

Reinforcement Learning II

Policy Optimization and Model-free predictions

Antoine SYLVAIN

EPITA

2022

Contents

- 1 Optimal policies
- 2 Dynamic Programming
- 3 Model-free predictions

Policies

- Specifies the behaviour of the agent
- A **deterministic policy** associates an action to each state
- It can be represented by a table

s_0	a_1
s_1	a_0
s_2	a_1

- A **stochastic policy** associates various actions with a probability for each state
- The sum of probabilities of the actions for a given state equals 1

Policies

- Deterministic policy:

$$\pi(s) = a$$

- Stochastic policy:

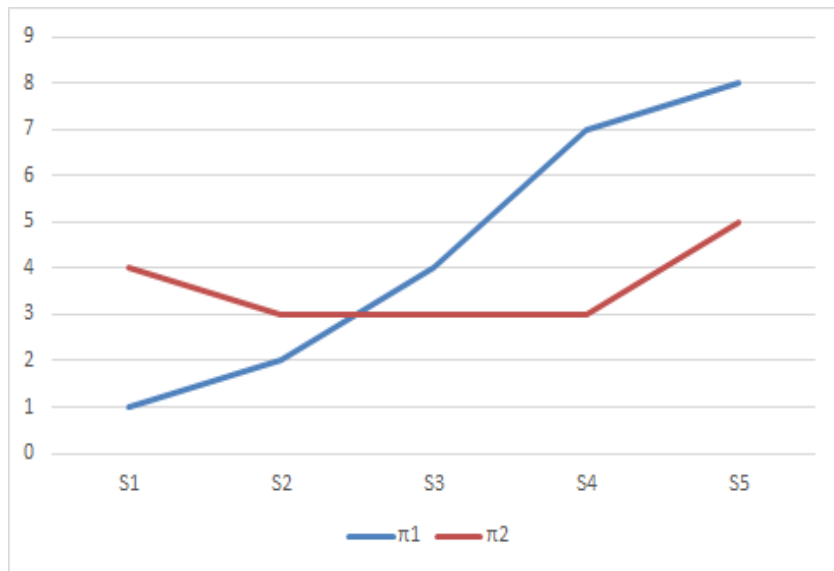
$$\pi(a|s) = \mathbb{P}[\mathcal{A}_t = a | \mathcal{S}_t = s]$$

Policies

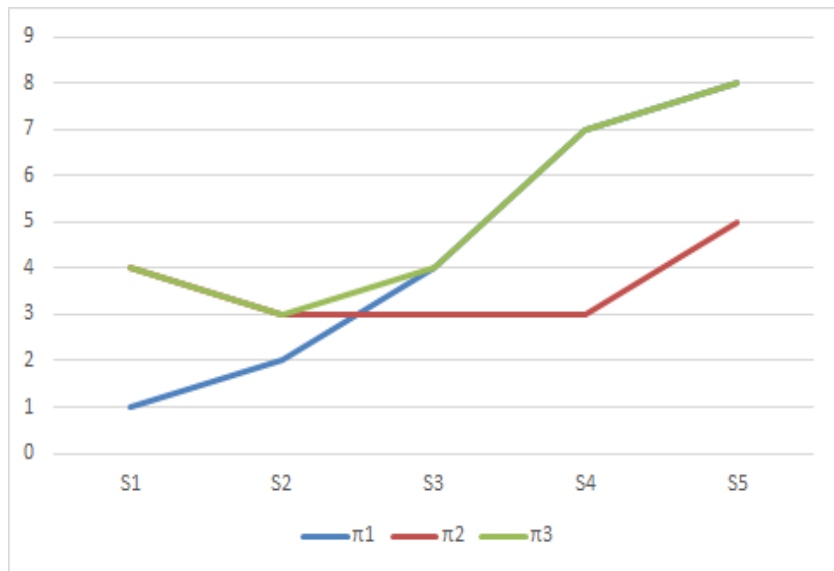
The best policy is the one that achieves the highest value

