# RL 7: Eligibility Traces

Michael Herrmann

University of Edinburgh, School of Informatics

04/02/2014

# Last time: TD (0) [Points to remember]

- TD(0) provides estimates of the value function in Markovian reward processes
- If the learning rate obeys the Robbins-Monro condition: $\sum_{t=0}^{\infty} \eta_t = \infty$, $\sum_{t=0}^{\infty} \eta_t^2 < \infty$, then the stochastic approximation finds a deterministic solution with probability 1.
- TD(0)

$$\delta_{t+1} = r_t + \gamma \hat{V}_t(s_{t+1}) - \hat{V}_t(s_t)$$

$$\hat{V}_{t+1} = \begin{cases} \hat{V}_t(s) + \eta\delta_{t+1} & \text{if } s = s_t \\ \hat{V}_t(s) & \text{otherwise} \end{cases}$$

- Today: $TD(\lambda)$ interpolates between $TD(0)$ and MC, but can be used as well in non-episodic tasks
- Appropriate tuning of $\lambda$ (and $\varepsilon$, $\gamma$) can lead to a significantly faster convergence than MC or TD(0).

# Monte-Carlo: Learning from Experience

- Random sampling to obtain numerical results
  - Define a domain of possible inputs
  - Generate inputs randomly from a probability distribution over the domain
  - Perform a deterministic computation on the inputs
  - Aggregate the results

Wikipedia

- Applied to find solutions to problems that are not analytically or deterministically numerically solvable.
- Use information collected along a whole trajectory (requires episodic tasks)

# TD, DP, MC: Bootstrapping and Sampling

TD **bootstraps**: It updates its estimates of $V$ based on other estimates of $V$

DP (dynamic programming) also bootstraps

MC does not bootstrap: Estimates of complete returns are made at the end the episode

TD **samples**: Its updates are based on one path through the state space

MC also samples

DP does not sample: Its updates are based on all actions and all states than can be reached from the updating state

Examples: See e.g. random walk example S+B sect. 6.2

# Every-visit Monte-Carlo algorithm

- Episodic problem, the $k$-th episode starting at time $T_k$
- The starting state of each episode $s_{T_k}$ is drawn from a distribution $\pi_0$.
- Reward within episode $k+1$

$$R_t = \sum_{s=t}^{T_{k+1}-1} \gamma^{s-t} r_{s+1} \text{ for } t \in [T_k, T_{k+1})$$

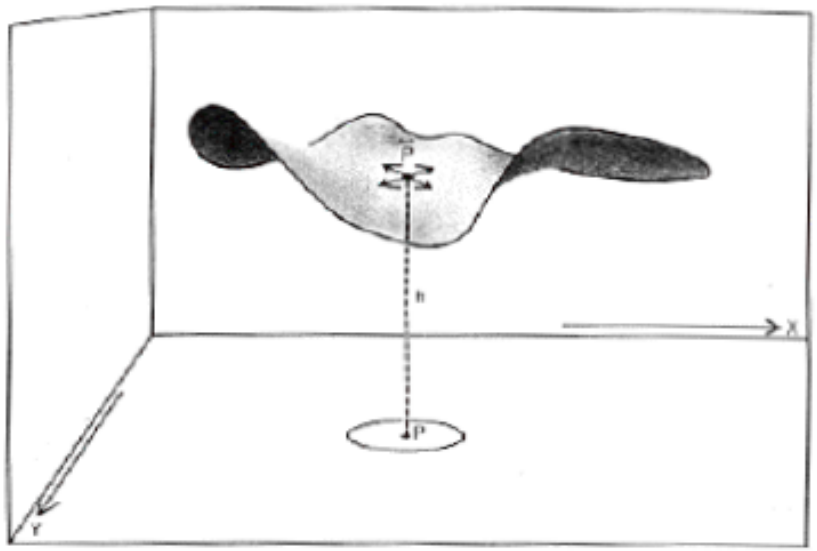
- Obviously, $V(s) = E[R_t | x_t = x]$ (if $x_t$ has non-zero prob.)

$$\hat{V}_{t+1}(s) = \hat{V}_t(s) + \eta_t \left( R_t - \hat{V}_t(s) \right) \mathbb{I}_{\{s_t = s\}}$$

- Uses multi-step predictions $\Longrightarrow$ multi-step method
- Also a stochastic approximation algorithm

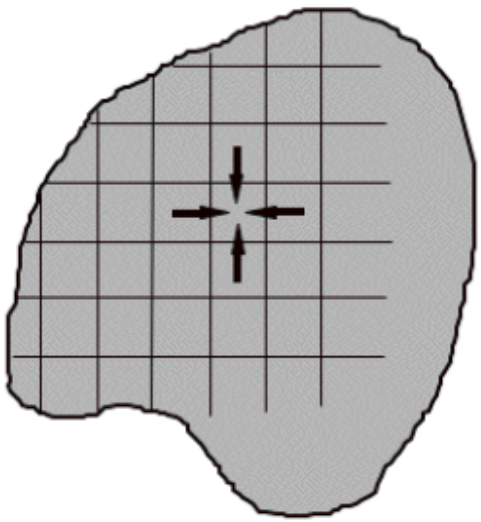
- Old mathematics problem [Dirichlet]:
  - Given function $f$ that has values everywhere on the boundary of a region in $\mathbb{R}^n$
  - Is there a unique continuous function $u$ twice continuously differentiable in the interior and continuous on the boundary, such that $u$ is harmonic in the interior and $u = f$ on boundary?

