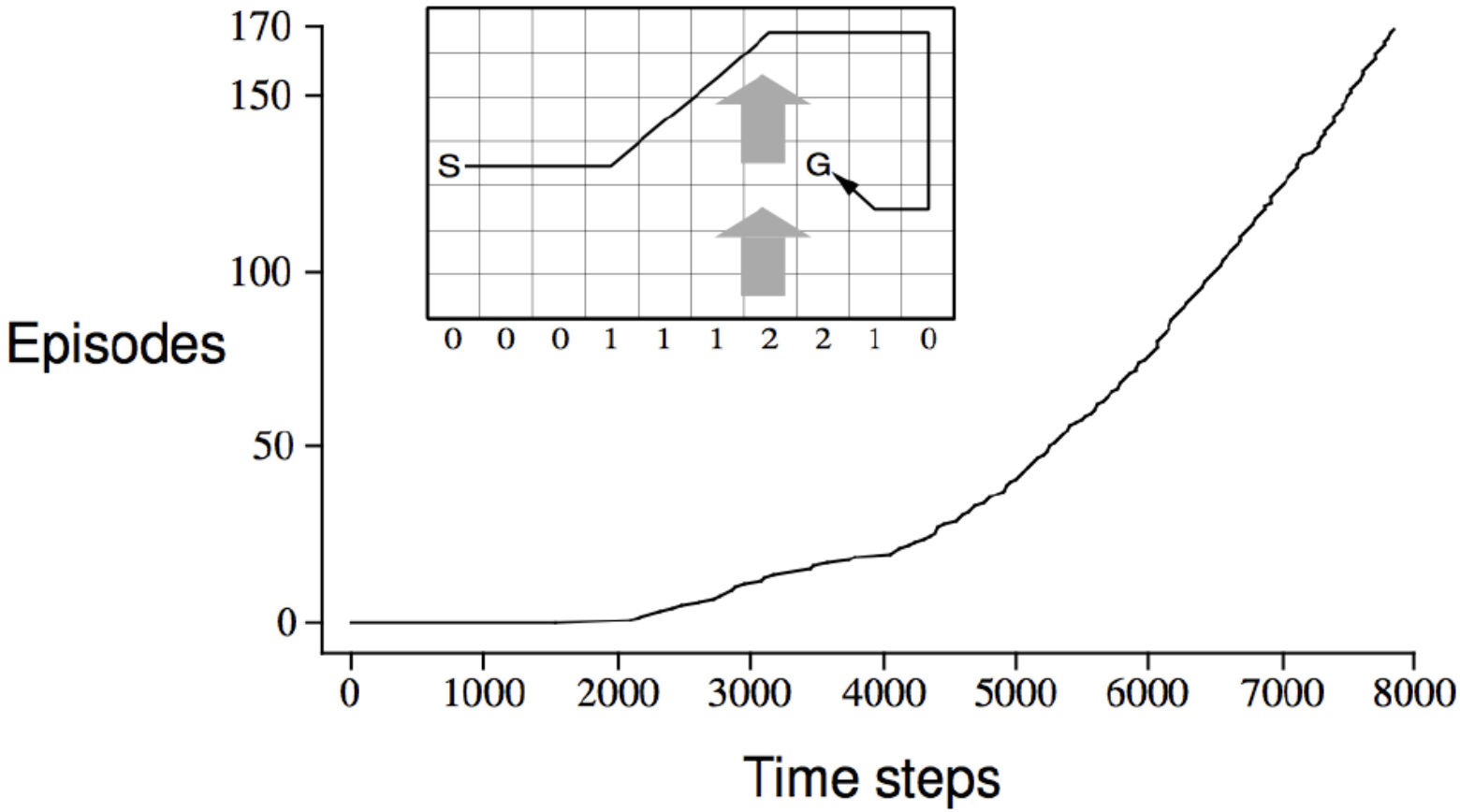
king's moves

undiscounted, episodic, reward = –1 until goal

SARSA Example

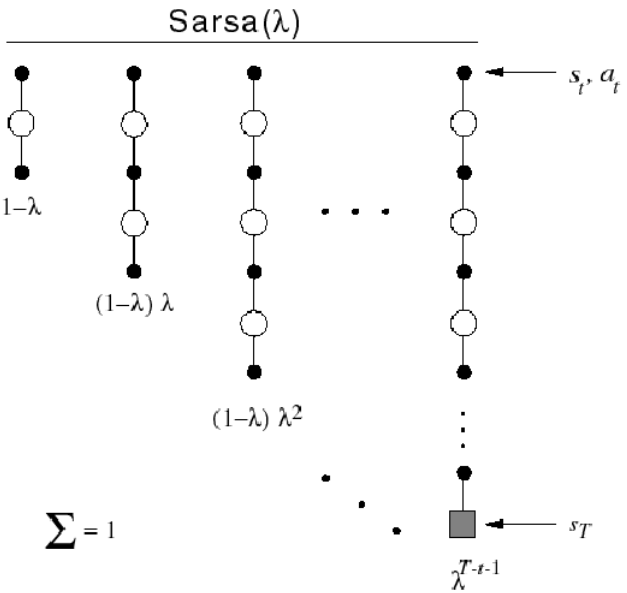
Results in the Windy Gridworld

Marcello Restelli



SARSA with Eligibility Traces

Marcello Restelli



- **Forward view**: update action–value $Q(s,a)$ to λ–return v_t^λ
- **Backward view**: use eligibility traces for state–action pairs

$$e_t(s,a) = \gamma\lambda e_{t-1}(s,a) + \mathbf{1}(s_t, a_t = s, a)$$

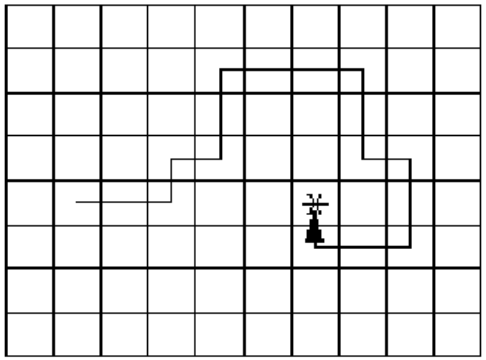
SARSA(λ) Algorithm

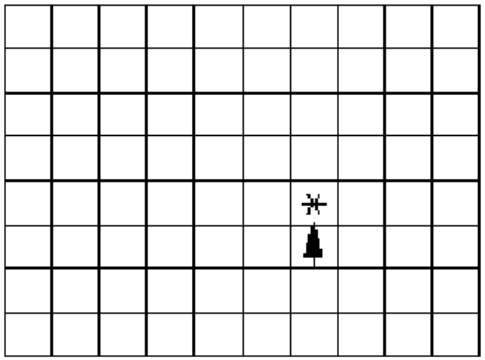
Marcello Restelli

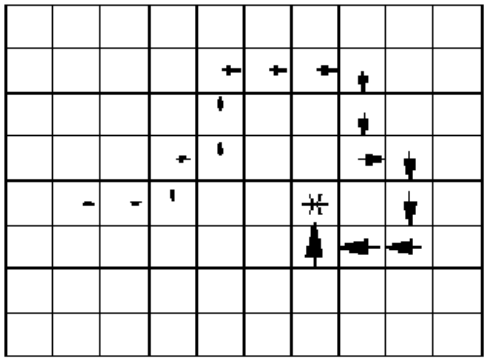
Initialize $Q(s, a)$ arbitrarily
loop
  $e(s, a) = 0$, for all s, a
  Initialize s, a
  repeat
    Take action a, observe r, s'
    Choose a' from s' using policy derived from Q (e.g., ϵ–greedy)
    $\delta \leftarrow r + \gamma Q(s', a') - Q(s, a)$
    $e(s, a) \leftarrow e(s, a) + 1$
    for all s, a **do**
      $Q(s, a) \leftarrow Q(s, a) + \alpha\delta e(s, a)$
      $e(s, a) \leftarrow \gamma\lambda e(s, a)$
    end for
    $s \leftarrow s'; a \leftarrow a';$
  until s is terminal
end loop

SARSA(λ) Gridworld Example

Marcello Restelli







Off–Policy Learning

Marcello Restelli

- Learn about **target policy** $\pi(a|s)$
- While following **behavior policy** $\overline{\pi}(a|s)$
- Why is this important?
  - Learn from **observing** humans or other agents
  - **Re–use** experience generated from old policies $\pi_1, \pi_2, \ldots, \pi_{t-1}$
  - Learn about **optimal** policy while following **exploratory** policy
  - Learn about **multiple** policies while following **one** policy

Importance Sampling

Marcello Restelli

Estimate the expectation of a **different** distribution w.r.t. the distribution used to **draw samples**

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

Importance Sampling for Off–Policy Monte–Carlo

Marcello Restelli

- Use returns **generated** from $\overline{\pi}$ to **evaluate** π
- Weight return v_t according to **similarity** between policies
- Multiply **importance sampling corrections** along whole episode

$$v_t^{\mu} = \frac{\pi(a_t|s_t)}{\overline{\pi}(a_t|s_t)} \frac{\pi(a_{t+1}|s_{t+1})}{\overline{\pi}(a_{t+1}|s_{t+1})} \cdots \frac{\pi(a_T|s_T)}{\overline{\pi}(a_T|s_T)} v_t$$