Is there always an optimal policy ? (i.e. that returns a value greater or equal than the other policies for each state)

Comparing two policies



Combining two policies



Contents

- 1 Optimal policies
- 2 Dynamic Programming**
- 3 Model-free predictions

What is Dynamic Programming

- A collection of algorithms that can be used to compute optimal policies
- They require a perfect model of the environment as a MDP
- Optimal substructures
- Divide the problem into sub-problems
- Combine the solutions to the sub-problems

Policy Evaluation and Control

- **Policy Evaluation:** Evaluation of the value function of a given policy
- **Policy Control:** Finding the best policy (i.e. maximize value function)
- Control is the overall objective
- But, to find the best policy, you first need to be able to evaluate how good it is

Iterative Policy Evaluation

- Starts with arbitrary values for each non-terminal state
- At each iteration, update the values of these states with the Bellman equation and the updated values
- Evaluations converge to the value function

Iterative Policy Evaluation

Input: π (the policy to be evaluated)

Initialize an array $V(s) = 0$ for all $s \in \mathcal{S}^+$

Repeat:

$\Delta \leftarrow 0$

For each $s \in \mathcal{S}$:

$v \leftarrow V(s)$

$V(s) \leftarrow \sum_a \pi(a|s) \sum_{s', r} p(s', r|s, a)[r + \gamma V(s')]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until $\Delta < \theta$ (small positive number)

Output: $V \approx v_\pi$

Example

- Small grid world
- Four possible actions (up, down, left, right)
- Reward is -1 until a terminal state is reached
- Uniform random policy (0.25 for each action)
- Undiscounted ($\gamma = 1$)