- In layman's terms: What is the shape of a soap bubble in a warped ring?



# Two Approaches for Soap Bubble Example

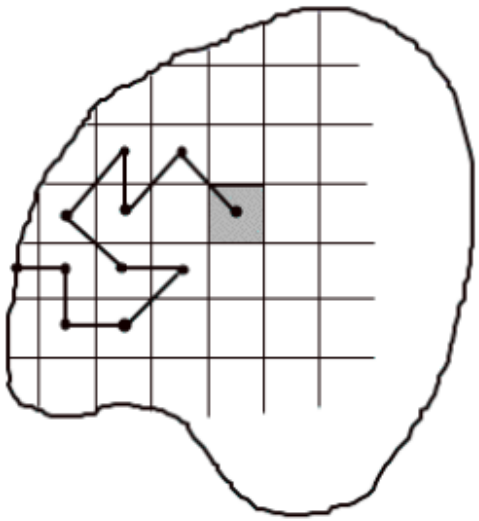
## Relaxation (e.g., finite-difference)



$$\phi_{xx}(x_i, y_j) = [u(x_{i+1}, y_j) - 2u(x_i, y_j) + u(x_{i-1}, y_j)]/(\Delta x)^2$$

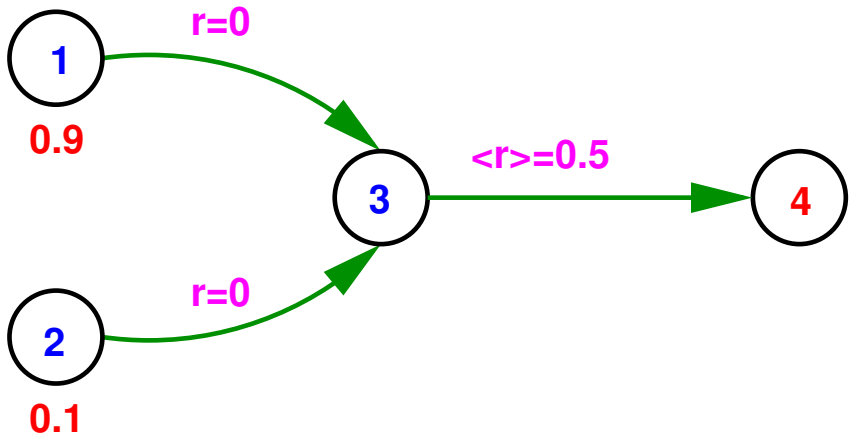
$$\phi_{yy}(x_i, y_j) = [u(x_i, y_{j+1}) - 2u(x_i, y_j) + u(x_i, y_{j-1})]/(\Delta y)^2$$

## Monte Carlo Method



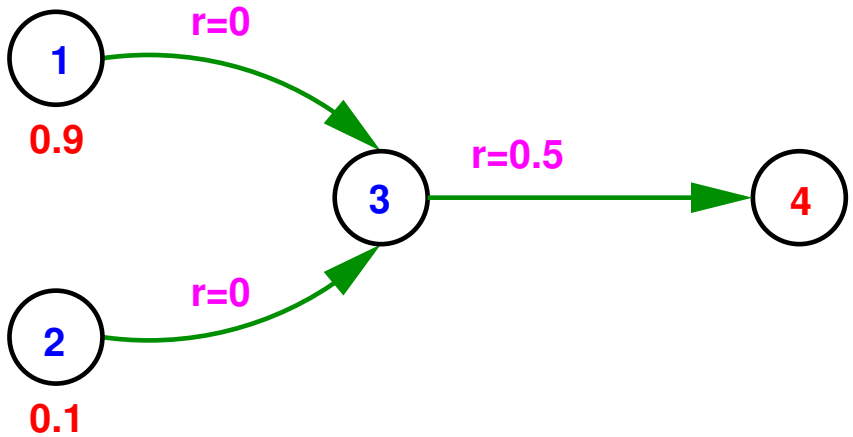
Approximate from sample paths (from interior/boundary)

# TD(0) or MC? (illustrative examples)




- Initial states: 1 and 2; episode end at state 4
- Episodes start more often in 1 than in 2
- For every $k$ visits of state 2, state 3 is visited $10k$ times
- For TD(0) update is averaging rewards when leaving state 3
- This is faster than update of 2, so 2 can follow this average
- For MC the variance of $R_t$ remains finite, so learning at 2 is slower

# TD(0) or MC? (illustrative examples)

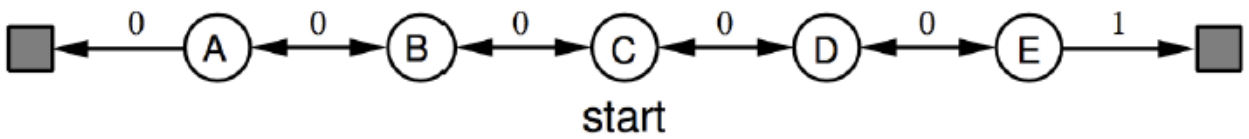



- Now with a fixed reward $r = 0.5$ when leaving state 3
- For TD(0) update is averaging rewards when leaving state 3
- Now 2 follows this average but on a slower time scale
- For MC the value at 2 is updated for each episode, i.e. it learns together with state 3