- Update value towards **corrected** return

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(v_t - Q(s_t, a_t))$$

- **Cannot use** if $\overline{\pi}$ is zero where π is non–zero
- Importance sampling can dramatically **increase variance**

Importance Sampling for Off–Policy Monte–Carlo

Derivation

Off–policy MC is derived from **expected return**:

$$
\begin{aligned}
Q^{\pi}(s,a) &= \mathbb{E}_{\pi}[v_t | s_t = s, a_t = a] \\
&= \sum \mathbb{P}[s_1, a_1, r_2, \ldots, s_T] v_t \\
&= \sum \mathbb{P}[s_1] \left(\Pi_{t=1}^{T} \overline{\pi}(s_t, a_t) P(s_t | s_{t-1}, a_{t-1}) \frac{\pi(s_t | a_t)}{\overline{\pi}(s_t, a_t)} \right) v_t \\
&= \mathbb{E}_{\overline{\pi}} \left[\Pi_{t=1}^{T} \frac{\pi(s_t, a_t)}{\overline{\pi}(s_t, a_t)} v_t | s_t = s, a_t = a \right]
\end{aligned}
$$

Off–Policy MC Control

Initialize, for all $s \in \mathcal{S}, a \in \mathcal{A}$:
$Q(s,a) \leftarrow$ arbitrary
$N(s,a) \leftarrow 0$
$D(s,a) \leftarrow 0$
$\pi \leftarrow$ an arbitrary deterministic policy
loop
  Using a policy $\overline{\pi}$, generate an episode
  $s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_{T-1}, a_{T-1}, r_T, s_T$
  $\tau \leftarrow$ latest time at which $a_\tau \neq \pi(s_\tau)$
  for all pair s, a appearing in the episode after τ **do**
    $t \leftarrow$ the time of first occurence (after τ) of s, a
    $w \leftarrow \Pi_{k=t+1}^{T-1} \frac{1}{\overline{\pi}(s_k, a_k)}$
    $N(s,a) \leftarrow N(s,a) + wR_t$
    $D(s,a) \leftarrow D(s,a) + w$
    $Q(s,a) \leftarrow \frac{N(s,a)}{D(s,a)}$
  end for
  for all $s \in \mathcal{S}$ **do**
    $\pi(s) \leftarrow arg\max_{a \in \mathcal{A}} Q(s,a)$
  end for
end loop

Importance Sampling for Off–Policy SARSA

Marcello Restelli

- Use **TD targets** generated from π to evaluate $\overline{\pi}$
- **Weight** TD target $r + \gamma Q(s', a')$ according to **similarity** between policies
- Only need a **single** importance sampling correction

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(r_{t+1} + \gamma \frac{\pi(a|s)}{\overline{\pi}(a|s)} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right)$$

- Much **lower variance** than Monte–Carlo importance sampling
- Policies only need to be similar over a **single step**

Importance Sampling for Off–Policy SARSA

Bellman expectation equation

Off–Policy SARSA comes from Bellman expectation equation for $Q^\pi(s,a)$

$$
\begin{aligned}
Q^\pi(s,a) &= \mathbb{E}_\pi \left[r_{t+1} + \gamma Q^\pi(s_{t+1}, a_{t+1}) | s_t = s, a_t = a \right] \\
&= R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) \sum_{a' \in \mathcal{A}} \pi(a'|s') Q^\pi(s',a') \\
&= R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) \sum_{a' \in \mathcal{A}} \overline{\pi}(a'|s') \frac{\pi(a'|s')}{\overline{\pi}(a'|s')} Q^\pi(s',a') \\
&= \mathbb{E}_\mu \left[r_{t+1} + \gamma \frac{\pi(a|s)}{\overline{\pi}(a|s)} Q^\pi(s_{t+1}, a_{t+1}) | s_t = s, a_t = a \right]
\end{aligned}
$$

Off–Policy Control with Q–learning

Marcello Restelli

- Learn about **optimal policy** $\pi = \pi^*$
- From experience sampled from **behavior policy** $\overline{\pi}$
- Estimate $Q(s,a) \approx Q^*(s,a)$
- Behavior policy **can depend on** $Q(s,a)$
  - e.g., $\overline{\pi}$ could be ϵ–greedy with respect to $Q(s,a)$
  - As $Q(s,a) \rightarrow Q^*(s,a)$, behavior policy $\overline{\pi}$ **improves**