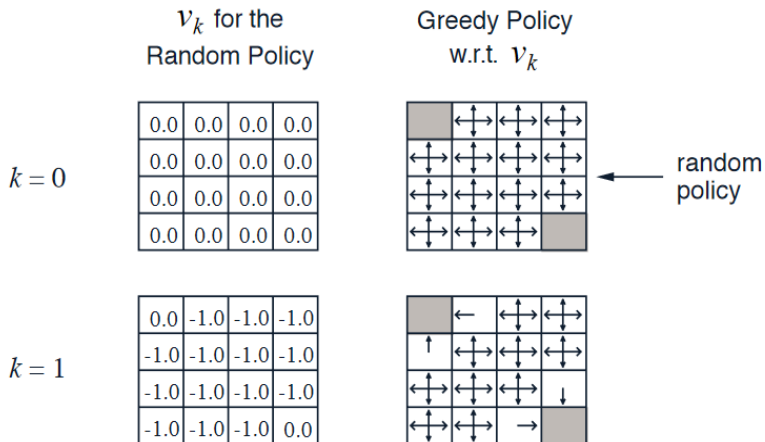
Example



	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

$R = -1$
on all transitions

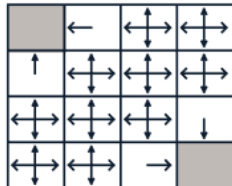
Example



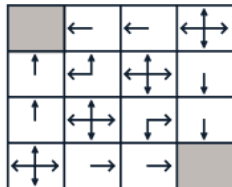
Example

 $k = 1$

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0


 $k = 2$

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0



Example

 $k = 2$

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0

	←	←	↕
↑	↖	↕	↓
↑	↕	↘	↓
↕	→	→	

 $k = 3$

0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

	←	←	↘
↑	↖	↘	↓
↑	↗	↘	↓
↖	→	→	

Example

 $k = 3$

0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

	←	←	↖
↑	↖	↖	↓
↑	↖	↖	↓
↖	→	→	

 $k = 10$

0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0

	←	←	↖
↑	↖	↖	↓
↑	↖	↖	↓
↖	→	→	

 $k = \infty$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

	←	←	↖
↑	↖	↖	↓
↑	↖	↖	↓
↖	→	→	

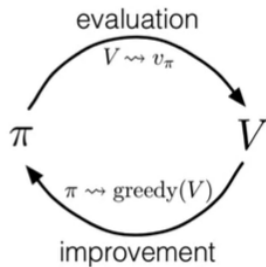
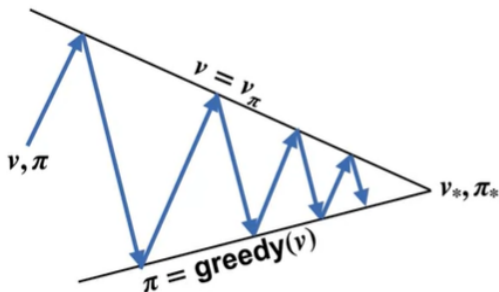
optimal policy

How to improve a policy ?

- Given a policy π
 - Evaluate the policy π
$$v_{\pi}(s) = \mathbb{E}_{\pi}[\mathcal{R}_{t+1} + \gamma\mathcal{R}_{t+2} + \gamma^2\mathcal{R}_{t+3} + \dots | \mathcal{S}_t = s]$$
 - Improve the policy by acting greedily with respect to v_{π}
$$\pi' = greedy(v_{\pi})$$
- In the Small Gridworld example, the improved policy was optimal, so $\pi' = \pi^*$
- But in most cases, we will need more iterations of evaluation/improvement
- Nonetheless, this process of policy evaluation always converge to π^*

Policy Iteration

Policy Iteration



Policy iteration for estimating $\pi \approx \pi^*$

Initialization: $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s) \forall s \in \mathcal{S}$

Policy Evaluation

Repeat:

$$\Delta \leftarrow 0$$

For each $s \in \mathcal{S}$:

$$v \leftarrow V(s)$$

$$V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$$

$$\Delta \leftarrow \max(\Delta, |v - V(s)|)$$

until $\Delta < \theta$ (small positive number)

Policy Improvement

policy-stable \leftarrow True

For each $s \in \mathcal{S}$:

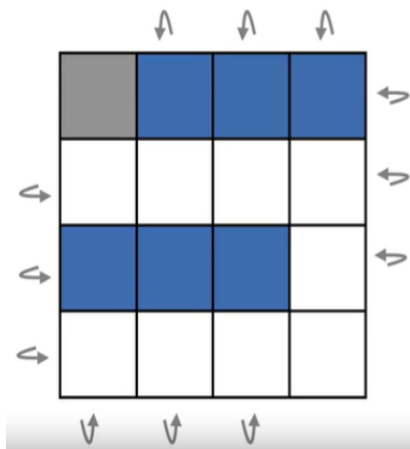
$$\text{old-action} \leftarrow \pi(s)$$

$$\pi(s) \leftarrow \operatorname{argmax}_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$$

If old-action $\neq \pi(s)$, then policy-stable \leftarrow False

If policy-stable, return $V \approx v^*$ and $\pi \approx \pi^*$, else **Policy Evaluation**