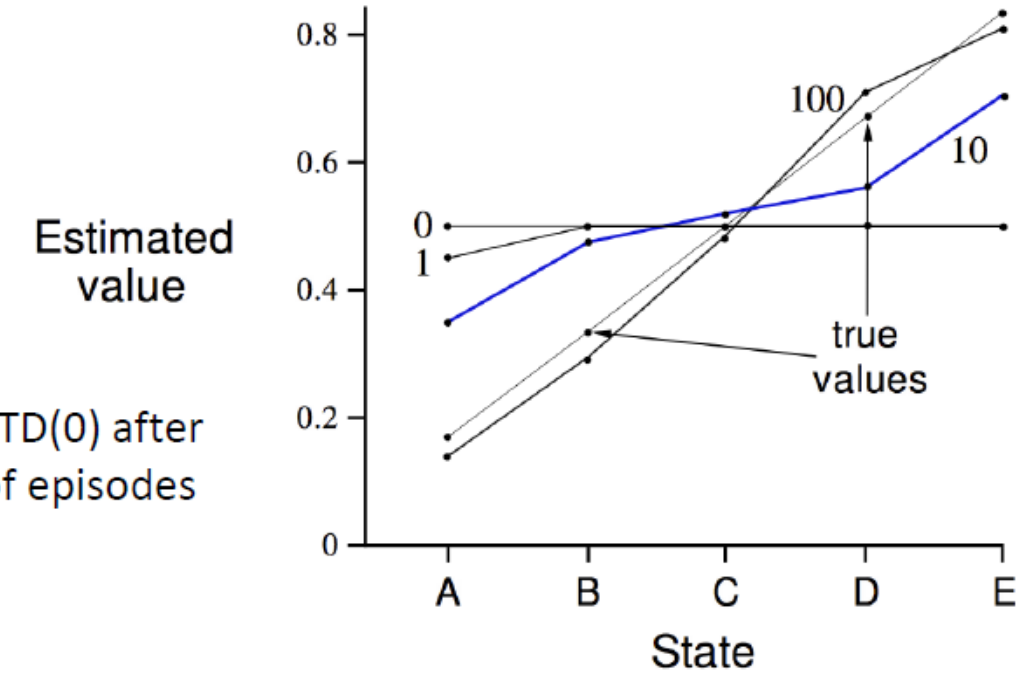
Conclusion: If the information in the reward signal is unreliable or even misleading, then TD(0) can be beneficial, but it may not be efficient $\Longrightarrow$ Unify MC and TD(0)