Q–learning Algorithm

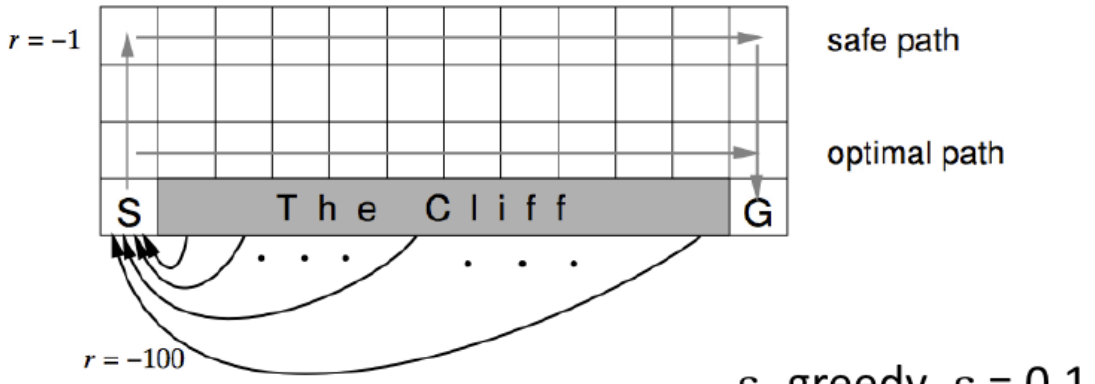
$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \max_{a \in \mathcal{A}} Q(s',a') - Q(s,a))$$

Initialize $Q(s,a)$ arbitrarily
loop
  Initialize s
  repeat
    Choose a from s using policy derived from Q (e.g., ϵ–greedy)
    Take action a, observe r, s'
    $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
    $s \leftarrow s'$;
  until s is terminal
end loop