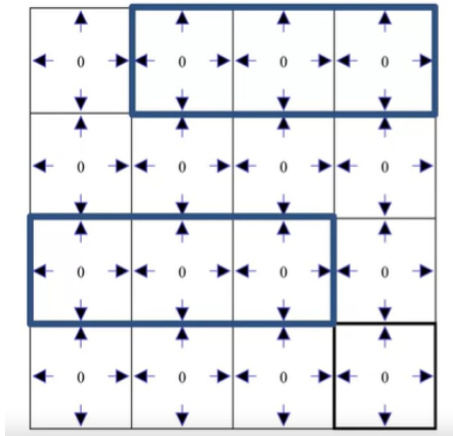
Example



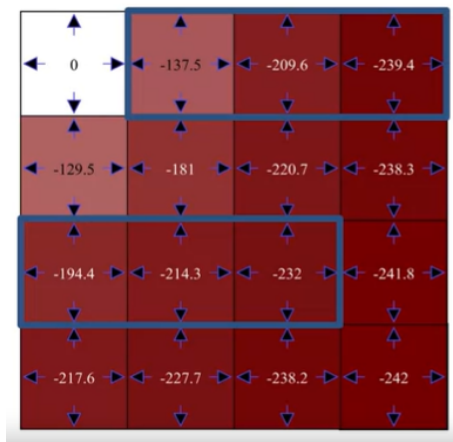
$$R = \begin{cases} -10 & \text{Blue states} \\ -1 & \text{Other states} \end{cases}$$