See Sutton, 1988 and Belezanay et al. 1999

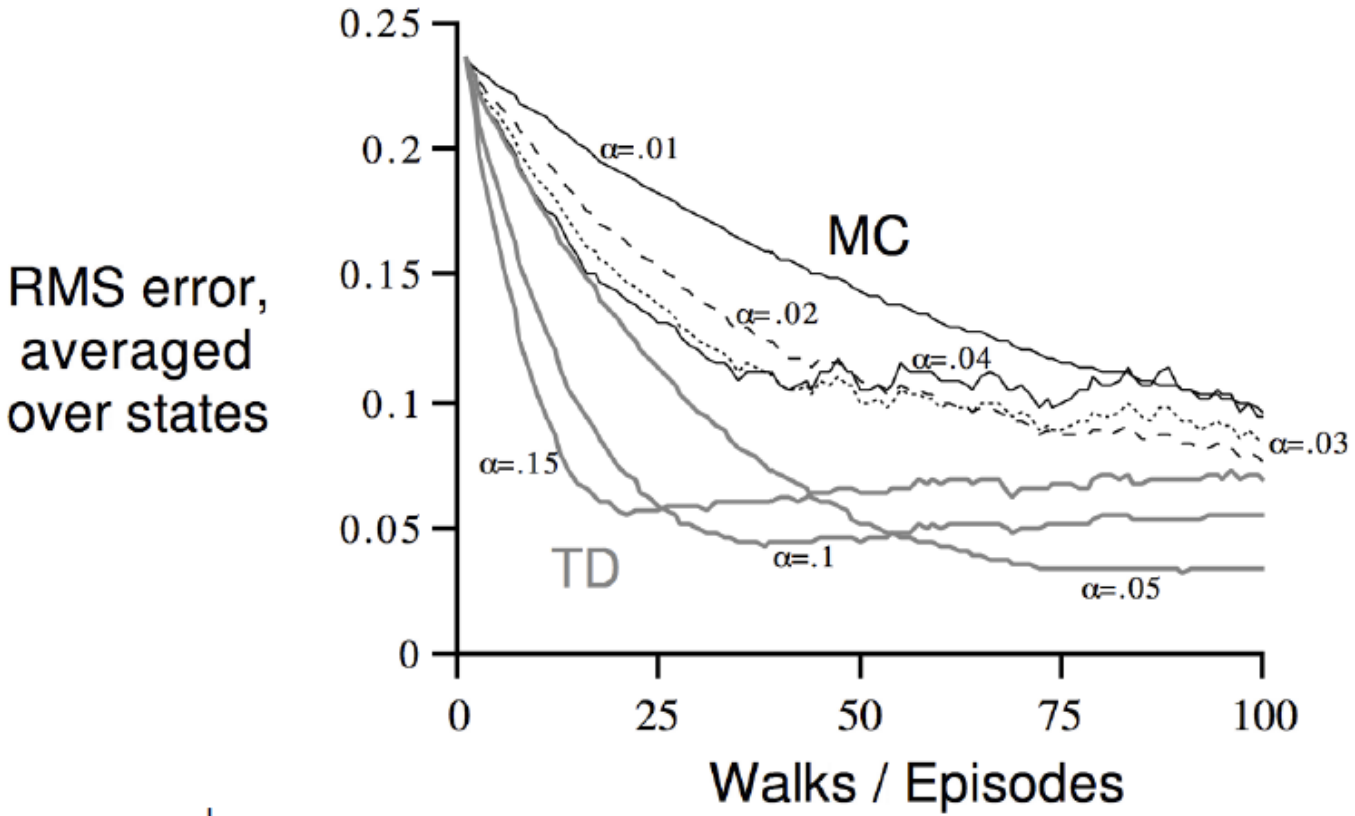
# Random walk example S+B sect. 6.2




Values learned by TD(0) after various numbers of episodes



# Random walk example S+B sect. 6.2



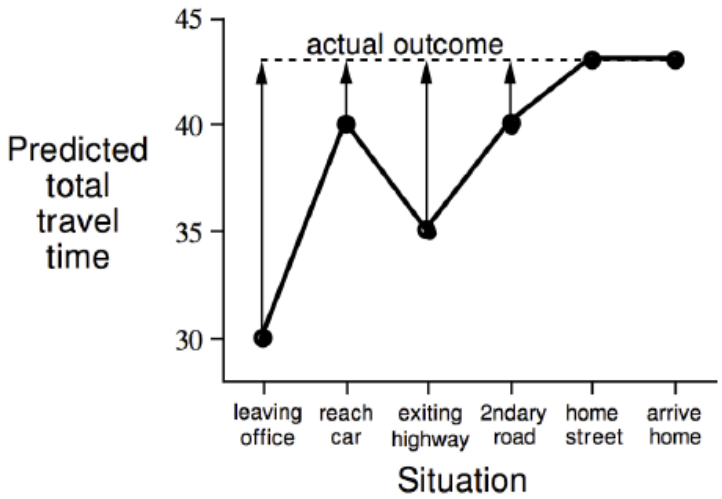

Data averaged over
100 sequences of episodes

# Example: Driving home

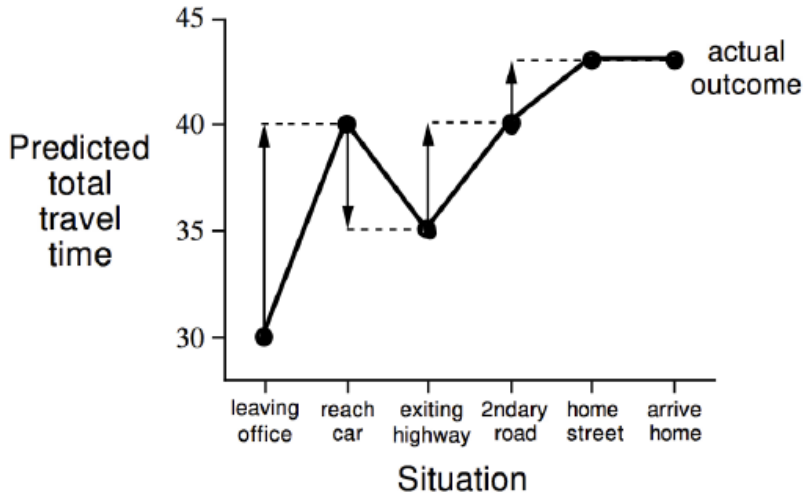
| State | Elapsed Time (minutes) | | Predicted Time to Go | Predicted Total Time |
|---|---|---|---|---|
| leaving office | 0 | | 30 | 30 |
| reach car, raining | 5 | (5) | 35 | 40 |
| exit highway | 20 | (15) | 15 | 35 |
| behind truck | 30 | (10) | 10 | 40 |
| home street | 40 | (10) | 3 | 43 |
| arrive home | 43 | (3) | 0 | 43 |

# Comparison: MC vs. TD

Expected time (= cost = neg. reward) to arrive home:




Changes recommended by MC (once at the end of episode)




Changes recommended by TD (updating one value per step)

# Eligibility Traces for TD

- TD learning can often be accelerated by the addition of *eligibility traces*.
- TD≡TD(0) updates the value function only for the immediately preceding state
- But $r_{t+1}$ provides useful information for learning earlier predictions as well
- Extend TD by eligibility traces:
  - Short-term memory of many previous state that are updated (though to a lesser extend the farther they are back)
  - Eligibility traces are usually implemented by an exponentially decaying memory trace, with decay parameter $\lambda$.
  - TD should thus be more specifically TD($\lambda$) with $0 \leq \lambda \leq 1$
  - Eligibility extends less wide into the past than the time horizon towards the future that is controlled by the discount rate.
  - TD(1) updates all the preceding predictions equally (for $\gamma = 1$)

www.scholarpedia.org/article/Temporal_difference_learning

# TD($\lambda$) [i.e. with eligibility traces]

TD(0):

$$\hat{V}_{t+1}(s) = \begin{cases} \hat{V}_t(s) + \eta\delta_{t+1} & \text{if } s = s_t \\ \hat{V}_t(s) & \text{otherwise} \end{cases}$$