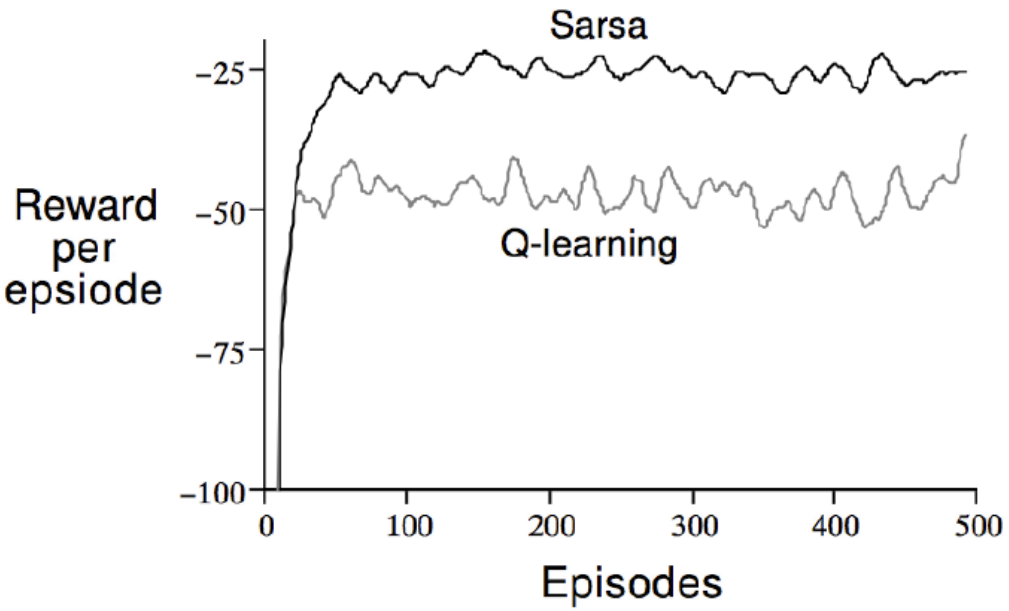
SARSA vs *Q*–learning

Cliffwalking

Marcello Restelli



ε–greedy, ε = 0.1



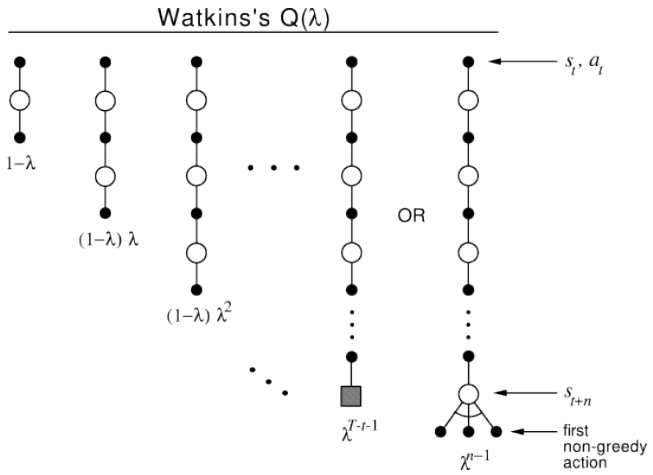
Q–learning vs SARSA

Marcello Restelli

- SARSA: $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]$ on–policy
- Q–learning: $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$ off–policy
- In the cliff–walking task:
  - Q–learning: learns **optimal policy** along edge
  - SARSA: learns a **safe non–optimal policy** away from edge
- ϵ–greedy algorithm
  - For $\epsilon \neq 0$ SARSA performs **better online**
  - For $\epsilon \to 0$ gradually, **both converge to optimal**

Q–Learning, the TD(λ) way

- How we can extend this to Q–learning?
- If you mark every state action pair as eligible, you backup over non–greedy policy
- **Watkins**:
  - Zero out eligibility trace after a non–greedy actions
  - Do max when backing up at first non–greedy choice



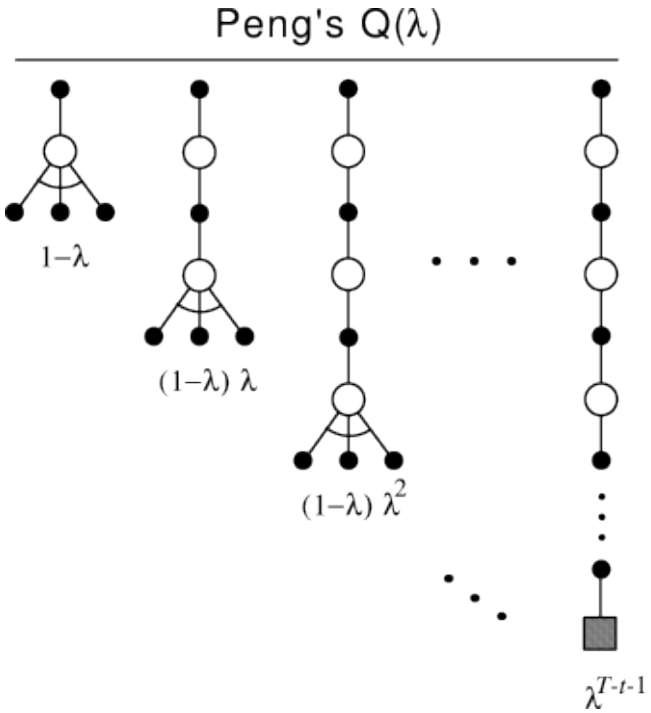
$$e_t(s,a) = \begin{cases} 1 + \gamma\lambda e_{t-1}(s,a) & \text{if } s = s_t, a = a_t, Q_{t-1}(s_t,a_t) = \max_a Q_{t-1}(s_t,a) \\ 0 & \text{if } Q_{t-1}(s_t,a_t) \neq \max_a Q_{t-1}(s_t,a) \\ \gamma\lambda e_{t-1}(s,a) & \text{otherwise} \end{cases}$$

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha\delta_t e_t(s,a)$$

$$\delta_t = r_{t+1} + \gamma \max_{a'} Q(s_{t+1},a') - Q_t(s_t,a_t)$$

Peng's $Q(\lambda)$

- Disadvantage to Watkins' method:
  - Early in learning, the eligibility trace will be "cut" (zeroed out) frequently resulting in little advantages to traces
- **Peng**:
  - Backup max action except at end
  - Never cut traces
- Disadvantage:
  - Complicated to implement



Relationship Between DP and TD

Marcello Restelli

Full Backup (DP)	**Sample backup (TD)**
Iterative Policy Evaluation $V(s) \leftarrow \mathbb{E}_\pi[r + \gamma V(s')\|s]$	TD Learning $V(s) \overset{\alpha}{\leftarrow} r + \gamma V(s')$
Q–Policy Iteration $Q(s,a) \leftarrow \mathbb{E}_\pi[r + \gamma Q(s',a')\|s,a]$	SARSA $Q(s,a) \overset{\alpha}{\leftarrow} r + \gamma Q(s',a')$
Q–Value Iteration $Q(s,a) \leftarrow \mathbb{E}_\pi[r + \gamma \max_{a' \in \mathcal{A}} Q(s',a')\|s,a]$	Q–learning $Q(s,a) \overset{\alpha}{\leftarrow} r + \gamma \max_{a' \in \mathcal{A}} Q(s',a')$