$$\gamma = 1$$

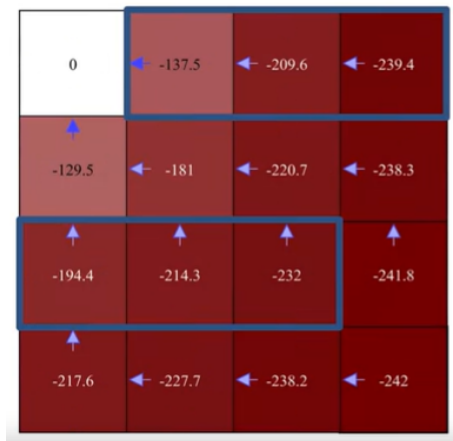
Example



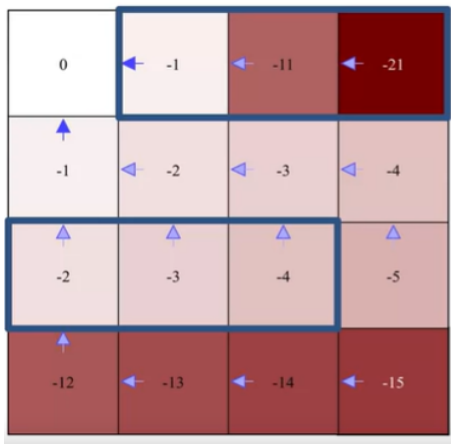
Example



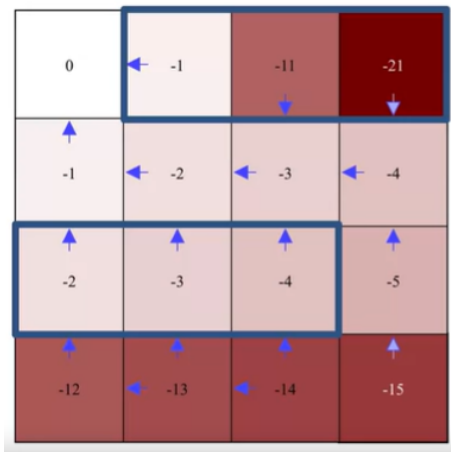
Example



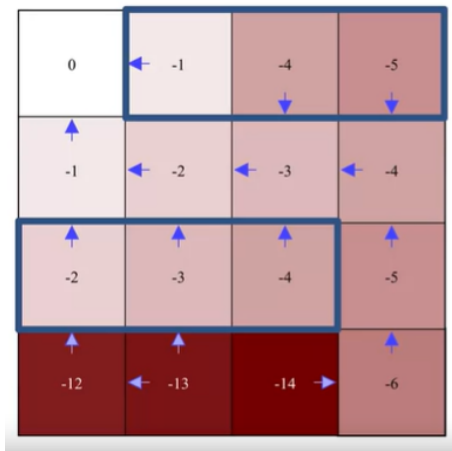
Example



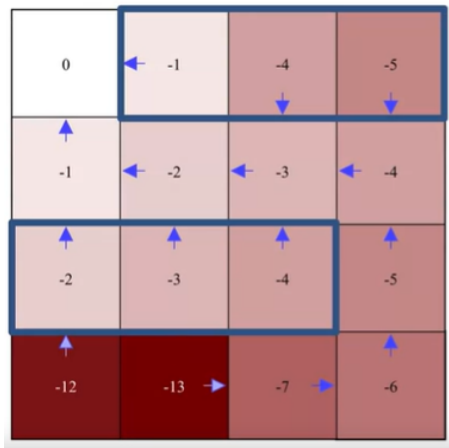
Example



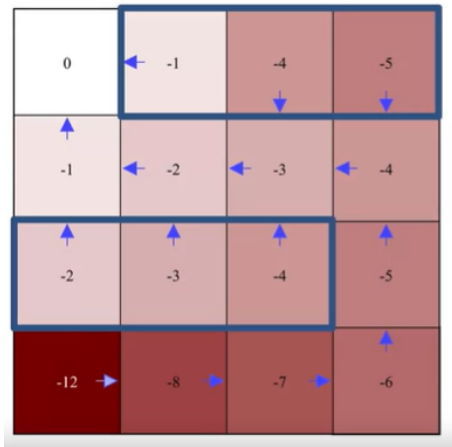
Example



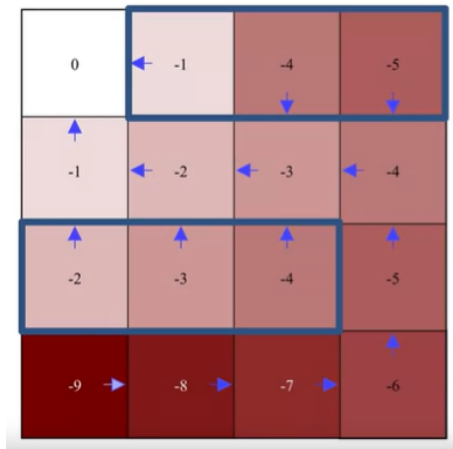
Example



Example



Example



Contents

- 1 Optimal policies
- 2 Dynamic Programming
- 3 Model-free predictions

Model-free algorithms

- Dynamic Programming enables us to find optimal policies in a Markov Decision Process
- But it has a costly pre-requisite : we need a complete knowledge of the MDP
- What if we have a limited knowledge of the MDP ?
- Can we make predictions based on our experience ?

Monte-Carlo Learning

- Methods that learn directly from experience
- Model-free: we have no previous knowledge of the MDP transitions and reward
- Learns from complete episodes: no bootstrapping, all episodes must terminate
- Idea: $\text{value} = \text{mean return}$

Monte-Carlo Policy Evaluation

- Goal: learn v_π from episodes of experience under policy π :
 $\mathcal{S}_1, \mathcal{A}_1, \mathcal{R}_2, \dots, \mathcal{S}_k \sim \pi$
- Return is the total discounted reward :
 $\mathcal{G}_t = \mathcal{R}_{t+1} + \gamma \mathcal{R}_{t+2} + \dots + \gamma^{\tau-1} \mathcal{R}_\tau$
- Value function is the expected return :
 $v_\pi(s) = \mathbb{E}_\pi[\mathcal{G}_t | \mathcal{S}_t = s]$
- Instead of expecting return, Monte-Carlo uses **empirical mean**

First-visit Monte-Carlo Method

- Estimates the value $v_{\pi}(s)$ of state s under policy π
- Each occurrence of state s in an episode is called a visit to s
- The first occurrence of s is thus the first visit to s
- The first-visit MC-method estimates $v_{\pi}(s)$ as the average of the returns following the first visit to s :

$$V(s) = S(s)/N(s)$$

with

$S(s) \leftarrow S(s) + \mathcal{G}_t$, the incremented total return

$N(s)$, the visits counter

First-visit Monte-Carlo Prediction

Input: π , a policy to be evaluated

Initialization:

$V(s) \in \mathbb{R} \forall s \in S$, arbitrarily

$Returns(s) \leftarrow$ an empty list, for all $s \in S$

Loop (for each episode):