TD($\lambda$):

$$\hat{V}_{t+1}(s) = \hat{V}_t(s) + \eta\delta_{t+1} e_{t+1}(s)$$

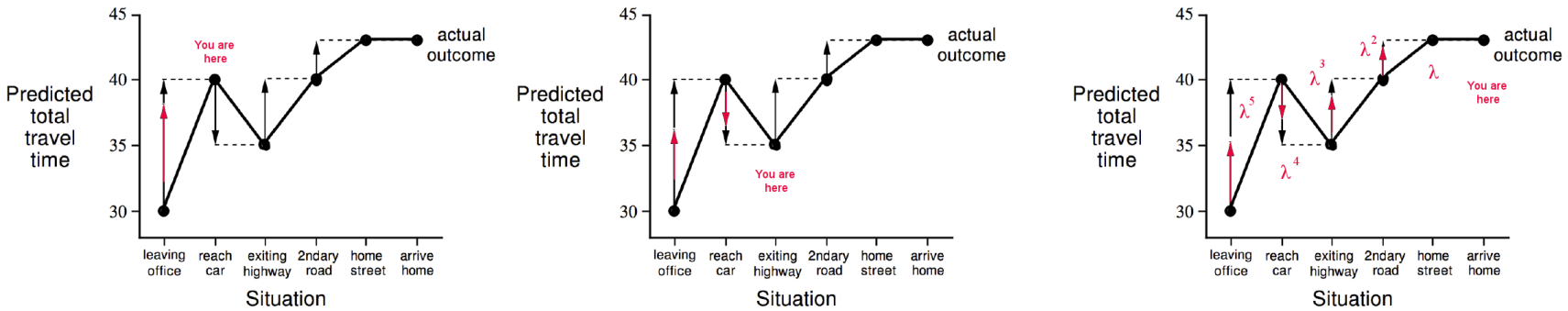
where $e$ is a (replacing) eligibility trace with parameter[s] $\lambda$ [and $\gamma$]

i.e. the value of all recently visited states is changed into the same direction.

$\lambda$ is called *trace decay parameter*

1-D maze example is solved essentially when the goal was found once (it may take a bit more time to full convergence). Homework: Test this numerically!

# Eligibility Traces

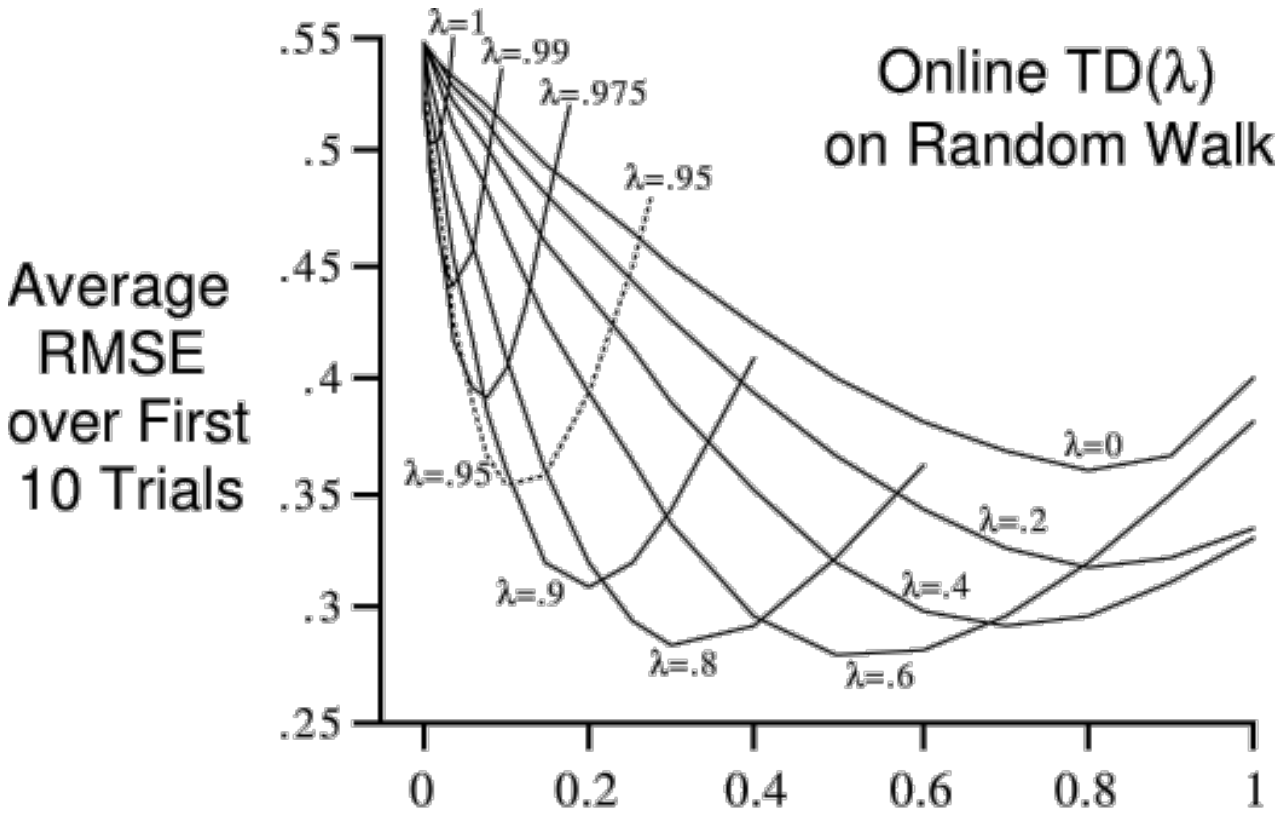



...

Changes when using TD with eligibility traces.

(This illustration is not to scale and does not reflect the accumulation of the changes)

# 19-step random walk task (s. Barto and Sutton)




Learning rate

# Eligibility Traces

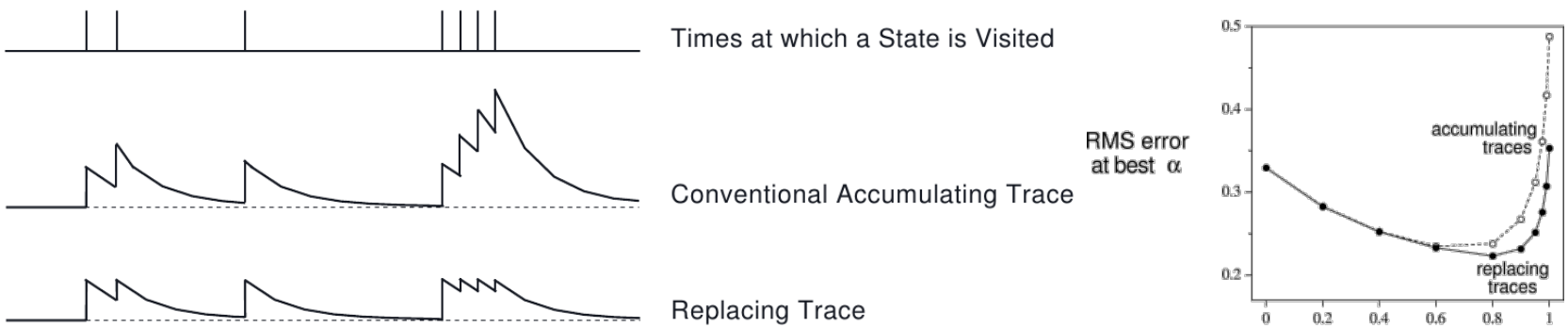
TD($\lambda$): Unification of TD (update only value of previous state) and MC (update all states since start) using eligibility traces

Accumulating eligibility trace (e.g. in certain dynamical problems)

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

Replacing eligibility trace (e.g. in a maze)

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




S.P. Singh & R.S. Sutton. Reinforcement learning with replacing eligibility traces. Rec. Adv. in RL 1996

# Summary

- TD($\lambda$) provides efficient estimates of the value function in Markovian reward processes
- It generalises MC methods, but can be used as well in non-episodic tasks
- Appropriate tuning of $\lambda$ (and $\gamma$) can lead to a significantly faster convergence than MC or TD(0).


## Acknowledgements

Many slides are adapted from web resources associated with Sutton and Barto's Reinforcement Learning book

... before being used by Dr. Subramanian Ramamoorthy in this course in the last three years.

... actually, today I used some of my own material and followed the book *Algorithms for Reinforcement Learning* by C. Szepesvari, Chapters 2.1 and 4.1.

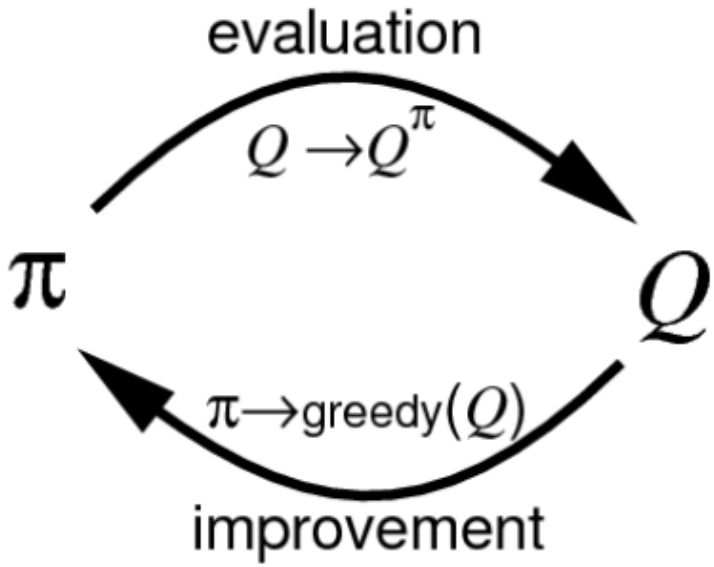
# Monte Carlo Control

- Policy Evaluation:

  Monte Carlo method

- Policy Improvement:

  Greedify with respect to state-value of action-value function



# Convergence of MC

- Policy improvement still works if evaluation is done with MC

$$\begin{aligned} Q^{\pi_k}\left(s, \pi_{k+1}(s)\right) &= Q^{\pi_k}\left(s, \arg\max_a Q^{\pi_k}(s,a)\right) \\ &= \max_a Q^{\pi_k}(s,a) \\ &\geq Q^{\pi_k}\left(s, \pi_k(s)\right) \\ &= V^{\pi_k}(s) \end{aligned}$$

- The expected reward for $\pi_{k+1}$ not worse than $\pi_k$
- We have to assume that the value function has stabilised, i.e. an infinite number of episodes