Generate an episode following π : $S_0, A_0, R_1, \dots, R_T$

$\mathcal{G} \leftarrow 0$

Loop for each step of episode $t = T - 1, T - 2, \dots, 0$:

$\mathcal{G} \leftarrow \gamma \mathcal{G} + R_{t+1}$

Unless S_t appears in S_0, S_1, \dots, S_{t-1} :

Append \mathcal{G} to $Returns(s)$

$V(S_t) \leftarrow avg(Returns(S_t))$

Every-visit Monte-Carlo Method

Quite the same, but you evaluate the returns following every visit to s

Example of blackjack

- 200 states:
 - Current sum (12-21)
 - Dealer's visible card (ace-10)
 - Usable ace ? (True or False)
- 2 actions:
 - **Stick**: Stop receiving cards and terminate
 - **Twist**: Take another card
- Stick reward:
 - +1 if sum of cards $>$ sum of dealer
 - 0 if sum of card = sum of dealer
 - -1 if sum of card $<$ sum of dealer
- Twist reward:
 - -1 if sum of cards $>$ 21 (terminal)
 - 0 otherwise
- Transitions: automatically twist if sum of cards $<$ 12
- Policy: Stick if sum = 20 or 21, else twist

Example of blackjack



Example of blackjack



Example of blackjack

S	Returns(S)	V(S)
(Usable Ace, Sum, Dealer)		

Dealer **14**








Player **20**

Example of blackjack

S	Returns(S)	V(S)
(Usable Ace, Sum, Dealer)		

Dealer 23



Player 20 $R = 1$

Example of blackjack

S (Usable Ace, Sum, Dealer)	Returns(S)	V(S)
A	Returns(A) = [1]	1

A = (NoAce, 20, 10)



Example of blackjack

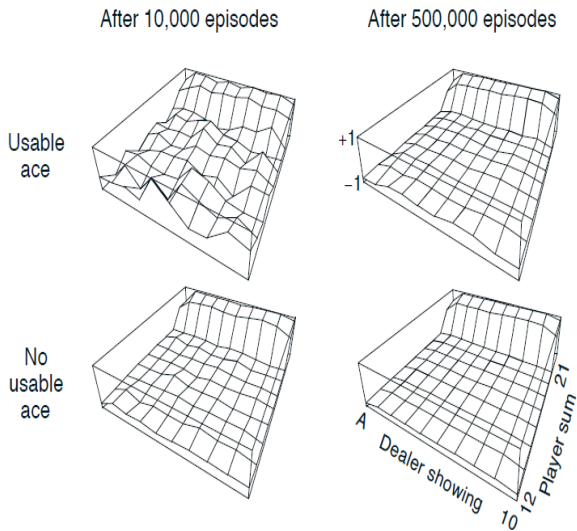
S (Usable Ace, Sum, Dealer)	Returns(S)	V(S)
A	Returns(A) = [1]	1
B	Returns(B) = [1]	1

A = (NoAce, 20, 10)

B = (NoAce, 13, 10)



Example of blackjack



Monte-Carlo Estimation of Action values

- If the model is not available, estimate (state-)action values rather than state values is particularly useful.
- With a model, like in dynamic-programming, state-value was enough. Without a model, it is not sufficient.
- So one of the primary goals of MC methods is to evaluate q^*
- The policy evaluation becomes an estimation of $q_\pi(s, a)$
- A state-action s, a is said to be visited in a an episode if state s is visited and action a is selected in it

Temporal-Difference Methods

- Learn directly from episodes of experience
- Model-free: we have no previous knowledge of the MDP transitions and reward
- Learn from incomplete episodes, by bootstrapping
- Updates a guess towards a guess

Temporal Difference Methods

- Whereas MC methods need to wait until the end of the episode to update the value of $V(S_t)$
- Temporal-Difference methods makes the update at step $t + 1$
- The simplest TD method is know as TD(0)
- $V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$
- α : stepsize parameter

TD(0) Algorithm

Input: π , the policy to be evaluated

Parameter: stepsize parameter $\alpha \in [0, 1]$

Initialization: $V(s) \forall s \in \mathcal{S}^+$, arbitrarily, except $V(\text{terminal}) = 0$

Loop for each episode:

$A \leftarrow$ action given by π for S

Select action A , observe R, S'

$V(S) \leftarrow V(S) + \alpha[R + \gamma V(S') - V(S)]$

$S \leftarrow S'$

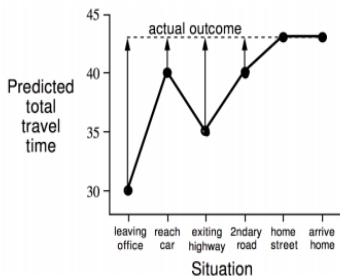
until S terminal

Example of driving home

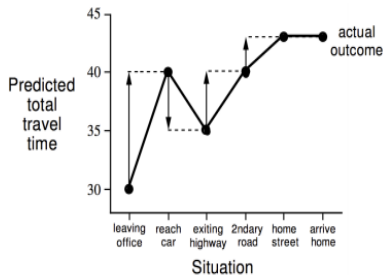
State	Elapsed Time (minutes)	Predicted Time to Go	Predicted Total Time
leaving office	0	30	30
reach car, raining	5	35	40
exit highway	20	15	35
behind truck	30	10	40
home street	40	3	43
arrive home	43	0	43

Example of driving home: MC vs TD

Changes recommended by
Monte Carlo methods ($\alpha=1$)



Changes recommended
by TD methods ($\alpha=1$)



Comparison of MC and TD

- TD can learn before knowing the final outcome
- TD can learn online step by step
- MC have to wait until the end of episode to know the return
- TD can learn without the final outcome
- TD can learn from incomplete sequences
- MC can only learn from complete sequences
- TD works in continuing (non-terminating) environments
- MC only works form episodic (terminating) environments

Comparison of MC and TD

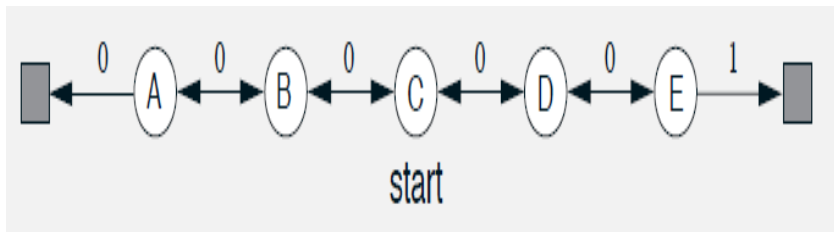
- MC return $G_t = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{\tau-1} R_\tau$ is an unbiased estimate of $v_\pi(S_t)$
- TD target $R_{t+1} + \gamma V(S_{t+1})$ is a biased estimate of $v_\pi(S_t)$
- TD target has much lower variance than the return:
 - MC return depends on many random actions, transitions, rewards
 - TD target depends on one random action, transition, reward

Comparison of MC and TD

- MC has high variance, no bias
 - Good convergence properties
 - Not very sensitive to initial value
 - Very simple understanding and use
 - TD has low variance, but some bias
 - Usually more efficient than MC
 - TD(0) converge to $v_{\pi}(S_t)$
 - More sensitive to initial value

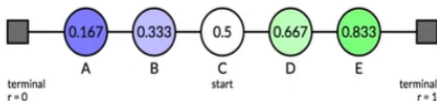
Random Walk Example

- All episodes start in the center state C
- Left and right states are terminal state
- Reward $+1$ if terminates on the right, else 0