# Policy iteration with MC

- Initialise $\mathcal{Q}(s,a)$, $\pi(s)$ arbitrary $\forall s, a$; $R_{\text{List}}(s,a)$ empty list
- Repeat
  - Generate an episode using exploring starts and $\pi$
  - for each pair $s$, $a$ in the episode
    - $R :=$ return following the first occurrence of $s$ and $a$
    - Append $R$ to $R_{\text{List}}(s,a)$
    - $\mathcal{Q}(s,a) :=$ average over $R_{\text{List}}(s,a)$
  - $\forall s$ in the episode: $\pi(s) := \arg\max_a \mathcal{Q}(s,a)$

# $\mathcal{Q}(\lambda)$ algorithm (s. Barto and Sutton)

Initialize $Q(s,a)$ arbitrarily and $e(s,a)=0$, for all $s,a$
Repeat (for each episode):
  Initialize $s$, $a$
  Repeat (for each step of episode):
    Take action $a$, observe $r$, $s'$
    Choose $a'$ from $s'$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
    $a^* \leftarrow \arg\max_b Q(s',b)$ (if $a'$ ties for the max, then $a^* \leftarrow a'$)
    $\delta \leftarrow r + \gamma Q(s',a^*) - Q(s,a)$
    $e(s,a) \leftarrow e(s,a) + 1$
    For all $s,a$:
      $Q(s,a) \leftarrow Q(s,a) + \alpha\delta e(s,a)$
      If $a' = a^*$, then $e(s,a) \leftarrow \gamma\lambda e(s,a)$
                    else $e(s,a) \leftarrow 0$
    $s \leftarrow s'$; $a \leftarrow a'$
  until $s$ is terminal

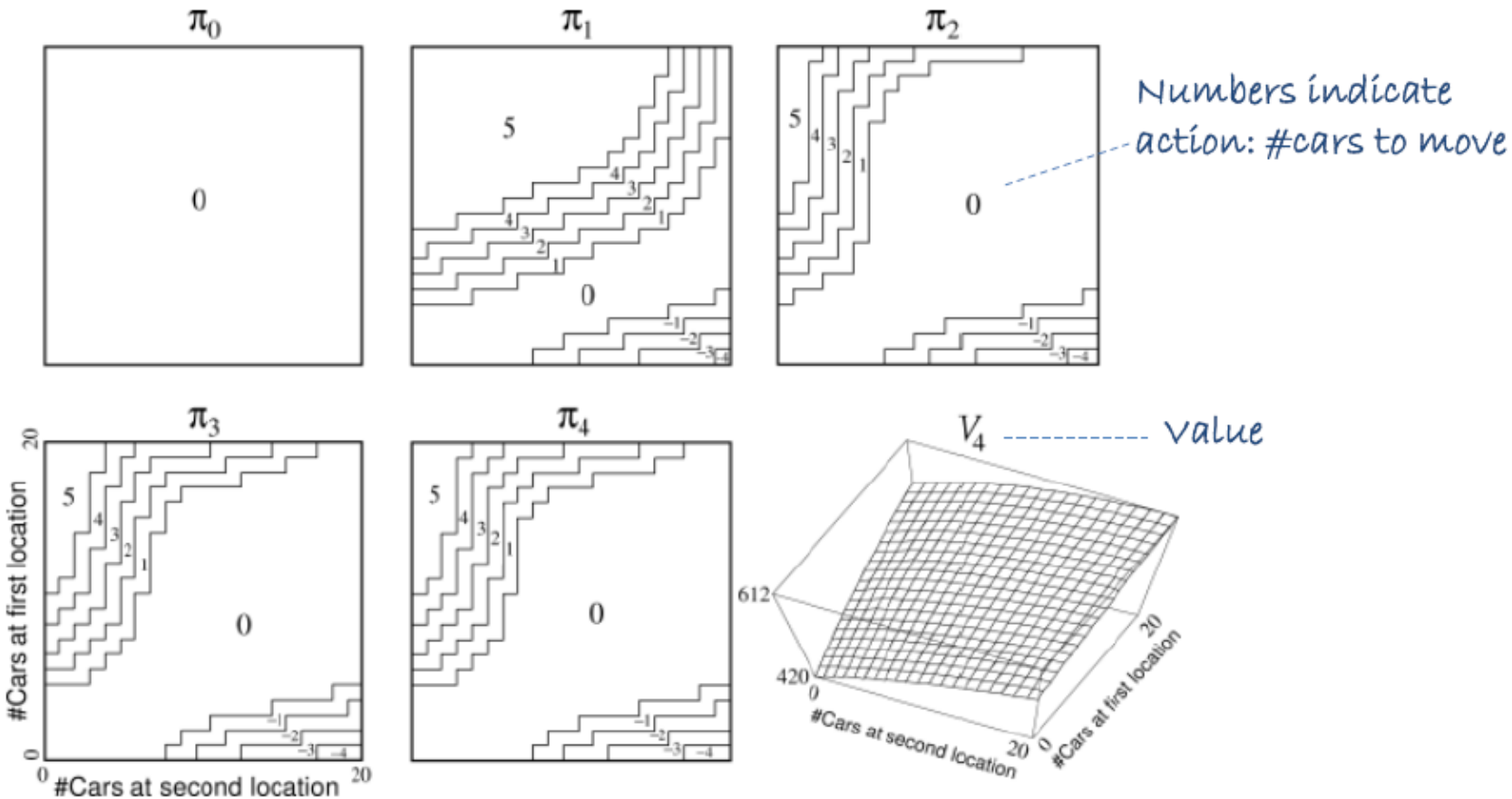
# Example I: Jack's car rental (4.2 in S+B)