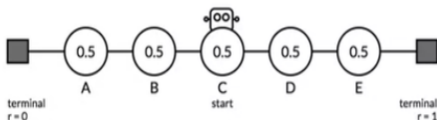


Random Walk Comparison

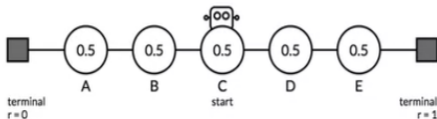
Target / Exact Values



Updates using TD Learning $V(S_t) \leftarrow V(S_t) + \alpha [R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$



Updates using Monte Carlo

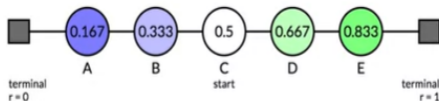


Random Walk Comparison

1st episode:

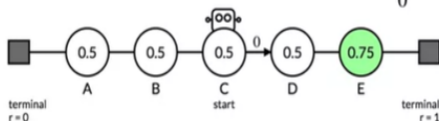
C,D,E,D,C,D,E,Right

Target / Exact Values

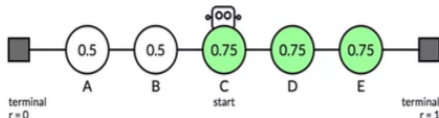


Updates using TD Learning

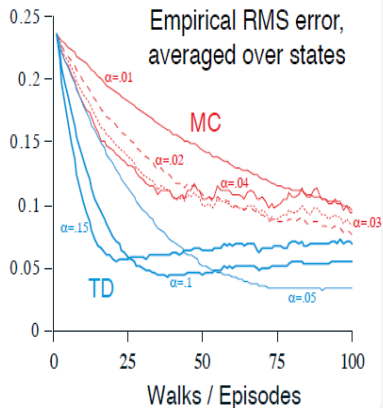
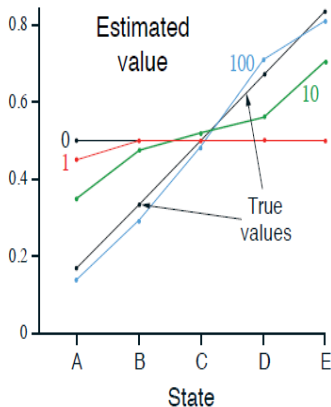
$$V(S_t) \leftarrow V(S_t) + \alpha [R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$



Updates using Monte Carlo



Random Walk Comparison



Batch MC and TD

- MC and TD converge
- But what about a batch experience of finite examples ?
- Repeatedly sample episodes
- Apply MC and TD(0) on these episodes

AB Example

Two states A and B, no discounting, 8 episodes of experience

B, 0

B, 1

B, 1

B, 1

B, 1

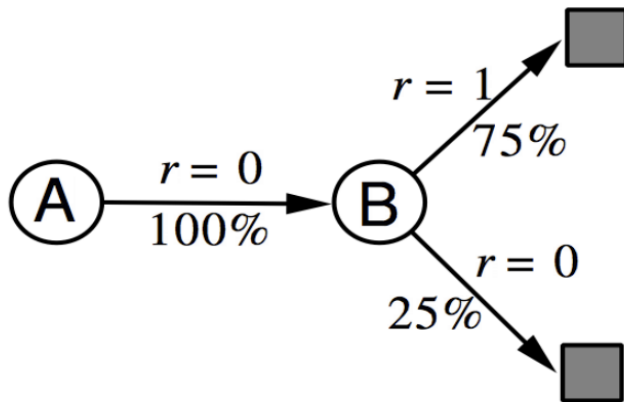
B, 1

B, 1

A, 0, B, 0

What are $V(A)$ and $V(B)$?

AB Example



Certainty Equivalence

- MC converges to solution with minimum mean-squared error
 - In the AB example, $V(A) = 0$
- TD(0) converges to solution of max likelihood Markov model
 - In the AB example, $V(A) = 0.75$

MC and TD

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