- $10 for each car rented (must be available when request received)
- Two locations, maximum of 20 cars at each
- Cars returned and requested randomly
  - Poisson distribution, $n$ returns/requests with probability $\frac{\lambda^n}{n!}e^{-\lambda}$
  - Location 1: Average requests = Average returns = 3
  - Location 2: Average requests = 4, Average Returns = 2
- Can move up to 5 cars between locations overnight (costs $2 each)

Problem setup:

- States, actions, rewards?
- Transition probabilities?

# Jack's car rental: Solution



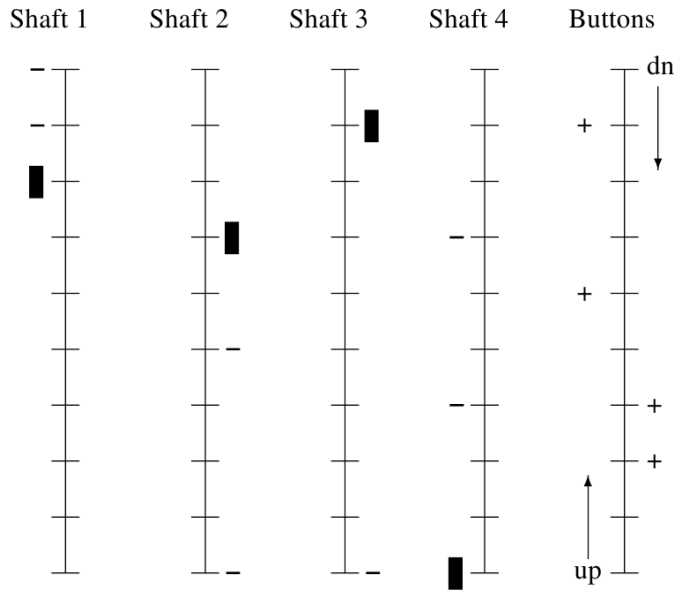
# Jack's car rental: Discussion

Difficult to adapt the solution to different conditions, e.g.

- Suppose first car moved is free but all other transfer cost \$2
  - From location 1 to location 2 (not other direction!)
  - Because an employee would anyway go in that direction, by bus
- Suppose only 10 cars can be parked for free at each location
  - More than 10 incur fixed cost of \$4 for using an extra parking lot

For more information see: cns.upf.edu/dani/materials/jack.pdf

# Example II: Elevator control: State space and reward signal




- $2^{18}$ possible combinations of the 18 hall call buttons (only one at top and bottom)
- $2^{40}$ possible combinations of the 40 car buttons
- $18^4$ possible combinations of positions and directions of cars: up/down to target floor

$2\hat{}18 \cdot 2\hat{}40 \cdot 18\hat{}4 \approx 10^{22}$ states

Formulation of goal:

- Minimise the average wait time
- Minimise the average system time (wait time plus travel time)
- Minimise the percentage of passengers that wait longer than some dissatisfaction threshold (usually 60 seconds)
- Minimise squared wait time (combination of first and third)

A. Barto, R. H. Crites. Improving elevator performance using RL. NIPS 8 (1996) 1017-1023.

# Elevator control: Continuous time problem

Modelled as discrete event systems, but the amount of time between events is a real-valued variable. A constant discount factor $\gamma$ is thus inadequate.

Use variable discount factor for cost $c_\tau$ (here minimised, instead of maximised reward)

$$\int_0^\infty e^{-\rho\tau} c_\tau d\tau \text{ instead of discrete version } \sum_{t=0}^{\infty} \gamma^t c_t$$

where $\rho \gtrsim 0$ is an inverse time scale corresponding to $1-\gamma$.
Now, for events at $t_x$ and $t_y$ the learning rule becomes

$$\Delta\hat{\mathcal{Q}}(s,a) = \eta\left(\int_{t_x}^{t_y} e^{-\rho(\tau-t_x)} c_\tau d\tau + e^{-\rho(t_y-t_x)} \min_b \hat{\mathcal{Q}}(u,b) - \hat{\mathcal{Q}}(s,a)\right)$$

Learning time 60,000 h of simulated elevator time (4 d @ 100 MIPS)

R.H. Crites and A.G. Barto, Elevator group control using multiple RL agents, Machine Learning 33 (1998) 235–262. (covered here only in part)

# Elevator control: Evaluation and practical issues

Results

- shown to "*outperform all of the elevator algorithms with which we compared them*"
- performance is restricted by certain rules (e.g. no reversals during tours)
- today's hybrid algorithms perform better

Conclusions

- "One of the greatest difficulties in applying RL to the elevator control problem was finding the correct temperature and step-size parameters"
- "The importance of focusing the experience of the learner onto the most appropriate areas of the state space cannot be overstressed" [represented functions by a neural network]
- RL algorithms can learn from actual (or simulated) experience and solve realistic stochastic dynamic optimisation problems

R.H. Crites and A.G. Barto. Elevator group control using multiple RL agents. Machine Learning 33 (1998) 235-262. (covered here only in part)